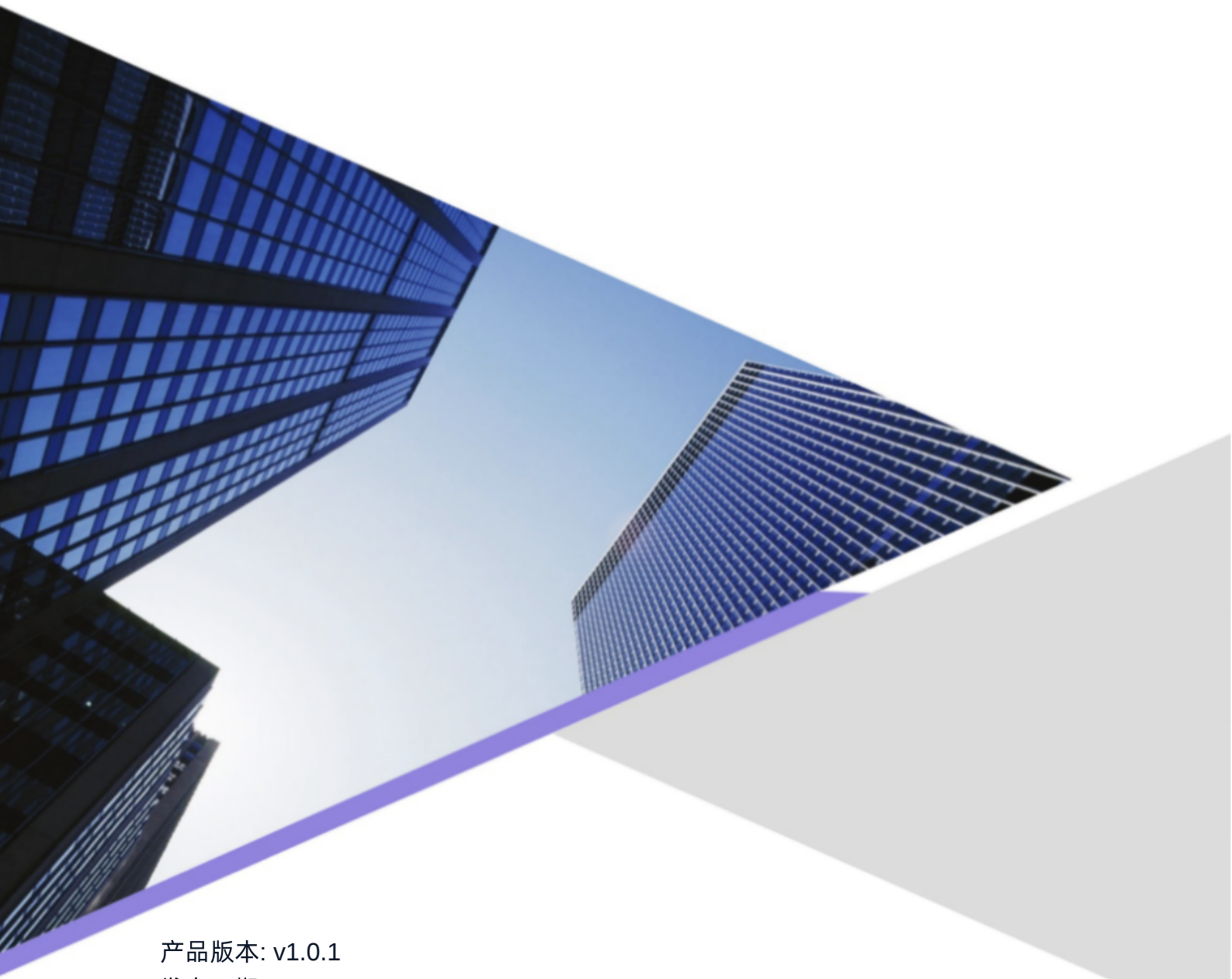
易捷行云
EasyStack

云原生云主机

使用手册



产品版本: v1.0.1
发布日期: 2023-06-20

目录

1 版本说明

1.1 版本说明书

版本信息

产品名称	产品版本	发布日期
	V1.0.1	2022-11-30

更新说明

新增功能

- 支持云原生云主机的创建/删除等生命周期管理
- 支持云原生云主机跨节点热迁移
- 支持给云原生云主机挂载Kubernetes 持久存储
- 支持云原生云主机使用统一SDN 网络
- 支持云原生云主机挂载高性能存储

