Virtualization

Overview

Introduction

Container-Orchestrated Virtual Machine Solution

Features

Product Features

Constraints and Limitations

Install

Install

Prerequisites

Procedure

Resource Quota Explanation

Images

Introduction

Advantages

Guides

Permissions

How To

Virtual Machine

Introduction

Guides

How To

Troubleshooting

Network

Introduction

Advantages

Guides

How To

Storage

Introduction

Advantages

Guides

Backup and Recovery

Introduction

Application Scenarios

Usage Limitations

Guides

Overview

Introduction

Introduction

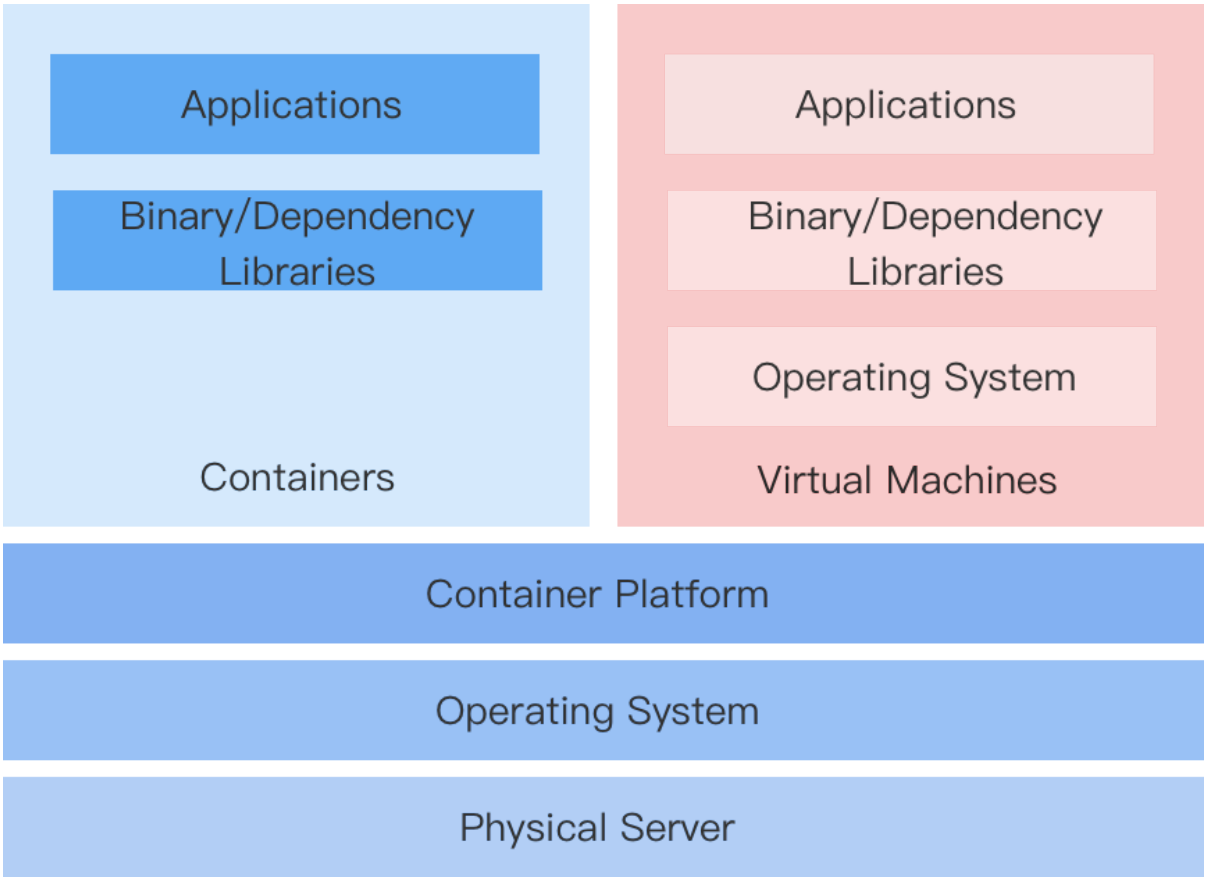
For enterprises using a virtual machine-based architecture, transitioning to a Kubernetes and container-based architecture inevitably requires application modernization. However, due to constraints such as the need for continuous business uptime or the difficulty in changing development habits, enterprises often cannot completely disengage from virtualization architecture in a short period.

Therefore, a solution that can uniformly configure, manage, and control container resources and virtual machine resources on the same platform becomes particularly important.

TOC

Container-Orchestrated Virtual Machine Solution

This platform implements a virtual machine (VMI, VirtualMachineInstance) solution based on the open-source component KubeVirt, allowing for easier and faster creation of container-orchestrated virtual machines and running virtualized applications.



Features

Rapid Transformation

There is no need to rewrite applications or modify images. Simply package the existing application into a qcow2 or raw format virtual machine image, and create a virtual machine using that image on the platform, allowing the application to be deployed to the container platform.

Maintain Behavioral Habits

Containerized virtual machines can be managed using a similar approach to traditional virtual machines, without needing to focus on the underlying container implementation, including virtual machine lifecycle management, disks and networks, and snapshot management.

Coexistence of Virtualization and Containerization

- The unified platform supports managing virtualized services while also enabling Kubernetes-based container scheduling and management.

- On the basis of continuing to use virtual machine workloads, it allows for a gradual modernization of containerized applications.
- The development of new containerized applications that need to interact with virtualized applications remains unaffected.

Product Features

- **Virtual Machine**: Supports creating virtual machines with images allocated by administrators and managing them, including starting and stopping virtual machines, managing snapshots, remote login to virtual machines, and modifying virtual machine configurations.
- **Virtual Disk**: Supports viewing and managing disk information created in the current project, including creating disks, viewing disk names, storage classes, capacities, and associated virtual machines.
- **Virtual Machine Snapshots**: Supports viewing details such as the status of virtual machine snapshots, the associated virtual machine, and the most recent rollback time.
- **Virtual Machine Images**: Supports viewing virtual machine image information under the current project, including image provision method and operating system.
- **Key Pairs**: Supports viewing and managing key pairs created in the current project, including creating key pairs and viewing the list of associated virtual machines.

Constraints and Limitations

It must be implemented based on a physical machine cluster, and KubeVirt components must be deployed within the cluster with virtualization enabled. The platform provides the capability to deploy KubeVirt components via Operator and an interface to enable virtualization, with all related configurations completed by the platform administrator.

Install

In order for project personnel to fully utilize virtualization features within the container platform, the platform administrator must perform the following operations to prepare the virtualization environment.

TOC

Prerequisites

- **Download** the **ACP Virtualization with KubeVirt** installation package corresponding to your platform architecture.
- **Upload** the **ACP Virtualization with KubeVirt** installation package using the Upload Packages mechanism.

- When using virtualization features, it is necessary to plan and prepare the network and storage environments in advance.
  Note:

  - If you need to connect to the virtual machine directly via IP, the cluster must use the Kube-OVN Underlay network mode. You can refer to the best practices Preparing Kube-OVN Underlay Physical Network.
  - It is recommended to use TopoLVM with Kubevirt, as it can provide near-hardware-level performance. If performance requirements are not high, Ceph distributed storage can also be used.

Storage Product	Description
TopoLVM	**Advantages**: Relatively lightweight and good performance. **Disadvantages**: Cannot be used across nodes, has low reliability, and cannot provide redundancy.
Ceph Distributed Storage	**Advantages**: Can be used across nodes, highly available, and has redundancy. **Disadvantages**: Redundant disk copies lead to lower utilization; performance is poorer.

    - If TopoLVM is used and multiple disks are configured, please ensure that the remaining storage capacity on the virtualization-enabled nodes can meet the total capacity of the multiple disks; otherwise, the virtual machine creation will fail.
    - If Ceph distributed storage is being used, please ensure that the network where the storage resides and the network where the virtual machines reside can communicate with each other.

Procedure

1 Enabling Node Virtualization

When the nodes of a self-built cluster are **physical machines**, you can control whether to allow Kubernetes to schedule Virtual Machine Instances (VMIs) on that node by

enabling or disabling the node virtualization switch.

- When the switch is enabled, new virtual machines are allowed to be scheduled on the physical machine node; Windows physical nodes do not support enabling virtualization.
- When the switch is disabled, new virtual machines are prevented from being scheduled on the physical machine node, but it does not affect virtual machines that are already running on that node.

Procedure

1. Enter **Administrator**.
2. In the left navigation bar, click **Cluster Management** > **Clusters**.
3. Click ***Self-Built Cluster Name***.
4. On the **Nodes** tab, click the ⋮ to the right of the node where you want to set the virtualization switch > **Enable Virtualization**.
5. Click **Confirm**.

2 Deploying Operator

1. Login, go to the **Administrator** page.
2. Click **Marketplace** > **OperatorHub** to enter the **OperatorHub** page.
3. Find the **ACP Virtualization with KubeVirt**, click **Install**, and navigate to the **Install ACP Virtualization with KubeVirt** page.
   Configuration Parameters:

Parameter	Recommended Configuration
Channel	The default channel is `alpha`.
Installation Mode	`Cluster`: All namespaces in the cluster share a single Operator instance for creation and management, resulting in lower resource usage.

Parameter	Recommended Configuration
Installation Place	Select `Recommended`, Namespace only support **kubevirt**.
Upgrade Strategy	`Manual`: When there is a new version in the Operator Hub, manual confirmation is required to upgrade the Operator to the latest version.

3 Creating a HyperConverged Instance

1. Enter **Administrator**.
2. Click **Marketplace** > **OperatorHub**.
3. Find the **ACP Virtualization with KubeVirt**, click it to enter the **ACP Virtualization with KubeVirt** detail info page.
4. Click **All Instances**
5. Click **Create Instance** on the **HyperConverged** instance card.
   Note: Only one **HyperConverged** instance needs to be created in each cluster.
6. Switch to YAML view and only replace the ***placeholder*** specified in the `spec.storageImport.insecureRegistries` field in the example with the correct **virtual machine image repository address**, for example: `192.168.16.214:60080`, keeping other parameters at their default values.

```yaml
spec:
  storageImport:
    insecureRegistries:
      - placeholder
```

Replacement result:

```yaml
spec:
  storageImport:
    insecureRegistries:
      - '192.168.16.214:60080'
```

7. Click **Create** and wait for the CDI and KubeVirt type instances to be automatically created in the resource list, while ensuring that the **status.phase** displayed in the YAML is `deployed`, indicating that the HyperConverged instance has been successfully created.

4 Configuring Virtual Machine Overcommit Ratio (Optional)

- Configuring the overcommit ratio for the cluster where the virtual machines reside in **Cluster Management** > **Clusters**.
- Or Configuring the overcommit ratio for the namespace where the virtual machines are located in **Project Management** > **Namespaces**.

Important Notes

- Virtual machines only support CPU overcommit ratios, and the recommended configuration value is between 2 and 4.
- Once the overcommit ratio is enabled for virtual machines, when creating a virtual machine, the container's request value (requests) is fixed as **specified limit value (limits) / VM overcommit ratio**, making the user's request set through YAML ineffective.
  For example: Assuming the CPU resource overcommit ratio is set to 4 for the virtual machine, if the user specifies a CPU limit value of 4c when creating the virtual machine, the CPU request value would be 4c/4 = 1c.

Resource Quota Explanation

The memory resource quota for virtual machines is limited by the memory resource quota of the namespace they reside in. Since the memory of the Pod hosting the virtual machine is usually larger than the actual available memory of the virtual machine, it is recommended to reserve 20% of the resources. When the remaining available resources in the namespace are below 20%, please promptly scale up the resources.

Images

Introduction

Introduction

Advantages

Guides

Adding Virtual Machine Images

Procedure

Update/Delete Virtual Machine I

Update/Dele

How To

Creating Windows Images Base

Prerequisites

Constraints and Limitations

Procedure

Creating Linux Images Based o

Prerequisites

Constraints and Limitations

Procedure

Exporting V

Procedure

Remote Access

Permissions

Permissions

Introduction

Alauda Container Platform Virtualization with KubeVirt leverages Kubernetes extended API capabilities to abstract virtual machine images as a **Custom Resource Definition** (CRD). It provides a user interface (UI) for users to easily import virtual machine images stored in remote repositories into ACP for usage.

TOC

Advantages

- Support for Mainstream Operating Systems
  Supports various commonly used Linux distributions and Windows operating systems.
- Multi-Architecture Support
  Compatible with both **X86_64** and **ARM64** architectures.
- Multi-Source Support
  Allows importing virtual machine images from:
  - Image registries
  - File servers
  - S3-compatible object storage
- Multi-Format Support
  Supports virtual machine images in **QCOW2** and **RAW** formats.

Guides

Adding Virtual Machine Images Update/Delete Virtual Machine I Update/Dele

Procedure

Adding Virtual Machine Images

The platform supports adding **X86_64** and **ARM64 (Alpha)** architecture virtual machine images, enabling developers to quickly create virtual machines for existing services and facilitate the migration of business systems.

TOC

Procedure

1. Access **Administrator**.
2. In the left navigation bar, click **Virtualization Management** > **Virtual Machine Images**.
3. Click **Add Virtual Machine Image**.
4. Refer to the instructions below to configure the relevant parameters.

Parameter	Description
Provisioning Method	Currently, only **Public Image** method is supported, meaning the added image can be used in assigned projects.
Operating System	Supported operating systems include: **CentOS/Ubuntu/RedHat/Debian/TLinux/Other Linux/Windows**

Parameter	Description
	(Alpha). Supported system architectures are: **X86_64** and **ARM64 (Alpha)**.
Source	• **Image Repository**: Virtual machine images stored in a container image repository. • **HTTP**: Virtual machine images stored on a file server using the HTTP protocol. • **Object Storage (S3)**: Virtual machine images that can be retrieved using Object Storage Protocol (S3). If they do not require authentication, please use HTTP as the source.
CPU Architecture	Tag CPU architecture information. For image repository sources, multiple selections are supported; for other sources, only single selection is allowed.
Image Address	Supports KVM virtual machine images, including qcow2/raw formats. • If from an image repository, enter `repository_address:image_version`, e.g., `registry.example.com/library/ubuntu:latest`. • If from an HTTP source, enter the image file URL, which must start with `http://` or `https://`, e.g., `http://192.168.0.1/vm_image/centos_7.8.qcow2`. • If from an Object Storage (S3), enter the image address that can be retrieved via Object Storage Protocol (S3), e.g., `https://endpoint/bucket/centos.qcow2`.

Parameter	Description
Authentication	Depending on whether the image repository requires authentication, you can toggle the switch on or off. If enabled, you can choose from existing image credentials or click **Add Credentials**, supporting only **Username/Password** type credentials. **Note**: When the source is Object Storage (S3), authentication cannot be turned off.
Assigned Project	Assign usage permissions for this image to projects. • All Projects: Assigns usage permissions of the image to all projects. • Specific Project: Assigns usage permissions of the image to a specified project. • No Assignment: Do not assign to any projects for now. After image creation, you can assign it through **Update Image** operation.

5. Click **Add**.

Update/Delete Virtual Machine Images

1. Navigate to **Administrator**.
2. In the left sidebar, click **Virtualization Management** > **Virtual Machine Images**.
3. Click ⋮ > **Update/Delete**.
4. After confirmation, click **Update/Delete**.

Update/Delete Image Credentials

1. Go to **Administrator**.
2. In the left navigation bar, click **Virtualization Management** > **Virtual Machine Images**.
3. In the **Image Credentials** tab, click ⋮ > **Update/Delete**.
4. After confirmation, click **Update/Delete**.

How To

Creating Windows Images Based on ISO using KubeVirt

This document discusses a virtual machine solution based on the open-source component KubeVirt, using KubeVirt virtualization technology to create a Windows operating system image through an ISO image file.

TOC

Prerequisites

- All components in the cluster are functioning correctly.

- Please prepare the Windows image and the latest virtio-win-tools ↗ in advance.
- Please prepare the repository for storing the image. This document takes the build-harbor.example.cn repository as an example, and please replace it according to your actual environment.

Constraints and Limitations

- When starting KubeVirt, the size of the custom image's filesystem will affect the speed of writing the image to the disk in PVC. If the filesystem is too large, it may result in extended creation times.
- It is recommended to keep the Linux root partition or Windows C drive below 100G to minimize the initial size. Subsequent expansion can be done through cloud-init (for Windows systems, it must be expanded manually after creation).

Procedure

1 Create Image

Create a container image from the prepared Windows and virtio-win ISO images, and push it to the repository. This document uses Windows Server 2019 as an example.

Create a Container Image from the Windows ISO

1. Navigate to the directory where the ISO image is stored, and execute the following command in the terminal to rename the ISO image to win.iso.

```
mv <ISO image name> win.iso # Replace <ISO image name> with the actual image name, e.g., mv en_windows_server_2019_x64_dvd_4cb967d8.iso win.iso
```

2. Execute the following command to create a Containerfile.

```
touch Containerfile
```

3. Edit the Containerfile, add the following content, and save it.

```
FROM scratch
ADD --chown=107:107 win.iso /disk/
```

4. Execute the following command to build the container image.

```
podman build -t build-harbor.example.cn/3rdparty/vmdisks/winiso:2019 -f Containerfile . # Replace the repository according to your actual environment
```

5. Execute the following command to push the image to the repository.

```
podman push build-harbor.example.cn/3rdparty/vmdisks/winiso:2019 # Replace the repository according to your actual environment
```

Create a Container Image from the virtio-win ISO

1. Navigate to the directory where the ISO image is stored, and execute the following command in the terminal to create a Containerfile.

```
touch Containerfile
```

2. Edit the Containerfile, add the following content, and save it.

```
FROM scratch
ADD --chown=107:107 virtio-win.iso  /disk/
```

3. Execute the following command to build the container image.

```
podman build -t build-harbor.example.cn/3rdparty/vmdisks/win-virtio:latest -f Containerfile . # Replace the repository according to your actual environment
```

4. Execute the following command to push the image to the repository.

```
podman push build-harbor.example.cn/3rdparty/vmdisks/win-virtio:la
test # Replace the repository according to your actual environment
```

2 Create Virtual Machine

1. Access the **Container Platform**.
2. In the left navigation bar, click on **Virtualization** > **Virtual Machines**.
3. Click on **Create Virtual Machine**.
4. Fill in the necessary parameters such as **Name**, **Image**, etc., in the form page. For detailed parameters and configuration, please refer to Create Virtual Machine.
5. Switch to YAML.
6. Replace the configuration under the spec.template.spec.domain.devices.disks field with the following content.

```
domain:
  devices:
    disks:
      - disk:
          bus: virtio
        name: cloudinitdisk
      - bootOrder: 1
        cdrom:
          bus: sata
        name: containerdisk
      - cdrom:
          bus: sata
        name: virtio
      - disk:
          bus: sata
        name: rootfs
        bootOrder: 10
```

7. Add the following content under the spec.template.spec.volumes field.

```
        - containerDisk:
            image: registry.example.cn:60070/3rdparty/vmdisks/wini
so:2019 # Replace the image according to your actual environment
          name: containerdisk
        - containerDisk:
            image: registry.example.cn:60070/3rdparty/vmdisks/win-
virtio # Replace the image according to your actual environment
          name: virtio
```

8. Check the YAML file. The complete YAML after finishing the configuration is as follows.

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  annotations:
    cpaas.io/creator: test@example.io
    cpaas.io/display-name: ""
    cpaas.io/updated-at: 2024-09-01T14:57:55Z
    kubevirt.io/latest-observed-api-version: v1
    kubevirt.io/storage-observed-api-version: v1
  generation: 16
  labels:
    virtualization.cpaas.io/image-name: debian-2120-x86
    virtualization.cpaas.io/image-os-arch: amd64
    virtualization.cpaas.io/image-os-type: debian
    virtualization.cpaas.io/image-supply-by: public
    vm.cpaas.io/name: aa-test
  name: aa-test
  namespace: acp-service-self
spec:
  dataVolumeTemplates:
    - metadata:
        creationTimestamp: null
        labels:
          vm.cpaas.io/reclaim-policy: Delete
          vm.cpaas.io/used-by: aa-test
        name: aa-test-rootfs
      spec:
        pvc:
          accessModes:
            - ReadWriteOnce
          resources:
            requests:
              storage: 100Gi
          storageClassName: vm-cephrbd
          volumeMode: Block
        source:
          http:
            url: http://192.168.254.12/kube-debian-12.2.0-x86-out.
qcow2
  running: true
  template:
    metadata:
      annotations:
```

