Menu

Виртуализация

Обзор

Введение

Решение виртуальных машин с оркестрацией контейнеров

Особенности

Функциональные возможности продукта

Ограничения и условия

Features

Virtual Machine Images

Virtual Machines

Virtual Machine Networking

Backup and Recovery

Установка

Установка

Предварительные требования

Процедура

Объяснение квоты ресурсов

Образы

Введение

Преимущества

Руководства

Как сделать

Разрешения

Виртуальная машина

Введение

Преимущества

Руководства

Как сделать

Устранение неполадок

Сеть

Введение

Преимущества

Руководства

Как сделать

Хранение данных

Введение

Преимущества

Руководства

Резервное копирование и восстановление

Введение

Сценарии применения

Ограничения использования

Руководства

Menu

Обзор

Введение

Решение виртуальных машин с оркестрацией контейнеров

Особенности

Функциональные возможности продукта

Ограничения и условия

Features

Virtual Machine Images

Virtual Machines

Virtual Machine Networking

Backup and Recovery

Введение

Для предприятий, использующих архитектуру на основе виртуальных машин, переход к архитектуре на основе Kubernetes и контейнеров неизбежно требует модернизации приложений. Однако из-за таких ограничений, как необходимость непрерывной работы бизнеса или сложность изменения привычек разработки, предприятия часто не могут полностью отказаться от архитектуры виртуализации за короткий срок.

Поэтому решение, которое может единообразно настраивать, управлять и контролировать ресурсы контейнеров и виртуальных машин на одной платформе, становится особенно важным.