依赖说明

> - 平台版本至少为v6.1.1。

2 产品介绍

2.1 什么是云原生云主机

云原生云主机（KubeVirt）基于原生Kubernetes，提供以容器为核心的虚拟化工作负载管理服务，支持将已有的虚拟化工作负载与容器化的工作负载结合于一个平台，支持在容器中与已有的虚拟化应用进行有交互的新微服务应用的开发。

产品优势

- **简单易用**

  通过Yaml文件一键部署，配置修改灵活。

- **与容器平台高度集成**

  通过kubectl和virtctl与容器平台高度集成，实现与容器共存。

- **灵活可配置**

  通过配置Yaml，灵活修改云主机配置。

- **VNC支持**

  支持使用VNC访问或创建云主机，无需额外配置终端。

主要功能

- **管理云主机电源**

  通过virtctl命令，不仅可以对已创建的云主机执行启动、暂停、关机、挂起或重启操作，实现无需采用IPMI便可简便管理云主机电源，也可以进行自动化的群集管理，简化系统管理员的操作流程。

- **管理云主机配置**

支持对云主机执行调整规格、调整启动顺序或编辑标签等操作，以灵活修改云主机配置，而不影响业务迁移。此外，对于企业管理员，仅需修改云主机的部分配置，即可实现配置管理，简单易用。

- **管理云主机存储**

  支持对云主机执行挂载/卸载云硬盘、挂载/卸载ISO、挂载/卸载USB设备等操作，以个性化管理云主机存储，满足不同种类镜像地存储和备份。此外，支持使用不同的启动顺序启动云主机。

- **管理云主机镜像**

  支持对云主机镜像执行创建、删除或编辑操作，以统一管理云主机镜像与云原生持久卷（PV），实现云主机镜像的可管理、可运维和可迁移，大大简化企业管理者对镜像的操作，使其内固化到持久卷（PV）中方便保存

2.2 使用场景

- **持续交付**

  基于自定义资源描述（CRD）配置，实现各类型云主机地持续交付。通过此操作可以灵活快速地创建各种类型云主机，并可以根据业务和环境变量对其进行配置，以帮助客户快速、灵活地配置业务云主机。

- **多盘热插拔**

  根据客户实际业务需求，酌情配置多个硬盘地热插拔。此操作适配于多种存储后端，可以灵活使用Kubernetes的持久卷声明（PVC）和持久卷（PV）资源，简化云主机资源地迁移和备份，也可以使用多种存储后端配置云主机资源。

2.3 基本概念

- **云原生云主机（KubeVirt）**

  KubeVirt重要组件包括virt-api、virt-controller、virt-handler、virt-launcher和Libvirtd。各组件的具体说明如下：

  - virt-api：是所有虚拟化相关处理流程的入口（Entry Point），负责更新和验证云主机的自定义资源描述（CRD），并提供RESTful API管理集群中云主机。
  - virt-controller：负责监控和管理集群中每个云主机及其相关Pod的状态。
  - virt-handler：负责监控每个云主机的状态变化，保持集群级云主机规格与相应libvirt域之间的同步，并报告Libvirt域状态和集群规格的变化。此外，还负责调用以节点为中心的插件以满足云主机规格定义的网络和存储要求。
  - virt-launcher：负责提供cgroups和名称空间，并负责托管云主机进程。
  - Libvirtd：virt-launcher借助于此组件管理云主机的生命周期。

2.4 产品获取

前提条件

在执行下述产品获取操作步骤前，请确保以下条件均已满足：

- 已成功获取并安装“Kubernetes容器服务”或“安全容器服务”云产品。获取并安装云产品的具体操作说明，请参考“产品与服务管理”帮助中的相关内容。
- 如需获取正式版云产品，请提前将已获取的许可文件准备就绪。