```
    cpaas.io/creator: test@example.io
    cpaas.io/display-name: ""
    cpaas.io/updated-at: 2024-09-01T14:55:44Z
    kubevirt.io/latest-observed-api-version: v1
    kubevirt.io/storage-observed-api-version: v1
  creationTimestamp: null
  labels:
    virtualization.cpaas.io/image-name: debian-2120-x86
    virtualization.cpaas.io/image-os-arch: amd64
    virtualization.cpaas.io/image-os-type: debian
    virtualization.cpaas.io/image-supply-by: public
    vm.cpaas.io/name: aa-test
spec:
  affinity:
    nodeAffinity: {}
  architecture: amd64
  domain:
    devices:
      disks:
        - disk:
            bus: virtio
          name: cloudinitdisk
        - bootOrder: 1
          cdrom:
            bus: sata
          name: containerdisk
        - cdrom:
            bus: sata
          name: virtio
        - disk:
            bus: sata
          name: rootfs
          bootOrder: 10
       interfaces:
        - bridge: {}
          name: default
    machine:
      type: q35
    resources:
      limits:
        cpu: "4"
        memory: 8Gi
      requests:
        cpu: "4"
```

```
            memory: 8Gi
      networks:
        - name: default
          pod: {}
      nodeSelector:
        kubernetes.io/arch: amd64
        vm.cpaas.io/baremetal: "true"
      volumes:
        - cloudInitConfigDrive:
            userData: >-
              #cloud-config
              disable_root: false
              ssh_pwauth: true
              users:
                - default
                - name: root
                  lock_passwd: false
                  hashed_passwd: $6$0vlhl57e$0rawYwaeu9jL6hBf3XP9l
k6XXaMUS9/W6LPbWRinUoXujo39lP3l98VOcOObtr.LDoAv/ylm85FLQmxwNlWFe/
          name: cloudinitdisk
        - containerDisk:
            image: registry.example.cn:60070/3rdparty/vmdisks/wini
so:2019 # Replace the image according to your actual environment
          name: containerdisk
          - containerDisk
             image: registry.example.cn:60070/3rdparty/vmdisks/win
-virtio # Replace the image according to your actual environment
           name: virtio
         - dataVolume:
            name: aa-test-rootfs
          name: rootfs
```

9. Click **Create**.
10. Click **Actions** > **VNC Login**.
11. When the prompt **press any key boot from CD or DVD** appears, press any key to enter the Windows installation program; if you do not see the prompt, click on **Send Remote Command** in the top left of the page, then select **Ctrl-Alt-Delete** from the dropdown menu to restart the server.
    Note: If a message appears at the top of the virtual machine details page stating **The current virtual machine has configuration changes that require a restart to take effect, please restart**, this message can be ignored; no restart is necessary.

3 Install Windows Operating System

1. Follow the installation instructions to install the system after entering the installation page.
   Note: During the partition selection step, the bus must be sata for the disk to be correctly recognized. Therefore, you need to select each partition in turn and click **Delete** to remove all partitions, allowing the system to handle it automatically.
2. After configuring the administrator account password, click **Send Remote Command** in the top left of the page, then select **Ctrl-Alt-Delete** from the dropdown menu.
3. When prompted **The Ctrl+Alt+Delete combination will restart the server, confirm to restart**, click **OK**.
4. Enter the password to access the Windows system desktop; at this point, the Windows operating system installation is complete.

4 Install virtio-win-tools

This tool primarily contains the necessary drivers.

1. Open File Explorer.
2. Double-click **CD Drive(E:) virtio-win-<version>**, run the **virtio-win-guest-tools** directory to enter the installation page, and follow the installation instructions. The <version> part should be based on the actual situation.
3. After the installation is complete, power off the Windows system.

5 Export Custom Windows Image

Please refer to Export Virtual Machine Image for the specific operation.

6 Use Windows Image

1. Access the **Container Platform**.
2. In the left navigation bar, click on **Virtualization** > **Virtual Machines**.
3. Click on **Create Virtual Machine**.
4. Fill in the necessary parameters on the form page. For the image, select the exported Windows image. For detailed parameters and configuration, please refer to Create

Virtual Machine.

5. (Optional) If using a newer operating system, such as Windows 11, enable features like clock, UEFI, TPM, etc. Switch to YAML and replace the original YAML file with the following YAML file.

```
apiVersion: kubevirt.io/v1
kind: VirtualMachineInstance
metadata:
  labels:
    special: vmi-windows
  name: vmi-windows
spec:
  domain:
    clock:
      timer:
        hpet:
          present: false
        hyperv: {}
        pit:
          tickPolicy: delay
        rtc:
          tickPolicy: catchup
      utc: {}
    cpu:
      cores: 2
    devices:
      disks:
      - disk:
          bus: sata
        name: pvcdisk
      interfaces:
      - masquerade: {}
        model: e1000
        name: default
      tpm: {}
    features:
      acpi: {}
      apic: {}
      hyperv:
        relaxed: {}
        spinlocks:
          spinlocks: 8191
        vapic: {}
      smm: {}
    firmware:
      bootloader:
        efi:
          secureBoot: true
```

```
      uuid: 5d307ca9-b3ef-428c-8861-06e72d69f223
    resources:
      requests:
        memory: 4Gi
  networks:
  - name: default
    pod: {}
  terminationGracePeriodSeconds: 0
  volumes:
  - name: pvcdisk
    persistentVolumeClaim:
      claimName: disk-windows
  - name: winiso
    persistentVolumeClaim:
      claimName: win11cd-pvc
```

6. Click **Create**.

7 Add Internal Route

By configuring a NodePort type internal route, expose the port for remote desktop connections.

1. Access the **Container Platform**.
2. In the left navigation bar, click on **Virtualization** > **Virtual Machines**.
3. Click on the virtual machine name created with the Windows image in the list to enter the details page.
4. Click on the **Add** icon next to **Internal Route** in the **Login Information** area.
5. Configure parameters according to the following instructions.

Parameter	Description
Type	Select **NodePort**.

Parameter	Description
Port	• Protocol: Select TCP. • Service Port: Use 3389. • Virtual Machine Port: Use 3389. • Service Port Name: Use rdp.

6. Click **OK** to return to the details page.
7. Click on the **Internal Route** link in the **Login Information** area.
8. Save the **Virtual IP** information in the basic information area and the **Host Port** information in the port area.

Remote Access

This document discusses using the Windows operating system for remote connection as an example. Other operating systems can use software that supports the RDP protocol for connection.

1. Open **Remote Desktop Connection**.
2. Enter the saved Virtual IP and Host Port from the **Add Internal Route** step, formatted as **Virtual IP:Host Port**, for example: 192.1.1.1:3389 .
3. Click **Connect**.

Creating Linux Images Based on ISO Using KubeVirt

This document describes a virtual machine solution implemented based on the open-source component KubeVirt. It utilizes KubeVirt virtualization technology to create a Linux operating system image from an ISO image file.

TOC

Prerequisites

- All components in the cluster are functioning properly.
- A Linux image should be prepared in advance. This document uses the Ubuntu operating system ↗ as an example.

- A repository for storing images should be prepared in advance. This document uses the build-harbor.example.cn repository as an example; please replace it according to your actual environment.

Constraints and Limitations

- When starting KubeVirt, the file system size of the custom image will affect the speed of writing the image to the PVC disk. If the file system is too large, it may result in a prolonged creation time.
- It is recommended to keep the Linux root partition size below 100G to minimize the initial size. After configuring cloud-init, allocate larger storage for the root partition when creating the virtual machine, and the system will automatically expand it.

Procedure

1 Convert Linux ISO Image into Container Image

1. Navigate to the directory where the ISO image is stored and execute the following command in the terminal to rename the ISO image to ubuntu.iso.

```
mv <ISO image name> ubuntu.iso # Replace <ISO image name> with the actual image name, e.g., mv ubuntu-24.04-live-server-amd64.iso ubuntu.iso
```

2. Create a Containerfile by executing the following command.

```
touch Containerfile
```

3. Edit the Containerfile, add the following content, and save it.

```
FROM scratch
ADD --chown=107:107 ubuntu.iso /disk/
```

4. Build the container image by executing the following command.

```
podman build -t build-harbor.example.cn/3rdparty/vmdisks/ubuntu-iso:24.04 . # Please replace the repository according to your actual environment
```

5. Push the image to the repository by executing the following command.

```
podman push build-harbor.example.cn/3rdparty/vmdisks/ubuntu-iso:24.04 # Please replace the repository according to your actual environment
```

2 Create Virtual Machine

1. Enter the **Container Platform**.
2. Click **Virtualization** > **Virtual Machines** in the left navigation bar.
3. Click **Create Virtual Machine**.
4. Fill in the parameters on the form page as follows. For specific parameters and configurations, please refer to Create Virtual Machine.

Parameter	Description
Select Image	Choose the template image for the virtual machine.
IP Address	Keep default, which will be obtained via **DHCP**.
Network Mode	Use **NAT** mode; do not use **bridged** mode here.

5. Switch to YAML.
6. Replace the configuration under the spec.template.spec.domain.devices.disks field with the following content.

```
      domain:
        devices:
          disks:
            - bootOrder: 1
              cdrom:
                bus: sata
              name: containerdisk
            - disk:
                bus: virtio
              name: cloudinitdisk
            - disk:
                bus: virtio
              name: rootfs
              bootOrder: 10
```

7. Add the following content under the spec.template.spec.volumes field.

```
        - containerDisk:
            image: registry.example.cn:60070/3rdparty/vmdisks/ubuntu-iso:24.04 # Please replace the image according to your actual environment
          name: containerdisk
```

8. Review the YAML file; the complete YAML configuration after completion is as follows.

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  annotations:
    kubevirt.io/latest-observed-api-version: v1
    kubevirt.io/storage-observed-api-version: v1
  labels:
    virtualization.cpaas.io/image-name: debian-2120-x86
    virtualization.cpaas.io/image-os-arch: amd64
    virtualization.cpaas.io/image-os-type: debian
    virtualization.cpaas.io/image-supply-by: public
    vm.cpaas.io/name: aa
  name: aa
spec:
  dataVolumeTemplates:
    - metadata:
        creationTimestamp: null
        labels:
          vm.cpaas.io/reclaim-policy: Delete
          vm.cpaas.io/used-by: aa
        name: aa-rootfs
      spec:
        pvc:
          accessModes:
            - ReadWriteOnce
          resources:
            requests:
              storage: 100Gi
          storageClassName: vm-cephrbd
          volumeMode: Block
        source:
          http:
            url: http://192.168.254.12/kube-debian-12.2.0-x86-out.
qcow2
  running: true
  template:
    metadata:
      annotations:
        cpaas.io/creator: test@example.io
        cpaas.io/display-name: ""
        cpaas.io/updated-at: 2024-09-09T03:49:08Z
        kubevirt.io/latest-observed-api-version: v1
        kubevirt.io/storage-observed-api-version: v1
```

```
    creationTimestamp: null
    labels:
      virtualization.cpaas.io/image-name: debian-2120-x86
      virtualization.cpaas.io/image-os-arch: amd64
      virtualization.cpaas.io/image-os-type: debian
      virtualization.cpaas.io/image-supply-by: public
      vm.cpaas.io/name: aa
  spec:
    accessCredentials:
      - sshPublicKey:
          propagationMethod:
            qemuGuestAgent:
              users:
                - root
          source:
            secret:
              secretName: test-xeon
    affinity:
      nodeAffinity: {}
    architecture: amd64
    domain:
      devices:
        disks:
          - bootOrder: 1
            cdrom:
              bus: sata
            name: containerdisk
          - disk:
              bus: virtio
            name: cloudinitdisk
          - disk:
              bus: virtio
            name: rootfs
            bootOrder: 10
         interfaces:
          - bridge: {}
            name: default
      machine:
        type: q35
      resources:
        limits:
          cpu: "1"
          memory: 2Gi
        requests:
```

```
            cpu: "1"
            memory: 2Gi
      networks:
        - name: default
          pod: {}
      nodeSelector:
        kubernetes.io/arch: amd64
        vm.cpaas.io/baremetal: "true"
      volumes:
        - containerDisk:
            image: registry.example.cn:60070/3rdparty/vmdisks/ubuntu-iso:24.04 # Please replace the image according to your actual environment
          name: containerdisk
        - cloudInitConfigDrive:
            userData: |-
              #cloud-config
              disable_root: false
              ssh_pwauth: false
              users:
                - default
                - name: root
                  lock_passwd: false
                  hashed_passwd: ""
          name: cloudinitdisk
        - dataVolume:
            name: aa-rootfs
          name: rootfs
```

9. Click **Create**.
10. Click **Actions** > **VNC Login**.
11. When prompted with **press any key boot from CD or DVD**, press any key to enter the Windows installation program; if you do not see the prompt, click **Send Remote Command** in the upper left corner of the page, and then click **Ctrl-Alt-Delete** from the dropdown menu to reboot the server.
    Note: If a message appears at the top of the virtual machine detail page stating **Current virtual machine has configuration changes that require a restart to take effect. Please restart.**, you can ignore this message; a restart is not necessary.

3 Install Linux Operating System

1. After entering the installation page, follow the installation guide to proceed. This document gives an example of installing the Ubuntu operating system; the configuration items during the installation process of different operating systems are generally similar, and thus will not be elaborated further. Some configuration items are explained below.

Configuration	Description
Installation Type	It is recommended to use a minimal installation to minimize the image size.
Storage Configuration	Choose custom storage. Format the disk to ext4 or xfs format and mount it to the root partition (/). **Note**: Do not use LVM for disk partitioning (Create volume group (LVM)).
SSH Configuration	Choose to install the OpenSSH tools for SSH access.

2. Wait for the installation to complete.

4 Modify YAML File

1. Enter the **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machines**.
3. Click on the **Virtual Machine Name** in the list to enter the details page.
4. Click **Stop**.
5. Click **Actions** > **Update** in the upper right corner.
6. Switch to YAML.
7. Confirm that the disk named **rootfs** under spec.template.spec.domain.devices.disks has a bootOrder of 1. If it is not 1, modify it to 1.
8. Delete the relevant content for the disk named **containerdisk** under spec.template.spec.domain.devices.disks; the specific content to delete is as follows.

```
            - bootOrder: 1
              cdrom:
                bus: sata
              name: containerdisk
```

9. Delete the relevant content for the disk named **containerdisk** under spec.template.spec.volumes; the specific content to delete is as follows.

```
        - containerDisk:
            image: registry.example.cn:60070/3rdparty/vmdisks/ubun
tu-iso:24.04
          name: containerdisk
```

10. Click **Update**.
11. Click **Start**.

5 Install Required Software and Modify Configuration

Note: The following commands and configuration files may vary slightly between different operating systems; please adjust according to your actual environment.

1. Enter your username and password to log in to the operating system.
2. Switch to root user privileges.
3. Install the software packages.
   - For CentOS series, execute the command:

     ```
     yum install cloud-utils cloud-init qemu-guest-agent vim
     ```

   - For Debian series, execute the command:

     ```
     apt install cloud-init cloud-guest-utils qemu-guest-agent vim
     ```

4. Edit the SSHD configuration file.
   1. Execute the following command to edit the sshd_config file.

```
vim /etc/ssh/sshd_config
```

2. Add the following configurations.

```
PermitRootLogin yes  # Allow the root user to log in with a pass
word
PubkeyAuthentication yes  # Allow key-based login
```

3. Save the modified configuration.

5. Execute the following command to delete the default password for the root user.

```
passwd -d root
```

6. Modify the source address file.

1. Execute the following command to modify the system's source address file and change the address to a suitable mirror site address.

```
vim /etc/apt/sources.list.d/ubuntu.sources
```

2. Save the configuration after modifications.

7. Modify the cloud-init configuration to automatically expand the root directory.

1. Execute the following command to edit the cloud.cfg configuration file.

```
vim /etc/cloud/cloud.cfg
```

2. Add the following configuration content.

```
runcmd:
  - [growpart, /dev/vda, 1]  # The growpart command is used to e
xtend the partition on the disk, which will extend the /dev/vda1
partition.
  - [xfs_growfs, /dev/vda1]  # The xfs_growfs command is used to
extend the XFS file system to occupy all available space on the
partition. /dev/vda1 is the partition where the file system to b
e extended is located. After extending the partition, using xfs_
growfs ensures that the file system itself is also expanded to t
he new partition size.
```

3. Save the configuration after modifications.

8. After completing the configuration, shut down the operating system.

6 Export and Use the Custom Linux Image

For specific operations, please refer to Export Virtual Machine Image.

Exporting Virtual Machine Images

This feature is used to export the system image of a virtual machine and upload it to object storage, allowing the files in object storage to be added as sources to the platform's virtual machine images.

TOC

Procedure

Note: All operations below must be performed on the control node of the cluster where the virtual machine resides.

1 Stopping the Virtual Machine

1. Go to **Administrator**.
2. In the left navigation bar, click **Virtualization Management** > **Virtual Machines**.

3. Click on the name of the virtual machine whose system image needs to be exported, which will redirect you to the virtual machine details page in the Container Platform.
4. Click **Stop**.

2 Creating the vmexport Resource

1. Open the CLI tool.
2. Execute the following command to set variables.

```
NAMESPACE=<namespace>
VM_NAME=<vm_name>
TTL_DURATION=2h
```

Parameter explanation:

- **NAMESPACE**: The name of the namespace where the virtual machine resides; replace the *<namespace>* part with this name.
- **VM_NAME**: The name of the virtual machine whose system image needs to be exported; replace the *<vm_name>* part with this name.
- **TTL_DURATION**: The lifetime of the export task, defaulting to 2 hours but can be increased as needed.

3. Execute the following command to create the vmexport resource.

```
cat <<EOF | kubectl create -f -
apiVersion: export.kubevirt.io/v1alpha1
kind: VirtualMachineExport
metadata:
  name: export-$VM_NAME
  namespace: $NAMESPACE
spec:
  ttlDuration: $TTL_DURATION
  source:
    apiGroup: "kubevirt.io"
    kind: VirtualMachine
    name: $VM_NAME
EOF
```

If similar echo information appears, it indicates successful creation.

```
virtualmachineexport.export.kubevirt.io/export-k1 created
```

4. Execute the following command to check the status of the vmexport resource.

```
kubectl -n $NAMESPACE get vmexport export-$VM_NAME -w
```

Echo information:

```
NAME         SOURCEKIND       SOURCENAME   PHASE
export-k1    VirtualMachine   k1           Ready
```

5. When the PHASE field in the echo information changes to Ready, type ctrl (control) + c to stop the watch operation.
6. Execute the following command to get the TOKEN.

```
TOKEN=$(kubectl -n $NAMESPACE get secret export-token-export-$VM_N
AME -o jsonpath={.data.token} | base64 -d)
```

3 Downloading the Virtual Machine Image File

1. Execute the following command to get the IP address of the virtual machine export Pod in the specified namespace and store it in the EXPORT_SERVER_IP environment variable.

```
EXPORT_SERVER_IP=$(kubectl -n $NAMESPACE get po virt-export-export
-$VM_NAME -o jsonpath='{.status.podIP}')
```

2. Execute the following command to set the URL environment variable, which points to the virtual machine's disk image file.

```
URL=https://$EXPORT_SERVER_IP:8443/volumes/$VM_NAME-rootfs/disk.im
g.gz
```

3. Execute the following command to download the image file, with the downloaded file named disk.img.gz.

```
curl -k -O -H "x-kubevirt-export-token: $TOKEN" $URL
```

4 Uploading the Virtual Machine Image File to Object Storage

Upload the downloaded image file to object storage. Any S3 tool can be used for the upload, and this document will introduce the mc (minio-client) tool as an example.

1. Execute the following command to configure the mc tool and connect to the specified S3 storage service.

```
mc alias set minio <ENDPOINT> <ACCESSKEY> <SECRETKEY>
```

   Parameter explanation:

   - ENDPOINT: The address of the S3 storage service; replace the `<ENDPOINT>` part with this address.
   - ACCESSKEY, SECRETKEY: The user ak and sk of the S3 storage service used for authentication; for related information, please refer to MinIO Object Storage ↗.

2. Execute the following command to create a bucket for storing the virtual machine image files.

```
mc mb minio/vmdisks
```

3. Execute the following command to upload the exported virtual machine image file disk.img.gz to the created bucket.

```
mc put disk.img.gz minio/vmdisks
```

5 Creating the Virtual Machine Image

1. Go to **Administrator**.

2. In the left navigation bar, click **Virtualization Management** > **Virtual Machine Images**.
3. Click **Add Virtual Machine Image**.
4. In the image address, fill in <*ENDPOINT*>/vmdisks/disk.img.gz, replacing the <ENDPOINT> part with the address of the S3 storage service. For other parameter explanations, please refer to Adding Virtual Machine Images.
5. Click **Add**.

Permissions

Function	Action	Platform Administrator	Platform auditors	Project Manager
virtualmachineimagetemplates `acp-virtualmachineimagetemplates`	View	✓	✓	✓
	Create	✓	×	×
	Update	✓	×	×
	Delete	✓	×	×

Virtual Machine

Introduction

Introduction

Guides

Creating Virtual Machines/Virtu

Prerequisites

Notes

Create Virtual Machine

Create Virtual Machine Group

Batch Operations on Virtual Ma

Procedure

Logging int

Procedure

Managing Key Pairs

Creating Key Pairs

Updating Key Pairs

Managing V

Reset Passwor

Update Key

ecific

on

Monitoring and Alerts

Monitoring

Alerts

Quick Location of Virtual Machines

Prerequisites

Procedure

How To

Configuring USB host passthro

- Feature Overview
- Use Cases
- Prerequisites
- Steps
- Operation Result
- Learn More

Virtual Machine Hot Migration

- Overview
- Constraints and Limitations
- Prerequisites
- Operation Steps

Virtual Mach

- Steps to Operat

Clone Virtua

- Ensure Prerequ
- Start Quickly

Physical GPU Passthrough Environment Prep

Configuring High Availability for Virtual Machines

- Overview
- Glossary
- Component Overview
- Flow of events during fencing and remediation
- Procedure

Create a VM

- Prerequisites
- Procedure

Troubleshooting

Pod Migration and Recovery fr Nodes

Hot Migration Error Messages and Solutions

Introduction

KubeVirt provides CRDs (Custom Resource Definitions) such as **VirtualMachine** and **VirtualMachineInstance** to abstract virtual machine (VM) resources. Based on these CRDs, users gain comprehensive VM management capabilities. Building on this foundation, **ACP Virtualization With KubeVirt** further enhances usability by offering a **Web Console**, enabling users to perform various operations with greater ease.

Guides

Creating Virtual Machines/Virtu

Prerequisites

Notes

Create Virtual Machine

Create Virtual Machine Group

Batch Operations on Virtual Ma

Procedure

Logging int

Procedure

Managing Key Pairs

Creating Key Pairs

Updating Key Pairs

Managing V

Reset Passwor

Update Key

ecific

on

Monitoring and Alerts

Monitoring

Alerts

Quick Location of Virtual Machines

Prerequisites

Procedure

Creating Virtual Machines/Virtual Machine Groups

Create a virtual machine (VirtualMachineInstance) using an image, and schedule the virtual machine to physical nodes with Kubevirt components installed and virtualization enabled.

You can create a single virtual machine through Create Virtual Machine, or you can quickly create multiple virtual machines (VirtualMachineInstance) with the same configuration by using Create Virtual Machine Group (virtualMachinePool).

TOC

Prerequisites

- Before creating a virtual machine using an image, please confirm the following with the platform administrator:

- The target cluster is a self-built cluster, and the Kubevirt components have been deployed.
- The target node must be a physical node with virtualization enabled.
- A virtual machine image has been added to the platform.

- If you need to use the physical GPU passthrough feature of the virtual machine, please contact the platform administrator for the following configuration:

  1. Obtain GPU passthrough environment preparation plan and prepare the necessary environment.
  2. Prepare the required physical GPU and enable the related features for physical GPU passthrough for the virtual machine.

Notes

When using Windows virtual machines, only logins via the **username/password** set in the virtual machine image are supported. Please contact the platform administrator to obtain this information in advance.

Create Virtual Machine

Procedure

Note: The following content presents an example of creating a virtual machine using a form, and you may also switch to YAML format for the operation.

1. Enter **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machines**.
3. Click **Create Virtual Machine**.
4. In the **Basic Information** area, fill in the name and display name of the virtual machine and set tags or annotations.

Parameter	Description
Tags	Used to select objects and find collections of objects that meet certain criteria. Must be a key-value pair, for example: *app.kubernetes.io/name: hello-app*.
Annotations	Used to provide any information to development and operations teams. Must be a key-value pair, for example: *cpaas.io/maintainer: kim*.

5. Set the machine type and choose a virtual machine image.

Parameter	Description
Specifications	You can select recommended usage scenarios or custom resource limits based on your needs.
Physical GPU (Alpha)	Select the model of the physical GPU; only one physical GPU can be allocated to each virtual machine. **Note**: Physical GPU passthrough for the virtual machine refers to the direct allocation of the actual Graphics Processing Unit (GPU) to the virtual machine in a virtualization environment, enabling it to directly access and utilize the physical GPU to achieve graphical performance equivalent to running directly on a physical machine, avoiding performance bottlenecks caused by virtual graphics adapters and enhancing overall performance.
Image	Choose a public image that has been assigned to the platform project by the platform administrator. **Note**: Only supports selecting images with the same **CPU architecture** as the cluster architecture.

6. In the **Storage** area, refer to the following instructions to configure the relevant information.

Parameter	Description
Disk Name	The name of the storage disk; the system disk name cannot be modified.

Parameter	Description
Type	• **Root Disk**: The system automatically creates a VirtIO type rootfs system disk to store the operating system and data. • **Data Disk**: Click to add multiple data disks for persistent data storage. Defaults to VirtIO device. **Note**: Data disk names must not duplicate existing disk names.
Volume Mode	• **File System**: Mount the disk as a mounted file directory. • **Block Device**: Mount the disk as a block device.
Storage Class	The platform maintains virtual machine disks by creating and managing persistent volume claims. You need to specify a storage class required for dynamically creating persistent volume claims. Different storage classes support different volume modes; if there is no available storage class for the selected volume mode, please contact an administrator for addition.
Capacity	The capacity required for the virtual machine storage; the minimum for the system disk is 20 G.
Delete with VM	Defaults to enabled, cannot be modified, indicating that the disk data will also be deleted when the virtual machine is deleted.

7. In the **Network** area, refer to the following instructions to configure the relevant information.

Parameter	Description
IP Address	• Defaults to **Dynamic (DHCP)**; an IP is dynamically assigned when the virtual machine starts, and the IP is released when the virtual machine stops. • If binding a **Static IP**, the virtual machine will always use this IP address even after a restart. If there are no available IPs in the current project, please release an IP appropriately first.
Network Mode	• **Bridged**: The virtual machine shares the same IP address as the container group and communicates externally through this IP address. • **NAT**: The virtual machine will be assigned an internal IP address but will translate to the container group IP address for external communication. Open ports indicate the exposed ports of the virtual machine, such as the SSH service port 22; not filling in **Open Ports** indicates that all ports are open.
Auxiliary Network Card	Add auxiliary network cards as needed. **Note**: • If auxiliary network card features are required or there are no available types of auxiliary network card networks, please contact the platform administrator for configuration. • SR-IOV types only support Linux operating systems on x86_64 architecture. • Defaults to obtaining IP addresses via DHCP. • After multiple reboots, SR-IOV virtual machines may exhibit two different VFs but with the same MAC address.

8. In the **Initialization Settings** area, refer to the following instructions to configure the relevant information.

Parameter	Description
Keys	Always use SSH keys for remote login verification. This method does not require password validation; it is recommended to log in to the virtual machine using keys. • You can use the keys already available in the platform or create new keys immediately; all relevant keys can be viewed on the **Virtualization > Key Pairs** page. • Only individuals with the private key can access the virtual machine via SSH. If multiple people are to maintain the virtual machine together, multiple keys can be associated, and private keys can be assigned to different users. If key leakage occurs, the associated key can be promptly revoked to reduce damage. • The public key of the SSH key is stored in the platform in a confidential form; the platform does not store your private key, so please keep it safe by yourself. • Please consult the relevant operating system documentation for the root user password.
Password	Use the operating system user and password for login verification, which can still be updated to the key method later. • The user is only an initial account; after the virtual machine is successfully created, you can also create other operating system users in the virtual machine for login. • The platform encrypts and stores your root user password, and you will not see its plaintext password again, so please keep it safe by yourself.
Start Immediately	Defaults to enabled. Enabling this option will start the virtual machine immediately after creation, otherwise only the virtual machine will be created.

9. (Optional) In the **Advanced Configuration** area, refer to the following instructions to configure the relevant information.

Parameter	Description
Health Check	• **Liveness Check**: Checks if the virtual machine is in a healthy state; if the detection result is abnormal, it will determine whether to restart the instance based on the health check configuration. • **Availability Check**: Checks if the virtual machine has completed startup and is in a normal service state; if the health status of the virtual machine instance is detected as abnormal, the state of the virtual machine will be updated. For related parameter descriptions.
Node Affinity	• **Preferred**: The virtual machine will be scheduled to nodes that meet affinity requirements whenever possible. The system will determine nodes capable of running the virtual machine by combining affinity weights and other scheduling requirements (e.g., compute resource requirements). • **Required**: The virtual machine will only be scheduled to nodes that fully meet affinity requirements.

10. After confirming that the information is correct, click **Create**.
    Wait for the virtual machine to change from **Creating** to **Running** status.

Related Operations

You can click the ⋮ icon on the right side of the list page or the **Actions** in the upper right corner of the details page to update or delete the virtual machine as needed. For other related operations like resetting passwords or updating keys, please refer to Manage Virtual Machines.

Note:

- Updates can only be performed when the virtual machine is in **Abnormal**, **Unknown**, or **Stopped** status.

- Updates do not support displaying disks that were separately attached or created after the virtual machine was created.
- By default, **Start Immediately** is disabled during updates; you can enable it as needed.

Create Virtual Machine Group

Procedure

Note: The following content presents an example of creating a virtual machine group using a form, and you may also switch to YAML format for the operation.

1. Enter **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machine Groups**.
3. Click **Create Virtual Machine Group**.
4. In the **Basic Information** area, refer to the following instructions to configure the information for the virtual machine group.

Parameter	Description
Number of Instances	The number of virtual machines created by the virtual machine group.
Anti-Affinity between Instances	If enabled, when scheduling multiple virtual machines to nodes, it will try to distribute the virtual machines across different nodes, which can enhance the high availability of a group of virtual machines.
Tags	Tags can be added to the virtual machine group. Tags can be used to select objects and find collections of objects that meet certain criteria. Must be a key-value pair, for example: *app.kubernetes.io/name: hello-app*.

5. In the **Virtual Machine Template** area, refer to Create Virtual Machine to configure unified tags, annotations, specifications, images, storage, and other information for all the virtual machines in the group.

6. After confirming that the information is correct, click **Create**.

   Tip: After successful creation, you can go to the **Virtual Machines** list page to view the information of the virtual machines created via the virtual machine group.

Batch Operations on Virtual Machines

Perform batch operations such as starting, stopping, restarting, and deleting virtual machines.

TOC

Procedure

1. Access the **Container Platform**.
2. Click on **Virtualization** > **Virtual Machines** in the left navigation bar.
3. Locate the target virtual machine, click ⋮ to perform operations on a single virtual machine, or refer to the image below for batch operations on virtual machines.
   Note:
   - The **Start/Batch Start** operation can be executed when the virtual machine is in a suspended or stopped state; the **Stop/Batch Stop** operation can be executed when the virtual machine is in a Preparing, Starting, Running, Suspended, Unknown, or Exception state; the **Restart/Batch Restart** operation can be executed when the virtual machine is in a Running state.
   - Performing a forced **Restart/Stop** operation on a virtual machine is equivalent to cutting off power to the virtual machine, which may result in loss of data that has not been written to disk.

4. Complete the operations according to the prompts on the interface. When the virtual machine changes to the states below, the operation is successful.

Operation	Status
Start Virtual Machine	Running
Stop Virtual Machine	Stopped
Restart Virtual Machine	Running

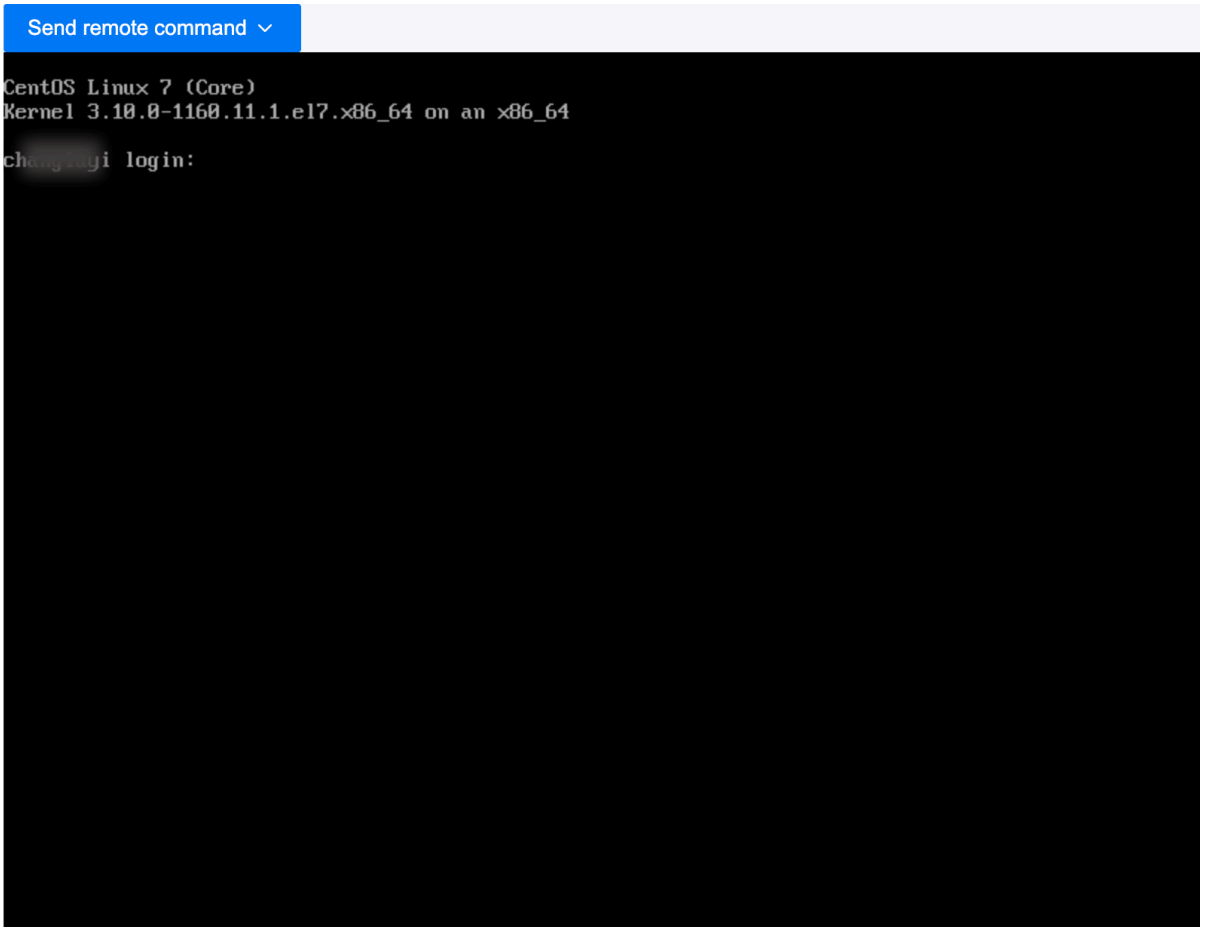
Logging into the Virtual Machine using VNC

Log into the virtual machine using the Web Console (VNC) as an emergency operation method.

TOC

Procedure

1. Access the **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machines**.
3. Click ⋮ > **VNC Login**.
4. The console window will open automatically; you will need to enter your username and password to log in.



Note:

- Supports sending common keyboard commands.
- Supports copying and pasting commands and parameters.

Managing Key Pairs

Create, update or delete key pairs.

TOC

Creating Key Pairs

1. Navigate to **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Key Pairs**.
3. Click **Create Key Pair**.
   Currently, only SSH type key pairs are supported. You can manually import keys or let the system automatically generate a key pair. When using the system-generated key pair, the platform supports automatically downloading the private key to your local machine. The platform will not save the private key.
4. Click **Create**.

Updating Key Pairs

1. Navigate to **Container Platform**.

2. In the left navigation bar, click **Virtualization** > **Key Pairs**.
3. Locate the ***Key Pair Name***, click ⋮ > **Update**.
4. After re-importing or having the system generate a new key pair, click **Update**.

Deleting Key Pairs

1. Navigate to **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Key Pairs**.
3. Locate the ***Key Pair Name***, click ⋮ > **Delete**, and confirm.

Managing Virtual Machines

TOC

Reset Password

Reset the **root** user password. This password also serves as the login password for the virtual machine when logging in using a password.

Procedure

1. Access the **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machines**.
3. Locate the virtual machine and select ⋮ > **Reset Password**.
4. Set the password.
5. Click **Reset**.
   Note: Please keep your password safe. To ensure environment security, the platform encrypts and stores your password, and you will not be able to see the plaintext password again.

Update Key

Update the SSH keys.

Procedure

1. Access the **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machines**.
3. Locate the virtual machine and select ⋮ > **Update Key**.
4. Select one or more associated keys, or **Create Key**.
5. Choose whether to restart immediately; updating keys requires a restart of the virtual machine to take effect.
6. Click **Update**.

Update Specifications

1. Access the **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machines**.
3. Locate the target virtual machine and click ⋮ > **Update Specifications**.

4. Modify the relevant resources based on the platform's recommended scenarios or custom needs.
5. Choose whether to **Restart Immediately**; the configuration will take effect after restarting.
6. Click **Update**.

Live Migration

Note: If you need documentation regarding live migration operations, please contact the administrator for assistance.

1. Access the **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machines**.
3. Locate the target virtual machine and click ⋮ > **Live Migration**.
4. Click **Confirm**.

Update NAT Network Configuration

When using NAT network mode, the platform by default opens port 22 for SSH services, and you can open other ports as needed.

Procedure

1. Access the **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machines**.
3. Click ***Virtual Machine Name***.
4. In the **Basic Information** section, click the icon to the right of **Open Port**.
5. Enter the port number and press the Enter key to confirm.
6. Choose whether to **Restart Immediately**; the configuration will take effect after restarting.
7. Click **Update**.

Update Tags and Annotations

1. Access the **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machines**.
3. Click ***Virtual Machine Name***.
4. In the **Basic Information** section, click the icon to the right of **Tags** or **Annotations**.
5. Configure as needed and click **Update**.

Add Service

1. Access the **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machines**.
3. Click ***Virtual Machine Name***.
4. In the **Login Information** section, click the icon to the right of Internal Route.
5. Refer to the Create Service page for quick addition of internal routes for the virtual machine.
6. Click **Confirm**.

Reinstall Operating System

It is strongly recommended to back up data before reinstalling the operating system to prevent data loss.

Note: This operation will clear all data in the virtual machine's **system disk**, as well as all **snapshots**, and is irreversible. Please proceed with caution!

Procedure

1. Access the **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machines**.
3. Locate the virtual machine and select ⋮ > **Reinstall Operating System**.

4. In the **Reinstall Operating System** window, configure the following parameters.
    - **Provisioning Method**: Currently supports public images.
    - **Select Image**: By default, the current operating system image will be used for reinstallation. If you wish to reinstall a new operating system, first select the operating system of the virtual machine image, then choose the virtual machine image that belongs to that operating system.
5. Click **Reinstall**.

Configure IP

Assign an IP to the virtual machine using dynamic allocation (DHCP) or bind a fixed IP to the virtual machine; the new IP will take effect after the virtual machine is restarted.

Procedure

1. Access the **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machines**.
3. Locate the target virtual machine and click ⋮ > **Configure IP**.
4. Configure the **IP Address**.
    - Fill in an available IP: binding a fixed IP means that even after restarting, the virtual machine will consistently use this IP address.
    - Leave the option blank: this will use dynamic allocation (DHCP) to acquire an IP, assigning the IP when the virtual machine starts and releasing it when the virtual machine stops.
5. Choose whether to **Restart Immediately**; the configuration will take effect after restarting.
6. Click **Configure**.