操作步骤

在顶部导航栏中，依次选择[产品与服务]-[产品与服务管理]-[云产品]，进入“云产品”页面获取并安装“云原生云主机”云产品。具体的操作说明，请参考“产品与服务管理”帮助中“云产品”的相关内容。

2.5 使用限制

- 目前，暂不支持对云主机执行编辑名称或创建快照操作。
- 云主机的热迁移操作依赖于csi的external provisioner对 **ReadWriteMany** 能力的支持。
- 云主机的编辑镜像配置操作依赖于csi的external provisioner对 **resize** 能力的支持。

2.6 权限说明

本章节主要用于说明云原生云主机各功能的用户权限范围。其中，√ 代表该类用户可对云平台内所有项目的操作对象执行此功能，**XX项目** 代表该类用户仅支持对XX项目内的操作对象执行此功能，未标注代表该类用户无权限执行此功能。

功能			云管理员	部门管理员/项目管理员/普通用户
镜像管理	上传		√	√
	删除			
云主机管理	创建		√	√
	查看所有			
	管理电源	启动		
		关机		
		暂停		
		恢复		
		重启		
	管理配置	调整规格		
		调整启动顺序		
	管理存储	热插拔		
	维护	热迁移		
		冷迁移		
	远程连接			
	删除			

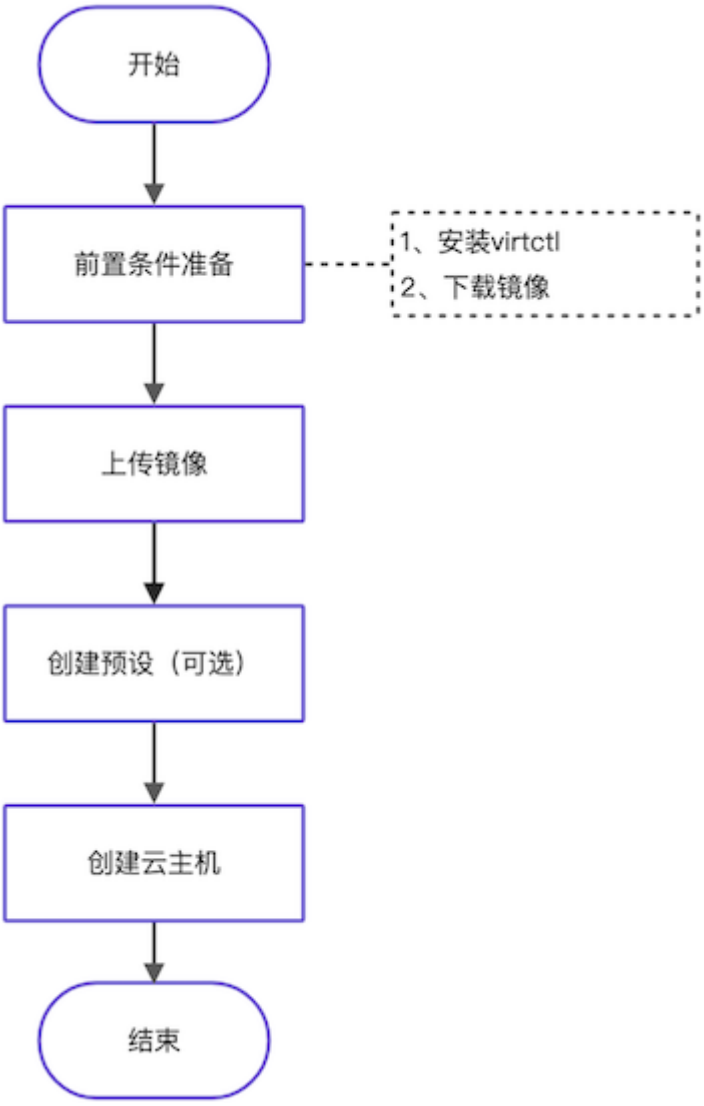
2.7 与其他服务的关系

服务	关系说明
Kubernetes容器服务/安全容器服务	提供承载云原生云主机运行的Kubernetes集群或安全容器集群。

3 快速入门

3.1 操作指引

云原生云主机云产品的主线使用流程及具体说明如下：



操作流程	描述

操作流程		描述
前置条件准备	安装virtctl	安装KubeVirt自带的命令行工具virtctl，以实现越过virt-launcher Pod层直接管理云主机。
	下载镜像	下载云主机创建时所需的镜像文件。
上传镜像		上传云主机创建时所需的镜像文件。
创建预设（可选）		为云主机创建不同计算和内存能力的规格模板，供云主机创建时使用。 请根据客户实际业务需求酌情配置。如无需通过预设创建云主机，可跳过本步骤。
创建云主机		依据客户实际业务需求，创建包含CPU、内存、操作系统、网络配置、云硬盘等基础资源的云原生云主机，为客户提供可靠、安全、灵活、高效的计算环境。

3.2 前置条件准备

在使用云原生云主机服务前，请先完成以下准备工作。

安装virtctl

virtctl是KubeVirt自带的类似kubectl的命令行工具，可以越过virt-launcher Pod层直接管理云主机，如控制云主机的启动、关机和重启等。

1. 获取virtctl安装包。具体命令如下:

```
wget
https://github.com/kubevirt/kubevirt/releases/download/v0.47.1/virtctl-
v0.47.1-linux-amd64
```

2. 修改文件执行权限。具体命令如下:

```
chmod a+x virtctl-v0.47.1-linux-amd64
```

3. 复制文件至 **/usr/local/bin/virtctl** 目录。具体命令如下:

```
cp virtctl-v0.47.1-linux-amd64 /usr/local/bin/virtctl
```

下载镜像

根据客户实际业务需求，下载对应类型的镜像文件。

- CentOS镜像
- Windows镜像

3.3 上传镜像

本操作用于预先上传云主机创建时所需要的镜像文件。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=general --wait-secs=240 --uploadproxy-url=https://cdi-uploadproxy --insecure
```

参数	说明
PVC名称	用于保存该镜像的持久卷声明（PVC）的名称。
PVC大小	用于保存该镜像的持久卷声明（PVC）的大小。 该参数值必须大于镜像转换后的大小。