Monitoring and Alerts

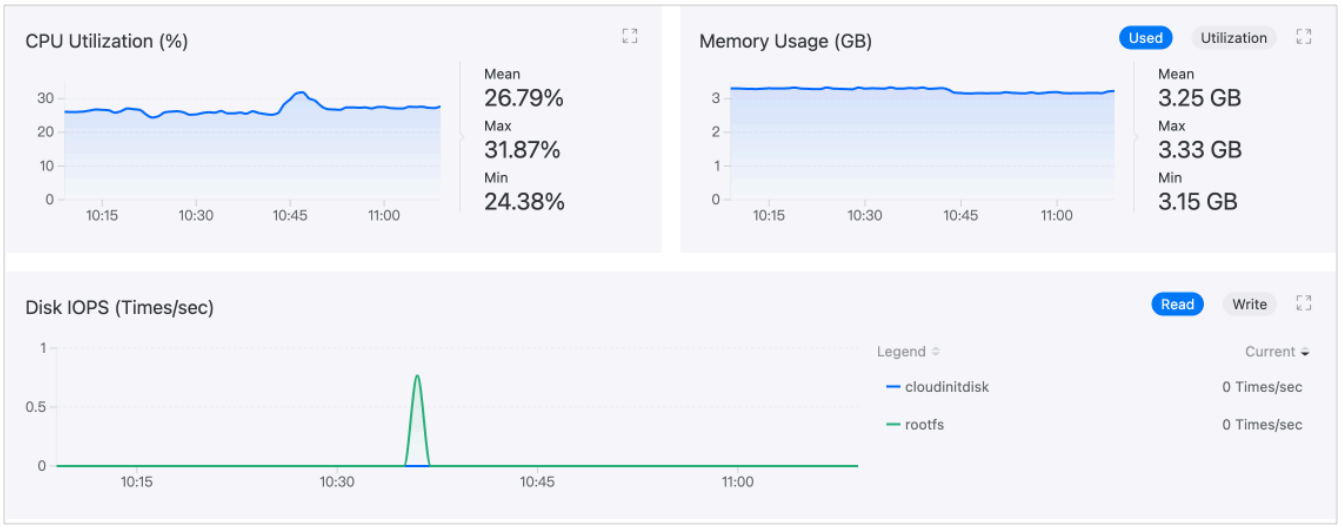
Monitor and alert on virtual machines in terms of CPU, memory, storage, and network. To facilitate timely alerts, notification policies can also be configured.

The intuitively presented monitoring data can be used to provide decision-making support for operations inspection or performance tuning, while the comprehensive alerting and notification mechanism will help ensure the stable operation of virtual machines.

TOC

Monitoring

By default, the platform collects commonly used performance monitoring metrics for virtual machines, including CPU, memory, storage, and network. Navigate to **Virtualization** > **Virtual Machines**, and on the **Monitoring** tab in the virtual machine details, you can view real-time monitoring data for the metrics.



Alerts

Configuring Alert Policies

To enable alerts, you must first create an alert policy. An alert policy describes the objects you wish to monitor, the conditions under which you wish to be alerted, and how you will be notified of relevant alerts. Navigate to **Container Platform** > **Virtualization** > **Virtual Machines**, and in the virtual machine details, click **Create Alert Policy** on the **Alerts** tab to complete the configuration.

Parameter	Description
Alert Type	- Metric Alert: The monitored object is a platform predefined metric, such as *Memory Usage Rate*. - Event Alert: The monitored object is the cause of an event, that is, the reason the virtual machine transitioned to its current state, e.g., BackOff, Pulling, Failed.
Trigger Condition	Composed of comparison operators, alert thresholds, and duration. By comparing the real-time monitoring results with the set thresholds, it determines whether to alert. If a duration is set, the platform will also compare the duration for which the monitored object has been in the alert state.
Alert Level	- Hint: The monitored object has expected issues that do not immediately affect business operations but pose potential risks. For example, if CPU usage exceeds 70% for 3 minutes.

Parameter	Description
	- Warning: The monitored object has operational risks that may affect normal business operations if not addressed promptly. For example, if CPU usage exceeds 80% for 3 minutes. - Serious: The monitored object has known issues that may lead to platform functionality failures, affecting normal business operations. - Disaster: The monitored object has failed, resulting in platform service interruptions, data loss, with significant impact.

Tip: The virtual machine alerting function is similar to the platform's general alerting function. For more detailed configuration guidance, please refer to the general Alerts documentation.

Handling Alerts

Navigate to the **Alerts** tab, and if there are alert status strategies indicated, please address them promptly.

Binding Notification Policies

In addition to real-time alerts on the **Alerts** tab, the platform also supports sending alert information via email, SMS, and other means to relevant personnel, notifying them to take necessary measures to resolve issues or prevent failures. The notification policy needs to be set up by contacting the administrator.

Quick Location of Virtual Machines

The platform supports displaying the list of virtual machines by cluster, allowing platform administrators to quickly locate the namespace of the virtual machine and perform operations such as scaling up or troubleshooting, thereby improving operational efficiency.

TOC

Prerequisites

Ensure that the virtualization feature is enabled for the current cluster before use. Please refer to Install.

Procedure

1. Go to **Administrator**.
2. In the left navigation bar, click **Virtualization Management** > **Virtual Machines**.
3. Select ***Cluster*** to view the list of virtual machines in that cluster.
4. You can quickly locate the virtual machine by its name, IP address, or creator.
5. Click on the virtual machine ***Name*** link to enter the details page of that virtual machine, where you can perform operations such as scaling up or troubleshooting.

How To

Configuring USB host passthro

- Feature Overview
- Use Cases
- Prerequisites
- Steps
- Operation Result
- Learn More

Virtual Machine Hot Migration

- Overview
- Constraints and Limitations
- Prerequisites
- Operation Steps

Virtual Mach

- Steps to Operat

Clone Virtua

- Ensure Prerequ
- Start Quickly

Physical GPU Passthrough Environment Prep

Configuring High Availability for Virtual Machines

- Overview
- Glossary
- Component Overview
- Flow of events during fencing and remediation
- Procedure

Create a VM

- Prerequisites
- Procedure

Configuring USB host passthrough

TOC

Feature Overview

USB(Universal Serial Bus) pass-through feature enables you to access and manage USB devices from a virtual machine.

Use Cases

Some applications running in virtual machines (VMs) have encryption requirements and need to interact with dedicated USB devices. In such cases, it is necessary to passthrough the USB

devices from the host machine to the VM.

Prerequisites

- The platform version must be at least v3.18.

Steps

1 Expose USB devices

To assign a USB device to a VM, the USB device must be exposed via a **ResourceName**. This can be configured by editing the `spec.permittedHostDevices.usbHostDevices` section in the **HyperConverged CR** under the kubevirt namespace.

Below is an example configuration for a USB device with **ResourceName** kubevirt.io/storage, where the vendor is `0bda` and the product is `8812`:

```yaml
spec:
  permittedHostDevices:
    usbHostDevices:
      - resourceName: kubevirt.io/storage
        selectors:
          - vendor: '0bda'
            product: '8812'
```

Tip

The vendor and product identifiers of a USB device can be obtained using the `lsusb` command. For example:

```
lsusb
Bus 001 Device 007: ID 0bda:8812 Realtek Semiconductor Corp. RT
L8812AU 802.11a/b/g/n/ac 2T2R DB WLAN Adapter
```

This command lists all connected USB devices, where ID displays the vendor:product pair.

2 Assign USB devices to a Virtual Machine

Now, in the VM configuration, you can add `spec.domain.devices.hostDevices.deviceName` to reference the `ResourceName` provided in the previous step and assign it a local name. For example:

```
spec:
  domain:
    devices:
      hostDevices:
        - deviceName: kubevirt.io/storage
          name: usb-storage
```

Tip

Ensure the VM is stopped before editing the configuration.

Operation Result

After completing the configuration, execute the `lsusb` command within the virtual machine. If the output lists the host node's USB device, the passthrough was successful. For example:

```
lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 002: ID 0bda:8812 Realtek Semiconductor Corp. RTL8812AU 80
2.11a/b/g/n/ac 2T2R DB WLAN Adapter
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Learn More

You may want to passthrough multiple USB devices to a virtual machine, such as a keyboard, mouse, or smart card device. We support assigning multiple USB devices under the same resourceName. Here's how to configure it:

1 Expose multiple USB devices

```yaml
spec:
  permittedHostDevices:
    usbHostDevices:
      - resourceName: kubevirt.io/peripherals
        selectors:
          - vendor: '0bda'
            product: '8812'
          - vendor: '062a'
            product: '4102'
          - vendor: '072f'
            product: 'b100'
```

> **Tip**
>
> Note: All USB devices must be physically connected and detected on the host to ensure successful assignment to the virtual machine.

2 Assign USB devices to a Virtual Machine

```yaml
spec:
  domain:
    devices:
      hostDevices:
        - deviceName: kubevirt.io/peripherals
          name: local-peripherals
```

Virtual Machine Hot Migration

TOC

Overview

The virtual machine hot migration technology allows for moving a virtual machine from one physical server to another without shutting down or interrupting the virtual machine. The platform's virtual machine solution is implemented based on the open-source component KubeVirt, which uses a mode called ProCopy for hot migration by default.

ProCopy

ProCopy (Pre-Copy Memory Migration) is a commonly used virtual machine migration technology that ensures service continuity during migration by pre-copying the virtual

machine's memory data. The specific process is as follows:

1. **Initial Phase**: At the start of the migration, the source host will copy the virtual machine's memory pages to the target host while the virtual machine continues to run. Because the virtual machine continues running, some memory pages may be modified during the copying process.
2. **Iterative Copying**: The source host repeatedly copies the modified memory pages to the target host until the number of modified pages decreases to an acceptable level. Each round of copying is called an iteration, and the number of unmodified memory pages gradually decreases after each iteration.
3. **Stop and Copy**: When the remaining un-copied memory pages are sufficiently few, the virtual machine will pause briefly (usually only a few seconds to a dozen seconds), during which the last memory pages are copied to the target host, and the virtual machine's CPU and device states are synchronized to the target host.
4. **Resume Operation**: The virtual machine resumes operation on the target host.

Constraints and Limitations

It is recommended that the two physical machines involved in the hot migration operation use the same hardware configuration. If the configurations are inconsistent (for example, different CPU models), migration may fail.

Prerequisites

Please enable the relevant virtual machine hot migration functions in advance.

Operation Steps

Deploy kubevirt-operator

Note: For detailed steps and parameter explanations, please refer to Deploy Operator.

1. Go to **Administrator**.

2. In the left navigation bar, click **App Store Management** > **Operators**.
3. Click **Cluster** at the top of the page to switch to the cluster where the Operator needs to be deployed.
4. In the OperatorHub tab, click **Deploy** on the **KubeVirt HyperConverged Cluster Operator** card.
5. Configure the parameters as needed and click **Deploy**. You can check the Operator deployment status in the **Deployed** tab.

Create HyperConverged Instance

For specific creation steps, please refer to Create HyperConverged Instance.

Prepare the Virtual Machine

Note: It is recommended to use the Kube-OVN Underlay network. For related configurations, please refer to Create Subnet (Kube-OVN Underlay Network).

1. Go to **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machine**.
3. Click **Create Virtual Machine**.
4. Click **More** in the **Basic Information** area to expand more configuration options, and click **Add** corresponding to **Annotations**, adding annotations according to the key-value pairs below. If the network plugin is Kube-OVN, there is no need to manually fill in this annotation.
   Note: Due to form restrictions, please enter the **value** of the annotation first before entering the **key** of the annotation.

	Annotation
Value	true
Key	kubevirt.io/allow-pod-bridge-network-live-migration

5. Configure other virtual machine parameters as needed. For specific parameter descriptions, please refer to the relevant product documentation.

Parameter	Description
Volume Mode	Must use **Block Mode**.
Storage Class	Must use **CephRBD block storage type storage class**.
Network Mode	Recommended to use **Bridge**.

6. Click **Create**.

Start Hot Migration

Note: Hot migration can only be started when the virtual machine status is **Running**.

1. Go to **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machine**.
3. Start the hot migration. There are two ways to do this:
   - Click ⋮ > **Hot Migration** on the right side of the virtual machine that needs to be migrated in the list.
   - Click the name of the virtual machine that needs to be migrated in the list to enter the detail information page, then click **Actions** > **Hot Migration**.
4. Click **Confirm**. You can check the migration progress through **Virtual Machine Status** or **Real-Time Events**. When the status changes from **Migrating** to **Running**, or when a real-time event appears with information like **Migrated: The VirtualMachineInstance migrated to node 10.1.1.1.**, it indicates that the migration was successful.

Virtual Machine Recovery

In certain scenarios, such as incorrect modifications to fstab or filesystem errors requiring fsck, virtual machines may fail to start properly. In such cases, you can utilize rescue mode to repair the root filesystem (rootfs) or retrieve data from the system.

TOC

Steps to Operate

Obtain Image Address

1. In the left navigation bar, click **Virtualization Management** > **Virtual Machine Images**.
2. Select the platform-provided **Source** as **Image Repository**, and the **Operating System** as either **CentOS or Ubuntu**. Click ⋮ > **Update** on the right.
3. Copy and save the **Image Address**. This document uses `192.168.1.1:11443/3rdparty/vmdisks/centos:7.9` as an example.
4. Click **Cancel**.

Modify Virtual Machine YAML File

1. Access the **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machines**.
3. Click ⋮ > **Stop** on the right side of the virtual machine that needs repair to **Stop** or **Force Stop** it.
4. Click ⋮ > **Update** on the right side of the virtual machine.
5. Switch to **YAML** and modify the following fields.

   - Add the following content under `spec.template.spec.domain.devices.disks`. Adding a bootOrder parameter can control which disk is prioritized during the virtual machine's boot process; a lower bootOrder value indicates higher priority.
     Note: If the original `spec.template.spec.domain.devices.disks` field contains `bootOrder: 1`, increase the original value to ensure that the newly added bootOrder value is lower than the original.

     ```
     disks:
       - bootOrder: 1
         disk:
           bus: virtio
         name: containerdisk
     ```

     Modified YAML example:

```
domain:
  devices:
    disks:
      - bootOrder: 1  # Added Field
        disk:
          bus: virtio
        name: containerdisk
      - disk:
          bus: virtio
        name: cloudinitdisk
      - disk:      # Increase the original bootOrder: 1 value
          bus: virtio
        name: rootfs
        bootOrder: 10
      - disk:
          bus: virtio
        name: "1"
```

- Add the following content under `spec.template.spec.volumes`.

  Note: Please replace the image address in the following `image` field with the one obtained from Obtain Image Address.

  ```
  - containerDisk:
      image: 192.168.1.1:11443/3rdparty/vmdisks/centos:7.9
    name: containerdisk
  ```

  Modified YAML example:

  ```
  volumes:
    - containerDisk:  # Added Field
        image: 192.168.1.1:11443/3rdparty/vmdisks/centos:7.9
      name: containerdisk
    - dataVolume:
        name: k2-rootfs
      name: rootfs
    - dataVolume:
        name: k2-1
      name: "1"
  ```

6. Click **Update**.

Note: After modifying the YAML file, do not switch to **Form**, just click **Update** directly.

7. Click ⋮ > **Start** on the right side of the virtual machine.

Mount the Original rootfs and Perform Repair

1. Log in to the virtual machine using the original password or key and enter the command `df -h /` to find that the rootfs filesystem has been replaced. You can use mount-related commands to mount it or fsck-related commands to check and repair the original filesystem.
2. After completion, shut down the virtual machine.

Restore the Virtual Machine YAML File

Follow the steps in Modify Virtual Machine YAML File to restore the virtual machine YAML file to its original state. At this point, the virtual machine can start normally.

Clone Virtual Machines on KubeVirt

This document provides step-by-step guidance on cloning virtual machines (VMs) using KubeVirt's `VirtualMachineClone` API.

TOC

Ensure Prerequisites

Before initiating a VM clone operation, make sure the following requirements are satisfied:

- **Snapshot-Capable Storage**: The Clone API relies on Snapshot & Restore functionalities. The virtual machine's storage class must support **volume snapshots**, and snapshot functionality must be explicitly enabled for that storage backend.

Start Quickly

Follow these quick steps to clone a VM:

1. **Prepare the Clone Manifest**:

   Create a file named `clone.yaml` with the following structure:

   ```yaml
   apiVersion: clone.kubevirt.io/v1beta1
   kind: VirtualMachineClone
   metadata:
     name: example-vm-clone
     namespace: ns-where-vm-run
   spec:
     source:
       apiGroup: kubevirt.io
       kind: VirtualMachine
       name: source-vm
     target:
       apiGroup: kubevirt.io
       kind: VirtualMachine
       name: target-vm
   ```

2. **Execute the Clone Operation**:

   Apply the manifest:

   ```bash
   kubectl create -f clone.yaml
   ```

3. **Monitor the Clone Status**:

   Wait until the cloning is completed successfully:

   ```bash
   kubectl wait vmclone example-vm-clone --for condition=Ready
   ```

4. **Verify the Cloned VM**:

   Inspect the cloned VM configuration:

   ```bash
   kubectl get vm target-vm -o yaml
   ```

5. **Fix the DataVolume Label (UI metadata)**:

   The platform UI links VMs to their disks through the label `vm.cpaas.io/used-by=<vm-name>` that is automatically added to every DataVolume. After a clone operation the new

DataVolume inherits the label from the *source* VM, so the UI still thinks it belongs to the old VM. Update the label on the newly created DV so the relationship displays correctly (functionality is **not** affected).

```
# List DataVolumes in the VM's namespace; the cloned DV name usually st
arts with "restore-"
kubectl get datavolumes -n <ns-where-vm-run>

# Overwrite the label to point to the cloned VM
kubectl label datavolume <new-dv-name> -n <ns-where-vm-run> vm.cpaas.i
o/used-by=<target-vm> --overwrite
```

Understand the VirtualMachineClone Object

View a Complete VirtualMachineClone Example

Here's a complete example of a `VirtualMachineClone` resource with detailed inline comments:

```yaml
apiVersion: clone.kubevirt.io/v1beta1
kind: VirtualMachineClone
metadata:
  name: detailed-vm-clone
  namespace: ns-where-vm-run
spec:
  # Source VM details
  source:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: vm-source

  # Target VM details
  target:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: vm-target

  # Filters for labels and annotations copied from source
  labelFilters:
    - "*"
    - "!exclude-key/*"
  annotationFilters:
    - "include-annotations/*"

  # Template filters to manage network annotations
  template:
    labelFilters:
      - "*"
    annotationFilters:
      - "!network-info/*"

  # Explicitly set new MAC addresses
  newMacAddresses:
    eth0: "02-00-00-aa-bb-cc"

  # Explicitly set SMBios serial
  newSMBiosSerial: "unique-serial-1234"

  # JSON patches to further customize the cloned VM
  patches:
    - '{"op": "add", "path": "/metadata/labels/new-label", "value": "new-value"}'
```

```
    - '{"op": "replace", "path": "/spec/template/metadata/annotations/new
-annotation", "value": "updated-value"}'
```

Understand Each Field

- **Source and Target**:
  - Define the original VM (`source`) and the cloned VM (`target`).
  - Auto-generated if the `target` name is omitted.
  - Both VMs must reside within the same namespace.
- **Label and Annotation Filters**:
  - Control copying or excluding labels/annotations from the source VM using wildcards (`*`) and negations (`!`).
- **Template Label and Annotation Filters**:
  - Useful for managing network-related annotations, especially with CNIs like Kube-OVN.
- **newMacAddresses**:
  - Optionally specify new MAC addresses for network interfaces.
  - Automatically regenerated if omitted.
- **newSMBiosSerial**:
  - Optionally specify a new SMBios serial.
  - Auto-generated based on VM name if not provided.
- **JSON Patches**:
  - Advanced customizations directly applied to VM specs.

Check Clone Operation Phases

The `.status.phase` of a `VirtualMachineClone` object changes according to the cloning process progress. The table below explains each phase:

Phase	Explanation
SnapshotInProgress	Creating a snapshot of the source VM, initial step when cloning a running VM.
CreatingTargetVM	Snapshot is complete; creating metadata and specification for the target VM.
RestoreInProgress	DataVolume and PersistentVolumeClaim creation in progress, restoring data from snapshot.
Succeeded	Operation successfully completed. Target VM and storage are ready.
Failed	Operation failed. Check `events` and `status.conditions` for detailed error information.
Unknown	Unable to determine the clone operation status, potentially indicating a controller issue.

Physical GPU Passthrough Environment Preparation

Physical GPU passthrough in virtual machines refers to the process of directly allocating the actual Graphics Processing Unit (GPU) to a virtual machine within a virtualization environment. This allows the virtual machine to access and utilize the physical GPU directly, achieving graphics performance equivalent to that of running directly on a physical machine. It avoids performance bottlenecks caused by virtual graphics adapters, thus enhancing overall performance.

TOC

Constraints and Limitations

The physical GPU passthrough functionality requires the use of the kubevirt-gpu-device-plugin; however, there is currently no ARM64 image available for the kubevirt-gpu-device-plugin, which means this functionality cannot be used in an operating system with an ARM64 CPU architecture.

Prerequisites

Chart and Image Preparation

Obtain the following Chart and images and upload them to an image repository. This document uses `build-harbor.example.cn` as an example repository address. For the specific method of obtaining the Chart and images, please contact the relevant personnel.

Chart

- build-harbor.example.cn/example/chart-gpu-operator:v23 .9.1

Images

- build-harbor.example.cn/3rdparty/nvidia/gpu-operator:v23 .9.0
- build-harbor.example.cn/3rdparty/nvidia/cloud-native/gpu-operator-validator:v23 .9.0
- build-harbor.example.cn/3rdparty/nvidia/cuda:12 .3.1-base-ubi8
- build-harbor.example.cn/3rdparty/nvidia/kubevirt-gpu-device-plugin:v1 .2.4
- build-harbor.example.cn/3rdparty/nvidia/nfd/node-feature-discovery:v0 .14.2

Enabling IOMMU

The procedure for enabling IOMMU varies across different operating systems. Please refer to the documentation of the corresponding operating system. This document uses CentOS as an example, and all commands should be executed in the terminal.

1. Edit the `/etc/default/grub` file and add `intel_iommu=on iommu=pt` to the `GRUB_CMDLINE_LINUX` configuration option.

```
GRUB_CMDLINE_LINUX="crashkernel=auto rd.lvm.lv=centos/root rhgb quiet i
ntel_iommu=on iommu=pt"
```

2. Execute the following command to generate the `grub.cfg` file.

```
grub2-mkconfig -o /boot/grub2/grub.cfg
```

3. Restart the server.
4. Run the following command to confirm if IOMMU has been successfully enabled. If the output contains `IOMMU enabled`, then it indicates that it has been successfully enabled.

```
dmesg | grep -i iommu
```

Operating Steps

Note: All commands below should be executed in the CLI tool on the corresponding cluster Master node unless otherwise specified.

Create Namespace

Execute the following command to create a namespace named `gpu-system`. If the output displays `namespace/gpu-system created`, it indicates that the creation was successful.

```
kubectl create ns gpu-system
```

Deploy gpu-operator

1. Execute the following command to deploy the gpu-operator.

```
export REGISTRY=<registry> # Replace <registry> with the repository add
ress where the gpu-operator image is located, e.g.: export REGISTRY=bui
ld-harbor.example.cn

cat <<EOF | kubectl create -f -
apiVersion: operator.alauda.io/v1alpha1
kind: AppRelease
metadata:
  annotations:
    auto-recycle: "true"
    interval-sync: "true"
  name: gpu-operator
  namespace: gpu-system
spec:
  destination:
    cluster: ""
    namespace: "gpu-operator"
  source:
    charts:
    - name: <chartName> # Replace <chartName> with the actual chart pat
h, e.g.: name = example/chart-gpu-operator
      releaseName: gpu-operator
      targetRevision: v23.9.1
    repoURL: $REGISTRY
  timeout: 120
  values:
    global:
      registry:
        address: $REGISTRY
    nfd:
      enabled: true
    sandboxWorkloads:
      enabled: true
      defaultWorkload: "vm-passthrough"
EOF
```

2. Execute the following command to check if the gpu-operator has synchronized. If `SYNC` shows as `Synced`, it indicates that it has synchronized successfully.

```
kubectl -n gpu-system get apprelease gpu-operator
```

Output information:

```
NAME           SYNC           HEALTH         MESSAGE          UPDATE   AGE
gpu-operator   Synced         Ready          chart synced     28s      32s
```

3. Execute the following command to retrieve the names of all nodes and find the GPU node name.

```
kubectl get nodes -o wide
```

4. Execute the following command to check if the GPU node has any pass-through capable GPU. If the output contains GPU information similar to `nvidia.com/GK210GL_TESLA_K80`, it indicates that there are pass-through capable GPUs.

```
kubectl get node <gpu-node-name> -o jsonpath='{.status.allocatable}' #
Replace <gpu-node-name> with the GPU node name obtained from Step 3
```

Output information:

```
{"cpu":"39","devices.kubevirt.io/kvm":"1k","devices.kubevirt.io/tun":"1
k","devices.kubevirt.io/vhost-net":"1k","ephemeral-storage":"4265627841
65","hugepages-1Gi":"0","hugepages-2Mi":"0","memory":"122915848Ki","nvi
dia.com/GK210GL_TESLA_K80":"8","pods":"256"}
```

5. At this point, the gpu-operator has been successfully deployed.

Configure Kubevirt

1. Execute the following command to enable the DisableMDEVConfiguration feature. If a message similar to `hyperconverged.hco.kubevirt.io/kubevirt-hyperconverged patched` is returned, it indicates successful enabling.

```
kubectl patch hco kubevirt-hyperconverged -n kubevirt --type='json' -p
='[{"op": "add", "path": "/spec/featureGates/disableMDevConfiguration",
"value": true }]'
```

2. In the terminal of the GPU node, execute the following command to obtain the pciDeviceSelector. The `10de:102d` part in the output is the value of pciDeviceSelector. {#pciDeviceSelector}

```
lspci -nn | grep -i nvidia
```

Output information:

```
04:00.0 3D controller [0302]: NVIDIA Corporation GK210GL [Tesla K80] [1
0de:102d] (rev a1)
05:00.0 3D controller [0302]: NVIDIA Corporation GK210GL [Tesla K80] [1
0de:102d] (rev a1)
08:00.0 3D controller [0302]: NVIDIA Corporation GK210GL [Tesla K80] [1
0de:102d] (rev a1)
09:00.0 3D controller [0302]: NVIDIA Corporation GK210GL [Tesla K80] [1
0de:102d] (rev a1)
85:00.0 3D controller [0302]: NVIDIA Corporation GK210GL [Tesla K80] [1
0de:102d] (rev a1)
86:00.0 3D controller [0302]: NVIDIA Corporation GK210GL [Tesla K80] [1
0de:102d] (rev a1)
89:00.0 3D controller [0302]: NVIDIA Corporation GK210GL [Tesla K80] [1
0de:102d] (rev a1)
8a:00.0 3D controller [0302]: NVIDIA Corporation GK210GL [Tesla K80] [1
0de:102d] (rev a1)
```

3. Execute the following command to retrieve the names of all nodes and find the GPU node name.

```
kubectl get nodes -o wide
```

4. Execute the following command to obtain the resourceName. The `nvidia.com/GK210GL_TESLA_K80` part in the output is the value of resourceName.

```
kubectl get node <gpu-node-name> -o jsonpath='{.status.allocatable}' #
Replace <gpu-node-name> with the GPU node name obtained from Step 3
```

Output information:

```
{"cpu":"39","devices.kubevirt.io/kvm":"1k","devices.kubevirt.io/tun":"1
k","devices.kubevirt.io/vhost-net":"1k","ephemeral-storage":"4265627841
65","hugepages-1Gi":"0","hugepages-2Mi":"0","memory":"122915848Ki","nvi
dia.com/GK210GL_TESLA_K80":"8","pods":"256"}
```


5. Execute the following command to add the passthrough GPU.

   Note: When replacing the <pci-devices-id> part in the command below with the pciDeviceSelector value obtained in Step 2, **all letters in the pciDeviceSelector must be converted to uppercase**. For example, if the pciDeviceSelector value obtained is `10de:102d`, it should be replaced with `export DEVICE=10DE:102D`.

   - Adding a single GPU card

     ```
     export DEVICE=<pci-devices-id> # Replace <pci-devices-id> with the pc
     iDeviceSelector obtained in Step 2, e.g.: export DEVICE=10DE:102D
     export RESOURCE=<resource-name> # Replace <resource-name> with the re
     sourceName obtained in Step 4, e.g.: export RESOURCE=nvidia.com/GK210
     GL_TESLA_K80

     kubectl patch hco kubevirt-hyperconverged -n kubevirt --type='json' -
     p='
     [
       {
         "op": "add",
         "path": "/spec/permittedHostDevices",
         "value": {
           "pciHostDevices": [
             {
               "externalResourceProvider": true,
               "pciDeviceSelector": "'"$DEVICE"'",
               "resourceName": "'"$RESOURCE"'"
             }
           ]
         }
       }
     ]'
     ```

   - Adding multiple GPU cards

     Note: When adding multiple GPU cards, each pciDeviceSelector value used to replace <pci-devices-id> must be unique.

```
export DEVICE1=<pci-devices-id1> # Replace <pci-devices-id1> with the pciDeviceSelector obtained in Step 2
export RESOURCE1=<resource-name1> # Replace <resource-name1> with the resourceName obtained in Step 4
export DEVICE2=<pci-devices-id2> # Replace <pci-devices-id2> with the pciDeviceSelector obtained in Step 2
export RESOURCE2=<resource-name2> # Replace <resource-name2> with the resourceName obtained in Step 4

kubectl patch hco kubevirt-hyperconverged -n kubevirt --type='json' -p='
[
  {
    "op": "add",
    "path": "/spec/permittedHostDevices",
    "value": {
      "pciHostDevices": [
        {
          "externalResourceProvider": true,
          "pciDeviceSelector": "'"$DEVICE"'",
          "resourceName": "'"$RESOURCE"'"
        },
        {
          "externalResourceProvider": true,
          "pciDeviceSelector": "'"$DEVICE2"'",
          "resourceName": "'"$RESOURCE2"'"
        }
      ]
    }
  }
]'
```

- Adding new GPU cards after already adding GPU cards

```
export DEVICE=<pci-devices-id> # Replace <pci-devices-id> with the pc
iDeviceSelector obtained in Step 2
export RESOURCE=<resource-name> # Replace <resource-name> with the re
sourceName obtained in Step 4
export INDEX=<index> # index is a zero-based array index, use the num
ber to replace <index>, for example: if one GPU card has already been
added, and now you want to add another one, index should be 1, i.e.,
export INDEX=1

kubectl patch hco kubevirt-hyperconverged -n kubevirt --type='json' -
p='
[
  {
    "op": "add",
    "path": "/spec/permittedHostDevices/pciHostDevices/'"${INDEX}"'",
    "value": {
      "externalResourceProvider": true,
      "pciDeviceSelector": "'"$DEVICE"'",
      "resourceName": "'"$RESOURCE"'"
    }
  }
]'
```

Result Verification

After completing the above configuration steps, if the corresponding physical GPU can be selected when creating the virtual machine, it indicates that the physical GPU passthrough environment has been successfully prepared.

Note: If physical GPU passthrough needs to be configured, please enable the relevant features in advance.

1. Go to **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machines**.
3. Click **Create Virtual Machine**.
4. Configure the **Physical GPU (Alpha)** parameter for the virtual machine.

Parameter	Description
Physical GPU (Alpha)	Select the model of the configured physical GPU. Only one physical GPU can be assigned to each virtual machine.

5. At this point, the physical GPU passthrough environment has been successfully prepared.

Related Operations

Delete the Virtual Machine with Passthrough GPU

1. Go to **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machines**.
3. In the list page, click the ⋮ on the right side of the virtual machine to be deleted > **Delete**, or click the name of the virtual machine to be deleted to enter its detail information page, and click **Actions** > **Delete**.
4. Input the confirmation information to delete the virtual machine with passthrough GPU.

Remove GPU-related Configuration from KubeVirt

1. On the corresponding cluster Master node for the GPU, use the CLI tool to execute the following command to remove the GPU-related configuration from KubeVirt.

```
kubectl patch hco kubevirt-hyperconverged -n kubevirt --type='json' -p
='[{"op": "remove", "path": "/spec/permittedHostDevices"}]'
```

2. After deletion, if it is not possible to choose the corresponding physical GPU model when creating a virtual machine through **Container Platform**, it indicates that the deletion was successful. Please refer to Select Physical GPU Model for the specific steps to create a virtual machine.

Uninstall gpu-operator

1. Use the CLI tool on the corresponding cluster Master node for the GPU to execute the following command to uninstall the gpu-operator.

   ```
   kubectl -n gpu-system delete apprelease gpu-operator
   ```

   Output information:

   ```
   apprelease.operator.alauda.io "gpu-operator" deleted
   ```

2. Execute the command, and if you receive a response similar to the one below, it indicates that the gpu-operator has been successfully uninstalled.

   ```
   kubectl -n gpu-system get apprelease gpu-operator
   ```

   Output information:

   ```
   Error from server (NotFound): appreleases.operator.alauda.io "gpu-opera
   tor" not found
   ```

Configuring High Availability for Virtual Machines

TOC

Overview

Hardware is imperfect and software contains bugs. When node-level failures, such as the kernel hangs or network interface controllers (NICs) fail, the work required from the cluster does not decrease, and workloads from affected nodes need to be restarted somewhere.

However, some workloads, such as ReadWriteOnce (RWO) volumes and StatefulSets, might require at-most-one semantics.

Failures affecting these workloads risk data loss, corruption, or both. It is important to ensure that the node reaches a safe state, known as fencing before initiating recovery of the workload, known as remediation and ideally, recovery of the node also.

It is not always practical to depend on administrator intervention to confirm the true status of the nodes and workloads. To facilitate such intervention, Alauda Container Platform provides multiple components for the automation of failure detection, fencing and remediation.

Glossary

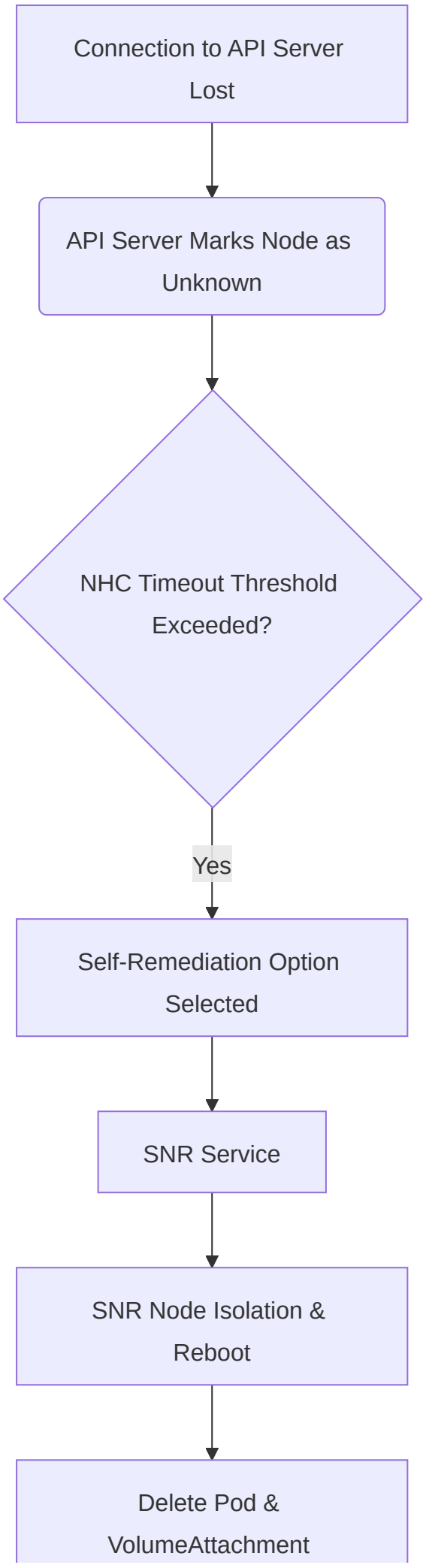
Acronym	Term
SNR	Self Node Remediation
NHC	Node Health Check

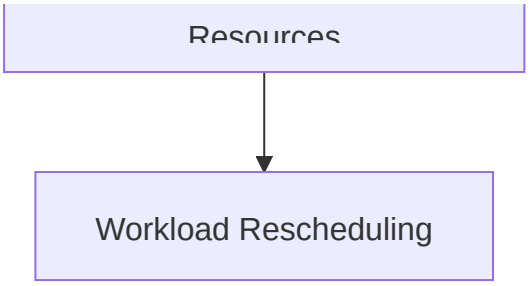
Component Overview

- **Self Node Remediation Operator**
  The Self Node Remediation Operator is a Alauda Container Platform add-on Operator that implements an external system of fencing and remediation that reboots unhealthy nodes and deletes resources, such as Pods and VolumeAttachments. The reboot ensures that the workloads are fenced, and the resource deletion accelerates the rescheduling of affected workloads. Unlike other external systems, Self Node Remediation does not require any management interface, like, for example, Intelligent Platform Management Interface (IPMI) or an API for node provisioning.
  Self Node Remediation can be used by failure detection systems, like Machine Health Check or Node Health Check.
- **Node Health Check Operator**
  The Node Health Check Operator is a Alauda Container Platform add-on Operator that implements a failure detection system that monitors node conditions. It does not have a built-in fencing or remediation system and so must be configured with an external system

that provides these features. By default, it is configured to utilize the Self Node Remediation system.

Flow of events during fencing and remediation


Connection to API Server Lost
API Server Marks Node as Unknown
NHC Timeout Threshold Exceeded?
Yes
Self-Remediation Option Selected
SNR Service
SNR Node Isolation & Reboot
Delete Pod & VolumeAttachment



Procedure

1 Operator Listing

- **Download** the **Alauda Build of SelfNodeRemediation** installation package corresponding to your platform architecture.
- **Upload** the **Alauda Build of SelfNodeRemediation** installation package using the Upload Packages mechanism.
- **Download** the **Alauda Build of NodeHealthCheck** installation package corresponding to your platform architecture.
- **Upload** the **Alauda Build of NodeHealthCheck** installation package using the Upload Packages mechanism.

2 Deploying Self Node Remediation Operator

1. Login, go to the **Administrator** page.
2. Click **Marketplace** > **OperatorHub** to enter the **OperatorHub** page.
3. Find the **Alauda Build of SelfNodeRemediation**, click **Install**, and navigate to the **Install Alauda Build of SelfNodeRemediation** page.
   Configuration Parameters:

Parameter	Recommended Configuration
Channel	The default channel is `stable`.
Installation Mode	`Cluster`: All namespaces in the cluster share a single Operator instance for creation and management, resulting in lower resource usage.

Parameter	Recommended Configuration
Installation Place	Select `Recommended`, Namespace only support **workload-availability**.
Upgrade Strategy	`Manual`: When there is a new version in the Operator Hub, manual confirmation is required to upgrade the Operator to the latest version.

3 Configuring Self Node Remediation Operator(optional)

The Self Node Remediation Operator creates the `SelfNodeRemediationConfig` CR with the name `self-node-remediation-config`. The CR is created in the namespace of the Self Node Remediation Operator.

> **Note**
>
> A change in the `SelfNodeRemediationConfig` CR re-creates the Self Node Remediation daemon set.

The `SelfNodeRemediationConfig` CR resembles the following YAML file:

```yaml
apiVersion: self-node-remediation.medik8s.io/v1alpha1
kind: SelfNodeRemediationConfig
metadata:
  name: self-node-remediation-config
  namespace: workload-availability
spec:
  safeTimeToAssumeNodeRebootedSeconds: 180
  watchdogFilePath: /dev/watchdog
  isSoftwareRebootEnabled: true
  apiServerTimeout: 15s
  apiCheckInterval: 5s
  maxApiErrorThreshold: 3
  peerApiServerTimeout: 5s
  peerDialTimeout: 5s
  peerRequestTimeout: 5s
  peerUpdateInterval: 15m
  hostPort: 30001
  customDsTolerations:
  - effect: NoSchedule
    key: node-role.kubernetes.io.infra
    operator: Equal
    value: "value1"
    tolerationSeconds: 3600
```

Parameters

Parameter	Description
safeTimeToAssumeNodeRebootedSeconds	Specify an optional time duration that the Operator waits before recovering affected workloads running on an unhealthy node. Starting replacement pods while they are still running on the failed node can lead to data corruption and a violation of run-once semantics. The Operator calculates a minimum duration using the values in the ApiServerTimeout, ApiCheckInterval, MaxApiErrorThreshold, PeerDialTimeout, and PeerRequestTimeout fields, as well as

Parameter	Description
	the watchdog timeout and the cluster size at the time of remediation.
watchdogFilePath	Specify the file path of the watchdog device in the nodes. If you enter an incorrect path to the watchdog device, the Self Node Remediation Operator automatically detects the softdog device path. If a watchdog device is unavailable, the `SelfNodeRemediationConfig` CR uses a software reboot.
isSoftwareRebootEnabled	Specify if you want to enable software reboot of the unhealthy nodes. By default, the value of `isSoftwareRebootEnabled` is set to `true`. To disable the software reboot, set the parameter value to `false`.
apiServerTimeout	Specify the timeout duration to check connectivity with each API server. When this duration elapses, the Operator starts remediation. The timeout duration must be greater than or equal to 10 milliseconds.
apiCheckInterval	Specify the frequency to check connectivity with each API server. The timeout duration must be greater than or equal to 1 second.
maxApiErrorThreshold	Specify a threshold value. After reaching this threshold, the node starts contacting its peers. The threshold value must be greater than or equal to 1 second.
peerApiServerTimeout	Specify the duration of the timeout for the peer to connect the API server. The

Parameter	Description
	timeout duration must be greater than or equal to 10 milliseconds.
peerDialTimeout	Specify the duration of the timeout for establishing connection with the peer. The timeout duration must be greater than or equal to 10 milliseconds.
peerRequestTimeout	Specify the duration of the timeout to get a response from the peer. The timeout duration must be greater than or equal to 10 milliseconds.
peerUpdateInterval	Specify the frequency to update peer information such as IP address. The timeout duration must be greater than or equal to 10 seconds.
hostPort	Specify an optional value to change the port that Self Node Remediation agents use for internal communication. The value must be greater than 0. The default value is port 30001.
customDsTolerations	Specify custom toleration Self Node Remediation agents that are running on the DaemonSets to support remediation for different types of nodes.

Note

- The Self Node Remediation Operator creates the CR by default in the deployment namespace.
- The name for the CR must be `self-node-remediation-config`.
- You can only have one `SelfNodeRemediationConfig` CR.

- Deleting the `SelfNodeRemediationConfig` CR disables Self Node Remediation.

4 Configuring Self Node Remediation Template(optional)

The Self Node Remediation Operator also creates the SelfNodeRemediationTemplate Custom Resource Definition (CRD). This CRD defines the remediation strategy for the nodes that is aimed to recover workloads faster. The following remediation strategies are available:

- **Automatic**
  This remediation strategy simplifies the remediation process by letting the Self Node Remediation Operator decide on the most suitable remediation strategy for the cluster. This strategy checks if the `OutOfServiceTaint` strategy is available on the cluster. If the `OutOfServiceTaint` strategy is available, the Operator selects the `OutOfServiceTaint` strategy. If the `OutOfServiceTaint` strategy is not available, the Operator selects the `ResourceDeletion` strategy. `Automatic` is the default remediation strategy.
- **ResourceDeletion**
  This remediation strategy removes the pods on the node, rather than the removal of the node object.
- **OutOfServiceTaint**
  This remediation strategy implicitly causes the removal of the pods and associated volume attachments on the node, rather than the removal of the node object. It achieves this by placing the `OutOfServiceTaint` strategy on the node.

The Self Node Remediation Operator creates the SelfNodeRemediationTemplate CR for the strategy self-node-remediation-automatic-strategy-template, which the Automatic remediation strategy uses.

The SelfNodeRemediationTemplate CR resembles the following YAML file:

```
apiVersion: self-node-remediation.medik8s.io/v1alpha1
kind: SelfNodeRemediationTemplate
metadata:
  creationTimestamp: "2022-03-02T08:02:40Z"
  name: self-node-remediation-<remediation_object>-deletion-template
  namespace: workload-availability
spec:
  template:
    spec:
      remediationStrategy: <remediation_strategy>
```

Parameters

Parameter	Description
remediation_strategy	Values: Automatic、ResourceDeletion、OutOfServiceTaint

5 Deploying Node Health Check Operator

1. Login, go to the **Administrator** page.
2. Click **Marketplace** > **OperatorHub** to enter the **OperatorHub** page.
3. Find the **Alauda Build of NodeHealthCheck**, click **Install**, and navigate to the **Install Alauda Build of NodeHealthCheck** page.
   Configuration Parameters:

Parameter	Recommended Configuration
Channel	The default channel is `stable`.
Installation Mode	`Cluster`: All namespaces in the cluster share a single Operator instance for creation and management, resulting in lower resource usage.
Installation Place	Select `Recommended`, Namespace only support **workload-availability**.

Parameter	Recommended Configuration
Upgrade Strategy	`Manual`: When there is a new version in the Operator Hub, manual confirmation is required to upgrade the Operator to the latest version.

6 Create NodeHealthCheck instance

Execute the following command on the cluster control node:

Command

```
cat << EOF | kubectl apply -f -
apiVersion: remediation.medik8s.io/v1alpha1
kind: NodeHealthCheck
metadata:
  name: nodehealthcheck-<name>
spec:
  minHealthy: <minHealthy>
  remediationTemplate:
    apiVersion: self-node-remediation.medik8s.io/v1alpha1
    kind: SelfNodeRemediationTemplate
    name: self-node-remediation-automatic-strategy-template
    namespace: workload-availability
  selector: <selector>
  unhealthyConditions:
    - duration: 300s
      status: 'False'
      type: Ready
    - duration: 300s
      status: Unknown
      type: Ready
EOF
```

Example

```
cat << EOF | kubectl apply -f -
apiVersion: remediation.medik8s.io/v1alpha1
kind: NodeHealthCheck
metadata:
  name: nodehealthcheck-worker
spec:
  minHealthy: 51%
  remediationTemplate:
    apiVersion: self-node-remediation.medik8s.io/v1alpha1
    kind: SelfNodeRemediationTemplate
    name: self-node-remediation-automatic-strategy-template
    namespace: workload-availability
  selector:
    matchExpressions:
      - key: node-role.kubernetes.io/control-plane
        operator: DoesNotExist
      - key: node-role.kubernetes.io/master
        operator: DoesNotExist
  unhealthyConditions:
    - duration: 300s
      status: 'False'
      type: Ready
    - duration: 300s
      status: Unknown
      type: Ready
EOF
```

Parameters:

Parameter	Description
name	resource name
minHealthy	Specify the minimum proportion of healthy nodes. Faulty nodes will only be repaired when the proportion of healthy nodes is greater than or equal to this value. The default value is 51%
selector	Specify LabelSelector to match the nodes to be inspected and self-repaired. Please avoid specifying control-plane and worker nodes simultaneously in the same instance

7 Verification(optional)

Simulate the failure of the running node of the virtual machine and confirm that the virtual machine is automatically scheduled to run on other nodes.

Create a VM Template from an Existing Virtual Machine

This document outlines how to create a reusable virtual machine (VM) template from an existing VM for rapid deployment of new VMs.

TOC

Prerequisites

- A properly deployed and configured KubeVirt environment.
- Access to the Web Console and kubectl tool.
- A configured VM with necessary software already installed.

Procedure

Step 1: Basic Configuration on the Virtual Machine

Inside the VM, perform the following steps:

- Install cloud-init↗.
- Install the `qemu-guest-agent`.
- Install any required software.

Once installations are complete, run the following commands to clean cloud-init data and shut down the VM:

```
cloud-init clean
shutdown -h now
```

Step 2: Create a VM Snapshot

Using the KubeVirt Web Console:

1. Navigate to **Virtualization** > **Virtual Machines**.
2. Select the VM intended to serve as a template.
3. Click **Actions**, select **Create Snapshot**, name your snapshot, and confirm.

Step 3: Retrieve Disk Snapshot Resource Name

Obtain the complete snapshot resource name using one of these methods:

- **Via Web Console:**
  - Navigate to **Storage** > **Volume Snapshots**.
  - Find and record the full snapshot resource name under "Data Source."
- **Using kubectl:**

```
kubectl get volumesnapshots -n <NAMESPACE>
```

Record the complete snapshot resource name from the output.

Step 4: Create a DataSource Resource

Create the following DataSource resource in the `kube-public` namespace, ensuring you replace placeholders with the actual snapshot name and namespace:

```
apiVersion: cdi.kubevirt.io/v1beta1
kind: DataSource
metadata:
  annotations:
    cpaas.io/display-name: MicroOS-Clone
  labels:
    virtualization.cpaas.io/image-os-arch: amd64
    virtualization.cpaas.io/image-os-type: linux
    virtualization.cpaas.io/storage-class: cephrbd
    virtualization.cpaas.io/access-mode: ReadWriteMany
    virtualization.cpaas.io/size: 30Gi
    virtualization.cpaas.io/volume-mode: Block
  name: microos-clone
  namespace: kube-public
spec:
  source:
    snapshot:
      name: <Your Snapshot Resource Name>
      namespace: <Your Snapshot Namespace>
```

Label Parameters Explanation:

Key	Possible Values	Description
virtualization.cpaas.io/image-os-arch	amd64, arm64	VM OS architecture
virtualization.cpaas.io/image-os-type	linux, windows	VM OS type

Key	Possible Values	Description
virtualization.cpaas.io/storage-class	storage class name	Default storage class, adjustable during VM creation
virtualization.cpaas.io/access-mode	ReadWriteOnce, ReadWriteMany	Disk access mode; use ReadWriteMany for VM live migration
virtualization.cpaas.io/size	Capacity (Gi, Ti, etc.)	Default disk size; specify appropriate size
virtualization.cpaas.io/volume-mode	Block, Filesystem	Disk volume mode; Block mode recommended for better performance

Important:

- Ensure the namespace is `kube-public`.
- These disk-related parameters can be modified during VM creation but providing defaults simplifies the process.

Step 5: Create a New VM Using the Template

1. Access the KubeVirt Web Console, go to **Container Platform** > **Virtualization** > **Virtual Machines**.
2. Click **Create Virtual Machine**.
3. Under **Image**, select **Image Instance** as the Provision Method.
4. Select your newly created DataSource from the dropdown.
5. Configure any additional parameters as required and complete the VM creation process.

You have now successfully created and deployed new VMs using your VM template.

Troubleshooting

Pod Migration and Recovery from Abnormal Shutdown of Virtual Machine Nodes

TOC

Problem Description

Whether the node is **gracefully shut down** or experiences an **abnormal crash**, the virtual machine Pods running on that node will not automatically migrate to other healthy nodes.

Cause Analysis

The platform implements a virtual machine solution based on the open-source component KubeVirt. However, from the perspective of KubeVirt, it cannot differentiate between an actual virtual machine crash and a connection failure caused by network or other issues. If virtual

machines are migrated to other nodes indiscriminately, it may lead to multiple instances of the same virtual machine existing concurrently.

Solutions

When maintaining virtual machine nodes, manual actions are required according to this document. For both **graceful shutdown** and **abnormal crash** situations, virtual machine Pods must be manually evicted or forcibly deleted.

Note: The following commands must be executed on the Master node of the corresponding cluster.

Migration of Virtual Machine Pods during Graceful Shutdown

1. In the CLI tool, execute the following command to obtain node information. The `NAME` field in the returned information is the `Node-Name`.

   ```
   kubectl get nodes
   ```

   Output:

   ```
   NAME             STATUS   ROLES                  AGE   VERSION
   1.1.1.211   Ready    control-plane,master   99d   v1.28.8
   ```

2. (Optional) Execute the following command to view the virtual machine instances under the node.

   ```
   kubectl get vmis --all-namespaces -o wide | grep <Node-Name>  # Replace <Node-Name> in the command with the Node-Name obtained in step 1
   ```

   Output:

```
test-test          vm-t-export-clone   13d     Running      1.1.1.1   1.
1.1.211   True    False
```

3. Before the graceful shutdown, execute the following command to evict all virtual machine Pods on the node to be shut down. If the output appears as follows, it indicates that the eviction was successful.

```
kubectl drain <Node-Name> --delete-local-data --ignore-daemonsets=true
--force --pod-selector=kubevirt.io=virt-launcher   # Replace <Node-Name
> in the command with the Node-Name of the node to be shut down
```

   Output:

```
Flag --delete-local-data has been deprecated, This option is deprecated
and will be deleted. Use --delete-emptydir-data.
node/1.1.1.211 cordoned
evicting pod test-test/virt-launcher-vm-t-export-clone-hmnkk
pod/virt-launcher-vm-t-export-clone-hmnkk evicted
node/1.1.1.211 drained
```

4. After all virtual machines are started on other nodes, shut down the node.
5. After the node is shut down and rebooted, execute the following command to mark the node as schedulable.

```
kubectl uncordon <Node-Name> # Replace <Node-Name> in the command with
the Node-Name of the shut down and rebooted node
```

   Output:

```
node/1.1.1.211 uncordoned
```

6. At this point, the original virtual machine instances on that node have been migrated to other healthy nodes, and this node is now available for new Pod scheduling after rebooting.

Recovery from Abnormal Shutdown

1. In the CLI tool, execute the following command to obtain node information. The `NAME` field in the returned information is the `Node-Name`.

```
kubectl get nodes
```

Output:

```
NAME            STATUS   ROLES                  AGE   VERSION
1.1.1.211   Ready    control-plane,master   99d   v1.28.8
```

2. Execute the following command to forcibly delete all virtual machine Pods on the node.

```
kubectl get po -A -l kubevirt.io=virt-launcher -o wide | grep <Node-Name> | awk '{print "kubectl delete pod --force -n " $1, $2}'  | bash  # Replace <Node-Name> in the command with the Node-Name of the node that crashed abnormally.
```

3. Execute the following command to delete the volume attachments on that node.

```
kubectl get volumeattachments.storage.k8s.io | grep <Node-Name> | awk '{print $1}' | xargs kubectl delete volumeattachments.storage.k8s.io  # Replace <Node-Name> in the command with the Node-Name of the node that crashed abnormally.
```

4. Execute the following command to query if there are Pods with the label kubevirt.io=virt-api on the node that crashed abnormally.

```
kubectl -n kubevirt get po -l kubevirt.io=virt-api -o wide | grep <Node-Name> # Replace <Node-Name> in the command with the Node-Name of the node that crashed abnormally.
```

If they exist, execute the following command to delete the Pods.

```
kubectl -n kubevirt get po -l kubevirt.io=virt-api -o name | xargs kubectl -n kubevirt delete --force --grace-period=0
```

5. Execute the following command to query if there are Pods with the label kubevirt.io=virt-controller on the node that crashed abnormally.

```
kubectl -n kubevirt get po -l kubevirt.io=virt-controller -o wide | gre
p <Node-Name> # Replace <Node-Name> in the command with the Node-Name o
f the node that crashed abnormally.
```

   If they exist, execute the following command to delete the Pods.

```
kubectl -n kubevirt get po -l kubevirt.io=virt-controller -o name | xar
gs kubectl -n kubevirt delete --force --grace-period=0
```

6. At this point, the virtual machine instances will be migrated to other healthy nodes after the node experiences an abnormal shutdown.

Hot Migration Error Messages and Solutions

Error Message	Cause	Solution
cannot migrate VMI which does not use masquerade, bridge with <annotation> VM annotation or a migratable plugin to connect to the pod network	The network configuration of the virtual machine does not support hot migration.	Please check the following configurations: • Check the CNI network plugin used by the current cluster; Kube-OVN is recommended. • Check whether the "kubevirt.io/allow-pod-bridge-network-live-migration": "true" annotation exists in the `metadata.annotations` and `spec.template.metadata.annotations` fields of the corresponding YAML file of the virtual machine; if not, please add it manually.
• cannot migrate VMI: Unable to determine if PVC <pvc name> is shared, live migration requires that all PVCs must be shared (using ReadWriteMany access mode)	The storage type of the virtual machine does not support multi-node read-write (RWX) access mode.	The parameters related to the virtual machine cannot be modified after creation. Therefore, please recreate the virtual machine and select a storage type that supports multi-node read-write (RWX); CephRBD block storage is recommended. If issues persist after recreation, please contact the relevant personnel for assistance.

Error Message	Cause	Solution
• cannot migrate VMI: PVC <pvc name> is not shared, live migration requires that all PVCs must be shared (using ReadWriteMany access mode) • cannot migrate VMI: Backend storage PVC is not RWX • cannot migrate VMI with non-shared HostDisk		
Other error messages	The virtual machine does not support hot migration.	Please contact the relevant personnel for assistance.

Network

Introduction

Introduction

Advantages

Guides

Configure Network

Configure IP

Connect to the virtual machine directly via IP

Add Internal Routes

How To

Control Virtual Machine Networ

Procedure

Configuring SR-IOV

Terminology

Constraints and Limitations

Configuring Support

Prerequisites

Introduction

ACP Virtualization With KubeVirt is deeply integrated with Kube-OVN, extending support for traditional virtual machine (VM) networking requirements and optimizing performance for specific scenarios.

TOC

Advantages

- IPv6 Support
  Full IPv6 Support.
- Static IP Retention
  Ensures VMs retain the same IP address after restarts, aligning with legacy VM usage patterns.
- Multi-Network Mode Support
  Supports multiple network modes such as container networks and SR-IOV, catering to diverse user scenarios.

Guides

Configure Network

Configure Network

TOC

Configure IP

Refs to Configure IP

Connect to the virtual machine directly via IP

Refs to Preparing Kube-OVN Underlay Physical Network

Add Internal Routes

Refs to Add Internal Routes

Practical Guide

Control Virtual Machine Networ

Procedure

Result Verification

Configuring SR-IOV

Terminology

Constraints and Limitations

Prerequisites

Procedures

Result Verification

Related Notes

Configuring Support

Prerequisites

Procedure

Control Virtual Machine Network Requests Through Network Policy

The platform's virtual machine solution is implemented based on the open-source component KubeVirt, which actually runs within Pods. By utilizing the functionality of Network Policies, it is possible to control the incoming and outgoing requests of virtual machines.

TOC

Procedure

1. Enter **Container Platform**.
2. In the left navigation bar, click **Network** > **Network Policies**.
3. Click **Create Network Policy**.
4. Configure the following parameters as needed.

Parameter	Description
Association Method	• **Compute Component**: Select the target compute component as needed; it is recommended to select **All** as the target compute component. • **Label Selector**: Match the Pods based on their labels.
Direction	• **Ingress**: Requests sent from the external to the Pod. • **Egress**: Requests sent from the Pod to the external; select this option if prohibiting the virtual machine from requesting a certain external address.
Protocol	Choose between TCP or UDP. **Note**: • When using domain names in the virtual machine to request external services, it is necessary to add a UDP protocol whitelist because DNS protocol uses UDP. • The form does not support configuring the ICMP protocol; once the whitelist rules are enabled, ICMP protocol will be disabled, which will result in the inability to perform Ping operations.
Access Ports	Specify which ports' traffic can be ingress or egress. If this field is left empty, traffic through all ports will be allowed by default. **Note**: It is necessary to allow ports 1053 and 53 for both UDP and TCP protocols here to permit DNS traffic egress; otherwise, domain name resolution will fail.
Remote Type	Specify the allowed remote types for access. Options include: compute component, namespace, and IP segments.

Parameter	Description
Exclude Remote	When the remote type is **IP Segment**, remove the specified IP from the whitelist (i.e., prohibit access). Single IP can be removed when input as `IP/32`. **Note**: This field only supports inputting IPs; if the corresponding IP of a domain name is unclear, use the command `curl -vvv <domain>` to request the domain and obtain the corresponding IP address from the returned information.

5. Click **Create**.

Result Verification

This document verifies the setup using a virtual machine to access www.example.com↗.

Step One: Create a Virtual Machine and Network Policy Allowing All Traffic Through

1. Create the virtual machine, please refer to Create Virtual Machine for detailed steps.
2. Configure the network policy in the command namespace of the virtual machine, adding whitelist rules for both TCP and UDP protocols, with the following parameters:
   - Whitelist for TCP Protocol:

Parameter	Description
Association Method	Select **Compute Component**.
Target Compute Component	Select **All**.
Direction	Select **Egress**.

Parameter	Description
Protocol	Select **TCP**.
Remote Type	Select **IP Segment**
Remote	Enter **0.0.0.0/0**, indicating that all traffic is allowed to egress.

- Whitelist Rules for UDP Protocol:

Parameter	Description
Direction	Select **Egress**.
Protocol	Select **UDP**.
Remote Type	Select **IP Segment**
Remote	Enter **0.0.0.0/0**, indicating that all traffic is allowed to egress.

3. After the network policy is created, log in to the virtual machine and execute the following command to request www.example.com↗.

```
curl www.example.com
```

4. The request is successful.

Step Two: Update Network Policy to Remove www.example.com↗ from Whitelist

1. Execute the following command to obtain the IP address for www.example.com↗, resulting in the IP address 93.184.215.14.

```
curl -vvv www.example.com
```

2. Update the network policy created in Step One, with the following updated parameters:

Parameter	Description
Exclude Remote	In the TCP protocol whitelist rules, fill in the exclude remote parameter with 93.184.215.14/32, indicating that IP address 93.184.215.14 is removed from the whitelist.

3. After updating the network policy, log in to the virtual machine and execute the following command to request www.example.com↗.