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： */root/kubevirt-demo/img/CentOS-7-x86_64-GenericCloud-2009.qcow2* 。

3.4 创建云主机

本操作用于创建包含CPU、内存、操作系统、网络配置、云硬盘等基础资源的云原生云主机，为客户提供可靠、安全、灵活、高效的计算环境。

1. 根据客户实际业务需求，规划云主机配置项。各配置项的具体说明及格式如下：

   - CPU:

     ```
     spec:
       template:
         spec:
         domain:
             cpu:
             cores:
     ```

   - 内存:

     ```
     spec:
       template:
         spec:
         domain:
             resources:
                 requests:
                     memory: <内存大小>G
     ```

   - 存储：

     - 持久卷声明（PVC）：

       ```
       spec:
         template:
           spec:
             volumes:
               - name: <数据卷名称>
                   persistentVolumeClaim:
                       claimName:
       ```

- 云硬盘（ DV，DataVolume）：在云主机创建时，指定的DV会同步创建一个PVC供云主机使用。在云主机删除时，该PVC也会同步销毁:

```
spec:
  template:
    spec:
      volumes:
        - name: <数据卷名称>
            dataVolume:
                name:
```

- 云硬盘模板（ DV，DataVolume）：在云主机创建时，DV模板会同步创建一个DV供云主机使用。在云主机删除时，该DV连同PVC也会同步销毁。在下述示例中，该DV模板定义的数据来源为另一个PVC，但是也支持http等多种来源:

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

> 警告:
> 在同一命名空间下，请确保各DV和DV模板的名称均不重复，

- 系统盘:

```
spec:
  template:
    spec:
        domain:
            devices:
                disks:
                - disk:
                    bus: virtio
                name: <系统盘名称>
```

- 启动顺序:

```
spec:
  template:
    spec:
        domain:
            devices:
                disks:
                - bootOrder: 1
                    cdrom:
                    bus: <第一启动盘类型>
                    name: <第一启动盘名称>
                - bootOrder: 2
                    disk:
                    bus: <第二启动盘类型>
                    name: <第二启动盘名称>
```

- 网络:
  - 使用Pod默认网络:

```
spec:
  template:
    spec:
        networks:
        - name: default
          pod: {}
```

- 使用集群第二网络kube-ovn：

  当承载云原生云主机运行的是安全容器集群（即已安装安全容器服务云产品）时，可在创建云主机时，使用安全容器服务云产品默认安装的kube-ovn。在配置下述内容前，请确保已安装Multus服务，且在安全容器服务的命名空间下已配置NetworkAttachmentDefinition资源:

  ```
  spec:
    template:
      spec:
          networks:
          - name: default
            multus:
              default: true
              networkName: secure-container/kube-ovn
  ```

  当使用集群第二网络kube-ovn时，接口设置仅支持bridge模式。此外，集群第二网络kube-ovn需配合固定IP地址使用，否则会由于kube-ovn的内置IPAM分配问题，导致云主机在重启后会网络异常（其中，mac_address请自动生成，不可复用已使用的mac地址）：

  ```
  spec:
    template:
      metadata:
          annotations:
              ovn.kubernetes.io/ip_address: <固定IP地址>
              ovn.kubernetes.io/mac_address: 00:00:00:63:D5:F9
  ```

- 接口：

  - masquerade模式（默认）：

    ```
    spec:
      template:
        spec:
            domain:
                devices:
                    interfaces:
                    - masquerade: {}
                    name: default
    ```

- bridge模式:

```
  spec:
    template:
      spec:
          domain:
              devices:
                  interfaces:
                  - bridge: {}
                  name: default
```

> 警告:
> - 当使用集群第二网络kube-ovn时，仅支持bridge模式。
> - 当使用Pod默认网络对接bridge模式时，请在KubeVirt提供的permitBridgeInterfaceOnPodNetwork配置项中禁用该关联，否则云主机将无法成功创建。

- 密钥/密码注入:

```
spec:
  template:
    spec:
        domain:
            devices:
                disks:
                - disk:
                bus: virtio
                name: cloudinitdisk
        volumes:
            - 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
```

2. 通过Yaml，创建云主机。

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:
            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:
```

- 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: win2k19-boot-dv
spec:
    running: true
    template:
    metadata:
        labels:
            kubevirt.io/domain: win2k19-boot-dv
    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:
```

4 用户指南

4.1 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： */root/kubevirt-demo/img/CentOS-7-x86_64-GenericCloud-2009.qcow2* 。
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>`

4.2 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:
            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$jCog26PDU3r4BKBxFS3bejiglTyTyvldqK7hiExVP8rgbIbJh3CDX
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:

c3NoLXJzYSBBQUFBQjNOemFDMXljMkVBQUFBREFRQUJBQUFCQVFEZVpuQW1MMUR4SDFsOXQ
wWUswNXV2bC9jZ3QwbytEMEdQZnN2SFg3Qm8vYlVuMzlyVXY3K3cvOFkvQnZXZ0RaZ2V4cj
J2a0E5NTJMRjBzekduL3VZa2VzVEJTdWVVamVhbjJvZGZkYmpPTnJ1RVdXVGM5ODBHaENPd
zgzYUR5bnc4VVA3d3BYOWlHWC9ZVWprTEY1YnN3UERpMlVsQkR4U015dzRzazAyTE5KdXRE
ajBEV0N4ZkcyK0lxOUVIUXM0ZVhvbHh4bFRHVjJVb1NlVlk0cU85V2JUUTF1QzZId3BTWkV
CbWdTb2U3eTVXMXUwSFU1TUpvbXJzWGFYTUx5RU1wdU5ub3hYVS82RGg4V0dJdGs0Mi8wUE
YwS21vdE0rbjdLS3NXSm9kaFR5NStOaDRaa0tZSThNOGN2aFZQQU1zQ0s3OEdwUTdFblZqQ
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>
```

4.3 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-servciemonitor
  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1号楼1层107-2号
南京易捷思达软件科技有限公司：
江苏省南京市雨花台区软件大道168号润和创智中心4栋109-110

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)