```
curl www.example.com
```

4. The request times out, indicating that the exclude remote functionality is effective.

Configuring SR-IOV

By configuring the physical server nodes to support the creation of virtual machines with SR-IOV (Single Root I/O Virtualization) network cards, lower latency for virtual machines is achieved, along with support for standalone IPv6 as well as dual-stack IPv4/IPv6 functionality.

TOC

Terminology

Term	Definition
Multus CNI	Acts as middleware for other CNI plugins to enable Kubernetes to support multiple network interfaces for Pods.
SR-IOV	Allows virtualization of the physical NIC on a node, splitting it into multiple VFs for use by Pods or virtual machines, providing superior network performance.
VF	A virtual device created from a physical PCI device; VFs can be allocated directly to virtual machines or containers, resembling independent physical PCI devices, significantly improving I/O performance.

Constraints and Limitations

The SR-IOV feature relies on glibc and only supports glibc versions 2.34 and above. However, both Kylin V10 and CentOS 7.x operating systems do not support this version, and thus, SR-IOV functionality cannot be used on these two operating systems.

Prerequisites

Obtain the following charts and images and upload them to the image repository. This document uses the repository address `build-harbor.example.cn` as an example. For specific methods to obtain the charts and images, please contact the relevant personnel.

Chart

- `build-harbor.example.cn/example/chart-sriov-network-operator:v3.15.0`

Images

- `build-harbor.example.cn/3rdparty/sriov/sriov-network-operator:4.13`
- `build-harbor.example.cn/3rdparty/sriov/sriov-network-operator-config-daemon:4.13`
- `build-harbor.example.cn/3rdparty/sriov/sriov-cni:4.13`
- `build-harbor.example.cn/3rdparty/sriov/ib-sriov-cni:4.13`
- `build-harbor.example.cn/3rdparty/sriov/sriov-network-device-plugin:4.13`
- `build-harbor.example.cn/3rdparty/sriov/network-resources-injector:4.13`
- `build-harbor.example.cn/3rdparty/sriov/sriov-network-operator-webhook:4.13`
- `build-harbor.example.cn/3rdparty/kubectl:v3.15.1`

Procedures

Note: All commands mentioned below are executed in the terminal.

1 Enabling SR-IOV in the Physical Machine's BIOS

Before configuration, use the following command to check the motherboard information.

```
dmidecode -t 1
# dmidecode 3.3
Getting SMBIOS data from sysfs.
SMBIOS 2.7 present.

Handle 0x0100, DMI type 1, 27 bytes
System Information
    Product Name: PowerEdge R620
    Version: Not Specified
    Serial Number: 7SJNF62
    UUID: 4c4c4544-0053-4a10-804e-b7c04f463632
    Wake-up Type: Power Switch
    SKU Number: SKU=NotProvided;ModelName=PowerEdge R620
    Family: Not Specified
```

The operation to enable SR-IOV in the BIOS varies among server manufacturers. Please refer to the corresponding manufacturer's documentation. Generally, the steps are as follows:

1. Reboot the server.
2. When the brand logo is displayed on the screen during BIOS POST, press the F2 key to enter the system setup.
3. Click **Processor Settings** > **Virtualization Technology**, and change **Virtualization Technology** setting to `Enabled`.
4. Click **Settings** > **Integrated devices**, and change **SR-IOV Global Enable** setting to `Enabled`.
5. Save the configuration and reboot the server.

2 Enabling IOMMU

The operation to enable IOMMU may vary across different operating systems. Please refer to the corresponding operating system documentation. This document uses CentOS as an example.

1. Edit the `/etc/default/grub` file and add `intel_iommu=on iommu=pt` to the `GRUB_CMDLINE_LINUX` configuration item.

```
GRUB_CMDLINE_LINUX="crashkernel=auto rd.lvm.lv=centos/root rhgb qu
iet intel_iommu=on iommu=pt"
```

2. Execute the following command to generate the `grub.cfg` file.

```
grub2-mkconfig -o /boot/grub2/grub.cfg
```

3. Reboot the server.
4. Execute the following command, and if the output shows `IOMMU enabled`, it indicates that the enabling is successful.

```
dmesg | grep -i iommu
```

3 Loading the VFIO Module in the System Kernel

1. Execute the following command to load the vfio-pci module.

```
modprobe vfio-pci
```

2. Once loaded, execute the following command. If the configuration information can be displayed normally, then the VFIO kernel module has been loaded successfully.

```
# For CentOS, execute the following command to check the VFIO load
ing status
lsmod | grep vfio
vfio_pci               41993  0
vfio_iommu_type1       22440  0
vfio                   32657  2 vfio_iommu_type1, vfio_pci
irqbypass              13503  2 kvm, vfio_pc


# For Ubuntu, execute the following command to check the VFIO load
ing status
cat /lib/modules/$(uname -r)/modules.builtin | grep vfio
kernel/drivers/vfio/vfio.ko
kernel/drivers/vfio/vfio_virqfd.ko
kernel/drivers/vfio/vfio_iommu_type1.ko
kernel/drivers/vfio/pci/vfio-pci-core.ko
kernel/drivers/vfio/pci/vfio-pci.ko
```

4 Creating VF Devices

1. Execute the following command to see the currently supported VF devices.

```
find /sys -name *vfs*

/sys/devices/pci0000:00/0000:00:03.0/0000:05:00.1/sriov_totalvfs
/sys/devices/pci0000:00/0000:00:03.0/0000:05:00.1/sriov_numvfs
/sys/devices/pci0000:00/0000:00:03.0/0000:05:00.0/sriov_totalvfs
/sys/devices/pci0000:00/0000:00:03.0/0000:05:00.0/sriov_numvfs
```

   The output information indicates as follows:

   - **0000:05:00 .1**: The PCI address of the SR-IOV physical NIC enp5s0f1.
   - **0000:05:00 .0**: The PCI address of the SR-IOV physical NIC enp5s0f0.
   - **sriov_totalvfs**: Number of supported VFs.
   - **sriov_numvfs**: Current number of VFs.

2. Execute the following command to get information on the physical machine's NIC.

```
ifconfig

enp5s0f0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.66.213  netmask 255.255.255.0  broadcast 192.
168.66.255
        inet6 1066::192:168:66:213  prefixlen 112  scopeid 0x0<glo
bal>
        inet6 fe80::a236:9fff:fe29:6c00  prefixlen 64  scopeid 0x2
0<link>
        ether a0:36:9f:29:6c:00  txqueuelen 1000  (Ethernet)
        RX packets 13889  bytes 1075801 (1.0 MB)
        RX errors 0  dropped 1603  overruns 0  frame 0
        TX packets 5057  bytes 440807 (440.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp5s0f1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::a236:9fff:fe29:6c02  prefixlen 64  scopeid 0x2
0<link>
        ether a0:36:9f:29:6c:02  txqueuelen 1000  (Ethernet)
        RX packets 1714  bytes 227506 (227.5 KB)
        RX errors 0  dropped 1604  overruns 0  frame 0
        TX packets 70  bytes 19241 (19.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

3. Execute the command `ethtool -i <NIC name>` to obtain the corresponding physical NIC's PCI address, as shown below.

```
ethtool -i enp5s0f0
driver: ixgbe
version: 5.15.0-76-generic
firmware-version: 0x8000030d, 14.5.8
expansion-rom-version:
bus-info: 0000:05:00.0     ## The PCI address of the enp5s0f0 NIC
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: yes


ethtool -i enp5s0f1
driver: ixgbe
version: 5.15.0-76-generic
firmware-version: 0x8000030d, 14.5.8
expansion-rom-version:
bus-info: 0000:05:00.1    ## The PCI address of the enp5s0f1 NIC
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: yes
```

4. Execute the following command to create a VF. This document takes configuring the enp5s0f1 NIC as an example. If multiple NICs need to be virtualized, all of them need to be configured.

```
cat /sys/devices/pci0000:00/0000:00:03.0/0000:05:00.1/sriov_totalv
fs   ## Check the number of supported VFs
63

echo 8 > /sys/devices/pci0000:00/0000:00:03.0/0000:05:00.1/sriov_n
umvfs  ## Set the current number of VFs

cat /sys/devices/pci0000:00/0000:00:03.0/0000:05:00.1/sriov_numvfs
## Check the current number of VFs
8
```

5. Execute the following command to check if the VFs were created successfully.

Note: You can see the configured 8 VF addresses, such as `05:10.1`. These VF addresses need to be supplemented with the **Domain Identifier**, resulting in the final format: `0000:05:10.1`.

```
lspci | grep Virtual
00:11.0 PCI bridge: Intel Corporation C600/X79 series chipset PCI
Express Virtual Root Port (rev 05)
05:10.1 Ethernet controller: Intel Corporation 82599 Ethernet Cont
roller Virtual Function (rev 01)
05:10.3 Ethernet controller: Intel Corporation 82599 Ethernet Cont
roller Virtual Function (rev 01)
05:10.5 Ethernet controller: Intel Corporation 82599 Ethernet Cont
roller Virtual Function (rev 01)
05:10.7 Ethernet controller: Intel Corporation 82599 Ethernet Cont
roller Virtual Function (rev 01)
05:11.1 Ethernet controller: Intel Corporation 82599 Ethernet Cont
roller Virtual Function (rev 01)
05:11.3 Ethernet controller: Intel Corporation 82599 Ethernet Cont
roller Virtual Function (rev 01)
05:11.5 Ethernet controller: Intel Corporation 82599 Ethernet Cont
roller Virtual Function (rev 01)
05:11.7 Ethernet controller: Intel Corporation 82599 Ethernet Cont
roller Virtual Function (rev 01)
```

5 Binding the VFIO Driver

1. Download the binding script, and execute the `python3 dpdk-devbind.py -b vfio-pci <VF address with domain identifier>` command to bind the 8 VFs of the enp5s0f1 NIC to the vfio-pci driver, as shown below.

```
python3 dpdk-devbind.py -b vfio-pci 0000:05:10.1
python3 dpdk-devbind.py -b vfio-pci 0000:05:10.3
python3 dpdk-devbind.py -b vfio-pci 0000:05:10.5
python3 dpdk-devbind.py -b vfio-pci 0000:05:10.7
python3 dpdk-devbind.py -b vfio-pci 0000:05:11.1
python3 dpdk-devbind.py -b vfio-pci 0000:05:11.3
python3 dpdk-devbind.py -b vfio-pci 0000:05:11.5
python3 dpdk-devbind.py -b vfio-pci 0000:05:11.7
```

2. After binding successfully, execute the following command to check the binding results. Look for the already bound VFs in the **Network devices using DPDK-compatible driver** area in the output result. Among them, the VF device ID is `10ed`.

```
python3 dpdk-devbind.py --status

Network devices using DPDK-compatible driver
============================================
0000:05:10.1 '82599 Ethernet Controller Virtual Function 10ed' drv
=vfio-pci unused=ixgbevf
0000:05:10.3 '82599 Ethernet Controller Virtual Function 10ed' drv
=vfio-pci unused=ixgbevf
0000:05:10.5 '82599 Ethernet Controller Virtual Function 10ed' drv
=vfio-pci unused=ixgbevf
0000:05:10.7 '82599 Ethernet Controller Virtual Function 10ed' drv
=vfio-pci unused=ixgbevf
0000:05:11.1 '82599 Ethernet Controller Virtual Function 10ed' drv
=vfio-pci unused=ixgbevf
0000:05:11.3 '82599 Ethernet Controller Virtual Function 10ed' drv
=vfio-pci unused=ixgbevf
0000:05:11.5 '82599 Ethernet Controller Virtual Function 10ed' drv
=vfio-pci unused=ixgbevf
0000:05:11.7 '82599 Ethernet Controller Virtual Function 10ed' drv
=vfio-pci unused=ixgbevf

Network devices using kernel driver
===================================
0000:01:00.0 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=eno
1 drv=tg3 unused=vfio-pci
0000:01:00.1 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=eno
2 drv=tg3 unused=vfio-pci
0000:02:00.0 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=eno
3 drv=tg3 unused=vfio-pci
0000:02:00.1 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=eno
4 drv=tg3 unused=vfio-pci
0000:05:00.0 'Ethernet 10G 2P X520 Adapter 154d' if=enp5s0f0 drv=i
xgbe unused=vfio-pci *Active*
0000:05:00.1 'Ethernet 10G 2P X520 Adapter 154d' if=enp5s0f1 drv=i
xgbe unused=vfio-pci

No 'Baseband' devices detected
==============================

No 'Crypto' devices detected
============================

No 'DMA' devices detected
```

```
=========================

No 'Eventdev' devices detected
==============================

No 'Mempool' devices detected
=============================

No 'Compress' devices detected
==============================

No 'Misc (rawdev)' devices detected
===================================

No 'Regex' devices detected
===========================
```

6 Deploying the Multus CNI Plugin

1. Go to **Administrator**.
2. In the left navigation bar, click **Cluster Management** > **Clusters**.
3. Click the name of the virtual machine cluster and switch to the **Plugins** tab.
   - Deploy the **Multus CNI** plugin.

7 Deploying the sriov-network-operator

Execute the following command to deploy the sriov-network-operator.

```
REGISTRY=<$registry>  # Replace the <$registry> part with the reposit
ory address where the sriov-network-operator image is located, for ex
ample: REGISTRY=build-harbor.example.cn
NICSELECTOR=["<nics>"] # Replace the <nics> part with the NIC names,
for example: NICSELECTOR=["ens802f1","ens802f2"], separate multiple w
ith commas
NUMVFS=<numVfs> # Replace the <numVfs> part with the number of VFs, f
or example: NUMVFS=8

cat <<EOF | kubectl create -f -
apiVersion: operator.alauda.io/v1alpha1
kind: AppRelease
metadata:
  annotations:
    auto-recycle: "true"
    interval-sync: "true"
  name: sriov-network-operator
  namespace: cpaas-system
spec:
  destination:
    cluster: ""
    namespace: "kube-system"
  source:
    charts:
    - name: <chartName> # Replace <chartName> with the actual chart p
ath, for example: name = example/chart-sriov-network-operator
      releaseName: sriov-network-operator
      targetRevision: v3.15.0
    repoURL: $REGISTRY
  timeout: 120
  values:
    global:
      registry:
        address: $REGISTRY
    networkNodePolicy:
      nicSelector: $NICSELECTOR
      numVfs: $NUMVFS
EOF
```

8 Setting Node Role Identifier Labels for Physical Nodes

Note: Before performing this operation, ensure that the Pod of the `sriov-network-operator` is running normally.

1. Go to **Administrator**.
2. In the left navigation bar, click **Cluster Management** > **Clusters**.
3. Click the cluster name and switch to the **Nodes** tab.
4. Click the physical node that supports SR-IOV ⋮ > **Update Node Labels**.
5. Set the node label as follows:
   - `node-role.kubernetes.io/worker: ""`
6. Click **Update**.

9 Checking if the Resources are Created Successfully

In the CLI tool, execute the command `kubectl -n cpaas-system get sriovnetworknodestates` to check if the `sriovnetworknodestates` resource has been created successfully. If you see similar output below, it indicates that creation was successful. If the resource creation fails, check if the Multus CNI plugin and sriov-network-operator have been deployed successfully.

```
kubectl -n cpaas-system get sriovnetworknodestates
NAME                     SYNC STATUS           AGE
192.168.254.88           Succeeded             5d22h
```

10 Setting SR-IOV Node Feature Labels for Physical Nodes

Note: Before performing this operation, ensure that the `sriovnetworknodestates` resource has been successfully created.

1. Go to **Administrator**.
2. In the left navigation bar, click **Cluster Management** > **Clusters**.
3. Click the cluster name and switch to the **Nodes** tab.
4. Click the physical node that supports SR-IOV ⋮ > **Update Node Labels**.

5. Set the node label as follows:

   - `feature.node.kubernetes.io/network-sriov.capable: "true"`

11 Checking NIC Device Support

1. Execute the command `lspci -n -s <VF address with domain identifier>` to obtain the current NIC device's vendor ID and device ID, as shown below.

   ```
   lspci -n -s 0000:05:00.1
   05:00.1 0200: 8086:154d (rev 01)
   ```

   The output indicates:

   - **8086**: Vendor ID.
   - **154d**: Device ID.

2. Execute the command `lspci -s <VF address with domain identifier> -vvv | grep Ethernet` to obtain the current NIC name, as shown below.

   ```
   lspci -s 0000:05:00.1 -vvv | grep Ethernet
   05:00.1 Ethernet controller: Intel Corporation Ethernet 10G 2P X52
   0 Adapter (rev 01)
   ```

3. In the cpaas-system namespace, locate the configuration file named `supported-nic-ids` with type ConfigMap, and check if the current NIC's configuration information is in the support list within its data section.
   Note: If the current NIC is not in the support list, you need to refer to Step 4 to add the NIC to the configuration file. If the current NIC is already in the support list, skip Step 4.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: supported-nic-ids
  namespace: cpaas-system
data:
  Broadcom_bnxt_BCM57414_2x25G: 14e4 16d7 16dc
  Broadcom_bnxt_BCM75508_2x100G: 14e4 1750 1806
  Intel_i40e_10G_X710_SFP: 8086 1572 154c
  Intel_i40e_25G_SFP28: 8086 158b 154c
  Intel_i40e_40G_XL710_QSFP: 8086 1583 154c
  Intel_i40e_X710_X557_AT_10G: 8086 1589 154c
  Intel_i40e_XXV710: 8086 158a 154c
  Intel_i40e_XXV710_N3000: 8086 0d58 154c
  Intel_ice_Columbiaville_E810: 8086 1591 1889
  Intel_ice_Columbiaville_E810-CQDA2_2CQDA2: 8086 1592 1889
  Intel_ice_Columbiaville_E810-XXVDA2: 8086 159b 1889
  Intel_ice_Columbiaville_E810-XXVDA4: 8086 1593 1889
```

4. Add the current NIC to the data section of the support list in the format `<NIC Name>: <Vendor ID> <Device ID> <VF Device ID>`, as shown below.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: supported-nic-ids
  namespace: cpaas-system
data:
  Broadcom_bnxt_BCM57414_2x25G: 14e4 16d7 16dc
  Broadcom_bnxt_BCM75508_2x100G: 14e4 1750 1806

  Intel_Corporation_X520: 8086 154d 10ed            ## Add new NIC
information

  Intel_i40e_10G_X710_SFP: 8086 1572 154c
  Intel_i40e_25G_SFP28: 8086 158b 154c
  Intel_i40e_40G_XL710_QSFP: 8086 1583 154c
  Intel_i40e_X710_X557_AT_10G: 8086 1589 154c
  Intel_i40e_XXV710: 8086 158a 154c
  Intel_i40e_XXV710_N3000: 8086 0d58 154c
  Intel_ice_Columbiaville_E810: 8086 1591 1889
  Intel_ice_Columbiaville_E810-CQDA2_2CQDA2: 8086 1592 1889
  Intel_ice_Columbiaville_E810-XXVDA2: 8086 159b 1889
  Intel_ice_Columbiaville_E810-XXVDA4: 8086 1593 1889
```

Parameter configuration explanation:

- **Intel_Corporation_X520**: The name of the NIC, which can be customized.
- **8086**: Vendor ID.
- **154d**: Device ID.
- **10ed**: VF Device ID, which can be found in the binding results.

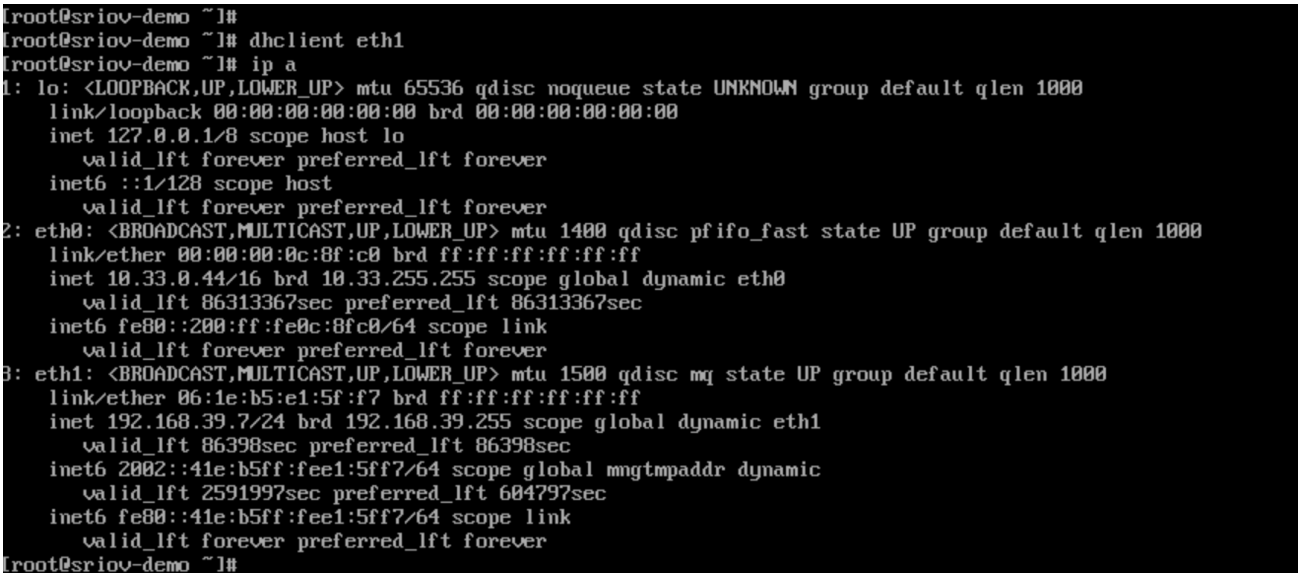
12 Configuring IP Address

Log in to the switch to configure DHCP (Dynamic Host Configuration Protocol).

Note: If it is not possible to use DHCP, please manually configure the IP address in the virtual machine.

Result Verification

1. Go to **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machines**.
3. Click **Create Virtual Machine**, and when adding an auxiliary network card, select **SR-IOV** as the **Network Type**.
4. Complete the creation of the virtual machine.
5. Access the virtual machine through VNC, you should see that eth1 has successfully obtained an IP address, indicating that the configuration has been successful.



Related Notes

Kernel Parameter Configuration for CentOS Virtual Machines

After the CentOS virtual machine uses the SR-IOV NIC, it is necessary to modify the kernel parameters for the corresponding NIC. The specific steps are as follows.

1. Open a terminal and execute the following command to modify the kernel parameters for the corresponding NIC. Replace the `<NIC Name>` part of the command with the actual NIC name.

```
sysctl -w net.ipv4.conf.<NIC Name>.rp_filter=2
echo "net.ipv4.conf.<NIC Name>.rp_filter=2" >> /etc/sysctl.conf
```

2. Execute the following command to load and apply all kernel parameter commands from the /etc/sysctl.conf file, so that the kernel configuration takes effect. When the value in the output information is 2, it indicates that the modification was successful.

```
sysctl -p
```

Output information:

```
net.ipv4.conf.<NIC Name>.rp_filter = 2
```

Configuring Virtual Machines to Use Network Binding Mode for IPv6 Support

The network binding mode is a plugin extension mechanism for virtual machine networking. By default, the platform uses a plugin called ManagedTap to enable IPv6 support for virtual machines. This plugin allows virtual machines to obtain IP addresses through the CNI's DHCP Server. Therefore, as long as the CNI's DHCP Server supports IPv6, virtual machines will also gain IPv6 capabilities.

Currently, we use Kube-OVN as the CNI. Since Kube-OVN's DHCP Server has full IPv6 support, virtual machines can achieve robust IPv6 functionality through the combination of ManagedTap and Kube-OVN.

TOC

Prerequisites

- ACP version must be v4.0.0 or higher.

- Kube-OVN is used as the CNI, and the virtual machine subnet is configured as Underlay.

Procedure

1 Add IPv6 Configuration to the Virtual Machine Subnet

```
kubectl edit subnet <subnet-name>
```

Add the following parameters under `spec`:

```
spec:
  enableDHCP: true
  enableIPv6RA: true
  u2oInterconnection: true
```

2 Create a Virtual Machine Using Network Binding Mode in the web console

When creating a virtual machine, select **Network Binding** as the network mode.

3 Access the Virtual Machine via VNC and Configure the Network Interface

For CentOS systems, edit the `/etc/sysconfig/network-scripts/ifcfg-enp1s0` file and add the following configuration:

```
IPV6INIT=yes
DHCPV6C=yes
IPV6_AUTOCONF=yes
```

restart network

```
systemctl restart network
```

4 Configure IPv6 Default Route

If the switch is configured to send Router Advertisement (RA) messages, manual route configuration is not required. The default route can be automatically learned through RA messages from the switch.

```
ip r r default via <subnet-v6-gateway>
```

Storage

Introduction

Introduction

Advantages

Guides

Managing Virtual Disks

Creating a Virtual Disk

Mounting a Virtual Disk

Expanding a Virtual Disk

Unmounting a Virtual Disk

Deleting a Virtual Disk

Introduction

ACP Virtualization with KubeVirt Storage provides persistent storage capabilities for virtual machines (VMs) by seamlessly integrating with Kubernetes-native storage mechanisms. It leverages **PersistentVolumeClaim** (PVC) to store VM disk data and utilizes the **Container Storage Interface** (CSI) to integrate with various storage systems. Additionally, the **Containerized Data Importer** (CDI) is employed to initialize VM disk data. Building on these foundations, the platform extends advanced functionalities for VM disk management, enabling comprehensive lifecycle control.

TOC

Advantages

- User-Friendly Operations
  Most VM disk operations can be easily performed via the **Web UI**, minimizing the need for CLI expertise.
- VM Disk Lifecycle Management
  Configure whether VM disks should be automatically deleted when the associated VM is terminated.

Guides

Managing Virtual Disks

Managing Virtual Disks

Data disks can be used to meet the data persistence requirements of the business.

TOC

Creating a Virtual Disk

Create a **data disk** for the virtual machine. Only **one** virtual disk can be added at a time; if multiple disks are needed, please repeat this operation.

Note: Virtual disks can be mounted online when the virtual machine is in **running** state.

Procedures

1. Access the **Container Platform**.
2. In the left navigation bar, click on **Virtualization** > **Virtual Disk**.
3. Click on **Create Virtual Disk**.
4. Configure the information based on the following instructions.

Parameter	Description
Volume Mode	- **File System**: Mount the disk in a way that mounts the file directory. - **Block Device**: Mount the disk as a block device.
Storage Class	The platform maintains virtual machine disks by automatically creating and managing persistent volume claims. You need to specify the storage class required for dynamically creating persistent volume claims. Different storage classes support different volume modes. If there are no available storage classes for the selected volume mode, please contact the administrator for addition.
Delete with VM	If enabled, the disk data will also be deleted when the virtual machine is deleted.
Mount	- **Do Not Mount**: Only create the virtual disk; it can be mounted later when needed. - **Mount to VM**: Select the target virtual machine to which the virtual disk needs to be mounted.

5. Click on **Create**.

Mounting a Virtual Disk

Mount the **data disk** to a virtual machine, attaching the already created virtual disk to the target virtual machine.

Note: Virtual disks can be mounted online when the virtual machine is in **running** state.

Procedures

1. Access the **Container Platform**.
2. In the left navigation bar, click on **Virtualization** > **Virtual Disk**.
3. Click ⋮ > **Mount** next to the virtual disk to be mounted.
4. Select the target virtual machine and click **Mount**.

Expanding a Virtual Disk

Expand the **system disk** and **data disk** already mounted to the virtual machine.

Procedures

1. Access the **Container Platform**.
2. In the left navigation bar, click on **Virtualization** > **Virtual Machine**.
3. Click the name of the virtual machine to enter the **Details** page.
4. In the **Virtual Disk** area, find the disk to be expanded and click ⋮ > **Expand**.
5. Enter the new capacity and click **Expand**.

Unmounting a Virtual Disk

Unmount the **data disk** from the virtual machine; only virtual machines in the **stopped** state can unmount disks.

Procedures

1. Access the **Container Platform**.
2. In the left navigation bar, click on **Virtualization** > **Virtual Disk**.
3. Click ⋮ > **Unmount** next to the virtual disk to be unmounted and confirm.

Deleting a Virtual Disk

Deletion is only supported when the virtual disk is in an unmounted state.

Note: System disks cannot be deleted.

Procedures

1. Access the **Container Platform**.
2. In the left navigation bar, click on **Virtualization** > **Virtual Disk**.
3. Click ⋮ > **Delete** next to the virtual disk to be deleted and confirm.

Backup and Recovery

Introduction

Introduction

Application Scenarios

Usage Limitations

Guides

Using Snapshots

Prerequisites

Notes

Creating a Snapshot

Rolling Back a Snapshot

Deleting a Snapshot

Using Velero

Prerequisites

Operation Steps

Introduction

ACP Virtualization With Kubevirt provides virtual machine snapshot capabilities, allowing users to back up and restore VMs via snapshots.

TOC

Application Scenarios

- Disaster Recovery & Failure Rollback
  When a virtual machine experiences data loss due to hardware failures, human errors (e.g., accidental file deletion), or malicious attacks (e.g., ransomware), snapshots serve as the last line of defense to restore operations.

Usage Limitations

- Creating a snapshot requires stopping the virtual machine first.
- The PVC (Persistent Volume Claim) used by the virtual machine disk must be configured with a multi-node shared access mode.

Guides

Using Snapshots

Using Velero

Using Snapshots

A virtual machine snapshot saves the current state of the virtual machine, and can be used to restore the virtual machine to that state in the event of an unexpected failure.

TOC

Prerequisites

- The **Volume Snapshot** has been deployed by the administrator in the platform management.
- Virtual machine snapshots are based on volume snapshots. Ensure that at least one disk is bound to a storage class that supports volume snapshots, such as CephFS built-in storage.

Notes

If there are multiple storage types of the same kind in the cluster, for example, attaching multiple different sources of Ceph RBD storage, the disk snapshot functionality may not work properly when the virtual machine is using such storage.

Creating a Snapshot

The contents included in a virtual machine snapshot: virtual machine settings and the state of the disks that support volume snapshots.

Procedures

1. Access **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machines**.
3. Locate the virtual machine and click ⋮ > **Create Snapshot**.
4. Fill in the snapshot description. The description can help you document the current state of the virtual machine, such as `Initial Installation`, `Before Application Upgrade`.
5. Click **Create**. The time taken for the snapshot depends on network conditions and workload, please be patient.
6. Check the snapshot status.
   - When the snapshot changes to `Ready`, it indicates that the creation was successful.
   - If the snapshot remains in `Not Ready` status for a long time, click > View the reasons and troubleshoot, then recreate the snapshot.

Rolling Back a Snapshot

Roll back the virtual machine settings and the disks that support volume snapshots to the state at the time the snapshot was created. For example, disks added after the snapshot creation will be removed; modified disk data will be restored.

Notes

If there are disks bound to a storage class that supports the LVM mechanism (for example, TopoLVM), please confirm with the administrator that the reclamation policy for that storage class is set to **Retain** (`reclaimPolicy: Retain`) to use the snapshot rollback feature correctly.

Procedures

1. Access **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machines**.
3. Click on ***Virtual Machine Name***.
4. In the **Snapshots** tab, locate the snapshot and click ⋮ > **Rollback**.
5. Read the prompt information on the interface, and click **Rollback** after confirming everything is correct.
   Note: The rollback operation cannot be aborted or undone, please proceed with caution.
6. Click on the snapshot name to check in the “Snapshot Rollback Records” if the rollback has been completed. The time required for the rollback depends on network conditions and workload, please be patient.

Description

- If the rollback fails, the virtual machine state remains unchanged. You can start the virtual machine normally or attempt to roll back the snapshot again.
- If the virtual machine is started during the rollback process, it will revert to the state before it was stopped, and upon stopping the virtual machine again, it will continue rolling back to the state at the time of snapshot creation.
- To avoid operational conflicts, please ensure that the most recent rollback record has been completed before performing other operations on that virtual machine.

Deleting a Snapshot

Delete unnecessary virtual machine snapshots to free up disk resources.

Notes

When deleting a rolled-back virtual machine snapshot, if the virtual machine disk needs to copy data based on the snapshot (for example, TopoLVM), you must wait until a virtual machine based on the rollback version has been started before deleting, otherwise the virtual machine will fail to start.

Procedures

1. Access **Container Platform**.
2. In the left navigation bar, click **Virtualization** > **Virtual Machines**.
3. Click on ***Virtual Machine Name***.
4. In the **Snapshots** tab, locate the target snapshot and click ⋮ > **Delete**.
5. Read the prompt information and click **Delete** after confirming everything is correct.

Using Velero

Velero is an open-source Kubernetes cluster backup and migration tool by VMware. KubeVirt provides a Velero plugin to support virtual machine backup and restoration.

TOC

Prerequisites

- **Kubernetes Version**: 1.20 or higher (Velero requirement)
- **S3 Storage**: For Velero BackupStorageLocation, use platform-provided Ceph or MinIO object storage
- **Block Storage**: Platform-provided Ceph RBD

Operation Steps

Preparation

1. Deploy Velero on the ACP platform via **Platform Management** → **Cluster Management** → **Backup and Recovery** → **Backup Repository**. Use object storage details to create the backup repository.
2. Enable the CSI feature and add the KubeVirt plugin:

   ```
   kubectl edit deploy -n cpaas-system velero
   ```

   Add the following to the Velero deployment:

   ```
   containers:
   - args:
     - server
     - --uploader-type=restic
     - --namespace=cpaas-system
     - --features=EnableCSI
     command:
     - /velero
   ```

   Add to the initContainers section:

   ```
   initContainers:
   - image: registry.example.org/3rdparty/kubevirt/kubevirt-velero-plugin:
   v0.7.0
     imagePullPolicy: IfNotPresent
     name: velero-plugin-kubevirt
     resources: {}
     terminationMessagePath: /dev/termination-log
     terminationMessagePolicy: File
     volumeMounts:
     - mountPath: /target
       name: plugins
   ```

3. Verify the KubeVirt plugin installation:

```
POD=$(kubectl -n cpaas-system get pod -l app.kubernetes.io/name=velero
-o jsonpath='{.items[0].metadata.name}')
kubectl -n cpaas-system exec -ti "$POD" -- /velero get plugins | grep k
ubevirt
```

Example output:

```
kubevirt-velero-plugin/backup-datavolume-action              BackupIte
mAction
kubevirt-velero-plugin/backup-datavolume-action              BackupIte
mAction
kubevirt-velero-plugin/backup-virtualmachine-action          BackupIte
mAction
kubevirt-velero-plugin/backup-virtualmachine-action          BackupIte
mAction
kubevirt-velero-plugin/backup-virtualmachineinstance-action  BackupIte
mAction
kubevirt-velero-plugin/backup-virtualmachineinstance-action  BackupIte
mAction
kubevirt-velero-plugin/restore-pod-action                    RestoreIt
emAction
kubevirt-velero-plugin/restore-pod-action                    RestoreIt
emAction
kubevirt-velero-plugin/restore-pvc-action                    RestoreIt
emAction
kubevirt-velero-plugin/restore-pvc-action                    RestoreIt
emAction
kubevirt-velero-plugin/restore-vm-action                     RestoreIt
emAction
kubevirt-velero-plugin/restore-vm-action                     RestoreIt
emAction
kubevirt-velero-plugin/restore-vmi-action                    RestoreIt
emAction
kubevirt-velero-plugin/restore-vmi-action                    RestoreIt
emAction
```

4. Adjust node-agent to mount the host kubelet directory and enable privileged mode (required for data movement via `/var/lib/kubelet`):

```
kubectl edit ds -n cpaas-system node-agent
```

Update with:

```
securityContext:
  privileged: true
volumeMounts:
- mountPath: /host_pods
  mountPropagation: HostToContainer
  name: host-pods
- mountPath: /var/lib/kubelet/plugins
  mountPropagation: HostToContainer
  name: host-plugins
- mountPath: /scratch
  name: scratch
securityContext:
  runAsUser: 0
volumes:
- hostPath:
    path: /var/lib/kubelet/pods
    type: ""
  name: host-pods
- hostPath:
    path: /var/lib/kubelet/plugins
    type: ""
  name: host-plugins
- emptyDir: {}
  name: scratch
```

Backup

1. Create a virtual machine with multiple disks as needed.
2. Create a backup resource:

```yaml
apiVersion: velero.io/v1
kind: Backup
metadata:
  annotations:
    velero.io/resource-timeout: 10m0s
    velero.io/source-cluster-k8s-gitversion: v1.30.4
    velero.io/source-cluster-k8s-major-version: "1"
    velero.io/source-cluster-k8s-minor-version: "30"
  labels:
    velero.io/storage-location: default
  name: example-backup
  namespace: cpaas-system
spec:
  csiSnapshotTimeout: 10m0s
  defaultVolumesToFsBackup: false
  hooks: {}
  includedNamespaces:
  - example-namespace
  itemOperationTimeout: 4h0m0s
  metadata: {}
  snapshotMoveData: true
  storageLocation: default
  ttl: 720h0m0s
```

3. Wait for the backup to complete and verify:

```
POD=$(kubectl -n cpaas-system get pod -l app.kubernetes.io/name=velero
-o jsonpath='{.items[0].metadata.name}')
kubectl -n cpaas-system exec -ti "$POD" -- /velero backup describe exam
ple-backup --details
kubectl -n cpaas-system exec -ti "$POD" -- /velero backup get example-b
ackup
```

Example output:

```
NAME             STATUS                          ERRORS   WARNINGS   CREAT
ED                          EXPIRES   STORAGE LOCATION   SELECTOR
example-backup   WaitingForPluginOperations   0        0          2025-
01-25 14:14:08 +0000 UTC   29d       default            <none>
```

4. Check dataupload status for data movement:

```
kubectl get dataupload -n cpaas-system
```

Example output:

```
NAME                    STATUS      STARTED   BYTES DONE    TOTAL BYTES
STORAGE LOCATION   AGE   NODE
example-backup-fhc6b   Completed   11h       21474836480   21474836480
default            11h   192.168.254.66
example-backup-qwwzh   Completed   11h       21474836480   21474836480
default            11h   192.168.254.15
```

5. Verify backup completion:

```
POD=$(kubectl -n cpaas-system get pod -l app.kubernetes.io/name=velero
-o jsonpath='{.items[0].metadata.name}')
kubectl -n cpaas-system exec -ti "$POD" -- /velero backup get example-b
ackup
```

Example output:

```
NAME             STATUS      ERRORS   WARNINGS   CREATED
EXPIRES   STORAGE LOCATION   SELECTOR
example-backup   Completed   0        0          2025-01-25 14:14:08 +0
000 UTC   29d       default            <none>
```

6. Check the backup repository (e.g., MinIO) for directories:

```
mc ls <backup-repository>
```

Example output:

```
[2025-01-26 09:23:08 CST]     0B backups/
[2025-01-26 09:23:08 CST]     0B kopia/
[2025-01-26 09:23:08 CST]     0B restic/
```

Restore

1. Delete the original virtual machine.
2. Create a restore resource via **Platform Management** → **Cluster Management** → **Backup and Recovery** → **Restore Management** or manually:

```
apiVersion: velero.io/v1
kind: Restore
metadata:
  annotations:
    cpaas.io/description: ""
  finalizers:
  - restores.velero.io/external-resources-finalizer
  name: example-restore
  namespace: cpaas-system
spec:
  backupName: example-backup
  excludedResources:
  - nodes
  - events
  - events.events.k8s.io
  - backups.velero.io
  - restores.velero.io
  - resticrepositories.velero.io
  - csinodes.storage.k8s.io
  - volumeattachments.storage.k8s.io
  - backuprepositories.velero.io
  hooks: {}
  includedNamespaces:
  - example-namespace
  itemOperationTimeout: 4h0m0s
  namespaceMapping: {}
```

3. Verify the restore:

```
kubectl exec -ti -n cpaas-system velero-5df7bb7598-ljjrn -- /velero restore get
```

Check datadownload status:

```
kubectl get datadownload -n cpaas-system
```

Example output:

```
NAME       STATUS      STARTED    BYTES DONE    TOTAL BYTES    STORAGE LO
CATION    AGE    NODE
example-restore-7lfcm    Completed    11m       21474836480    21474836480
default             15m    192.168.254.66
example-restore-cjcd8    Completed    15m       21474836480    21474836480
default             15m    192.168.254.66
```

Restore status:

```
NAME    BACKUP           STATUS      STARTED                          COM
PLETED                       ERRORS    WARNINGS    CREATED
SELECTOR
aa      example-backup    Completed    2025-01-26 01:53:14 +0000 UTC    202
5-01-26 02:01:24 +0000 UTC    0          2           2025-01-26 01:53:14 +0
000 UTC    <none>
```

4. Confirm the virtual machine is restored and operational.

Cross-Cluster Restore

1. Deploy Velero on the new cluster and complete the preparation steps.
2. Configure the same backup repository (same bucket and directory) as the original cluster. The backup resource will automatically appear.
3. Follow the restore steps above.

Restore to a Different Namespace

Add a `namespaceMapping` field to the restore spec:

```yaml
spec:
  namespaceMapping:
    example-namespace: test-namespace
```

Restore to a Different Storage Class

1. Create a ConfigMap in the `cpaas-system` namespace before initiating the restore:

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: change-storage-class-config
  namespace: cpaas-system
  labels:
    velero.io/plugin-config: ""
    velero.io/change-storage-class: RestoreItemAction
data:
  vm-cephrbd: vm-topolvm-a
```

2. Ensure storage capabilities (e.g., cross-node access, RWX) are consistent.

3. Note: RWX mode modification is not currently supported.