Содержание

Решение виртуальных машин с оркестрацией контейнеров

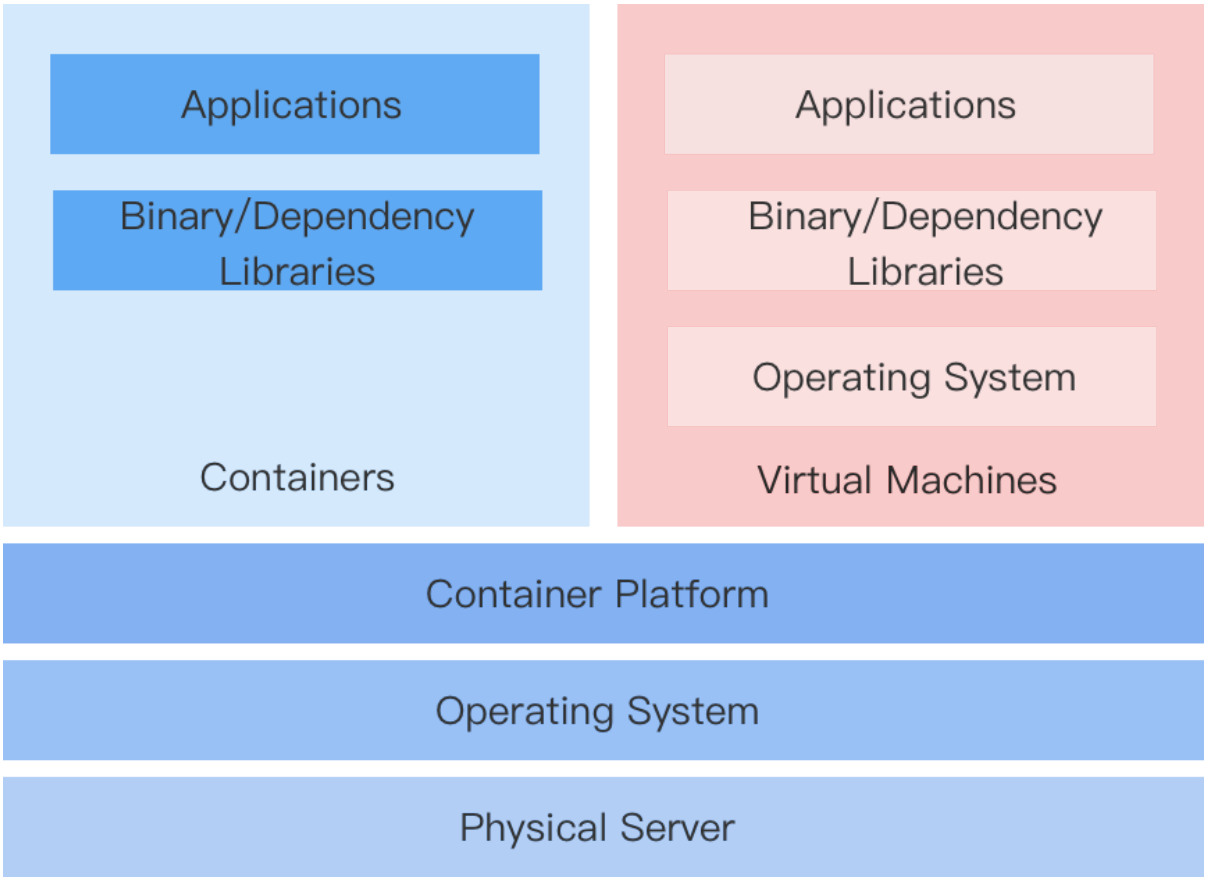
Особенности

Функциональные возможности продукта

Ограничения и условия

Решение виртуальных машин с оркестрацией контейнеров

Данная платформа реализует решение виртуальной машины (VMI, VirtualMachineInstance) на основе открытого компонента KubeVirt, что позволяет проще и быстрее создавать виртуальные машины с оркестрацией контейнеров и запускать виртуализированные приложения.



Особенности

Быстрая трансформация

Нет необходимости переписывать приложения или изменять образы. Достаточно упаковать существующее приложение в образ виртуальной машины формата qcow2 или raw и создать виртуальную машину на платформе с использованием этого образа, что позволяет развернуть приложение на контейнерной платформе.

Сохранение привычек работы

Контейнеризированные виртуальные машины можно управлять аналогично традиционным виртуальным машинам, не уделяя внимания внутренней реализации контейнеров, включая управление жизненным циклом виртуальной машины, дисками и сетями, а также управление снимками.

Сосуществование виртуализации и контейнеризации

- Унифицированная платформа поддерживает управление виртуализированными сервисами и одновременно обеспечивает планирование и управление контейнерами

на базе Kubernetes.

- При сохранении использования рабочих нагрузок виртуальных машин позволяет постепенно модернизировать контейнеризированные приложения.
- Разработка новых контейнеризированных приложений, которые должны взаимодействовать с виртуализированными приложениями, не затруднена.

Функциональные возможности продукта

- **Виртуальная машина**: Поддержка создания виртуальных машин с образами, выделенными администраторами, и управление ими, включая запуск и остановку виртуальных машин, управление снимками, удалённый вход в виртуальные машины и изменение конфигураций виртуальных машин.
- **Виртуальный диск**: Поддержка просмотра и управления информацией о дисках, созданных в текущем проекте, включая создание дисков, просмотр имён дисков, классов хранилища, ёмкостей и связанных виртуальных машин.
- **Снимки виртуальных машин**: Поддержка просмотра деталей, таких как статус снимков виртуальных машин, связанная виртуальная машина и время последнего отката.
- **Образы виртуальных машин**: Поддержка просмотра информации об образах виртуальных машин в текущем проекте, включая способ предоставления образа и операционную систему.
- **Ключевые пары**: Поддержка просмотра и управления ключевыми парами, созданными в текущем проекте, включая создание ключевых пар и просмотр списка связанных виртуальных машин.

Ограничения и условия

Реализация должна базироваться на кластере физических машин, при этом компоненты KubeVirt должны быть развернуты внутри кластера с включённой виртуализацией.

Платформа предоставляет возможность развертывания компонентов KubeVirt через Operator и интерфейс для включения виртуализации, при этом все связанные настройки выполняет администратор платформы.

Features

Содержание

Virtual Machine Images

Virtual Machines

Virtual Machine Networking

Backup and Recovery

Virtual Machine Images

- Управление образами виртуальных машин

  Платформа поддерживает создание, обновление и удаление образов виртуальных машин.

- Управление учетными данными образов

  Поддерживается создание, обновление и удаление учетных данных образов.

Virtual Machines

- Создание виртуальных машин/групп виртуальных машин

  Поддерживается быстрое создание отдельных виртуальных машин или групп виртуальных машин с одинаковыми конфигурациями.

- Управление виртуальными машинами

Поддерживаются различные операции управления, включая сброс пароля, обновление ключевых пар, обновление спецификаций, обновление меток/аннотаций, переустановку ОС и др.

- Управление дисками виртуальных машин

  Поддерживаются операции управления дисками, такие как создание, удаление, подключение, отключение и расширение дисков.

- Управление ключевыми парами

  Поддерживается создание, удаление и обновление SSH ключевых пар.

- Массовые операции с виртуальными машинами

  Поддерживаются массовые операции по запуску, остановке, перезапуску и удалению виртуальных машин.

- VNC вход в виртуальные машины

  Поддерживается вход в виртуальные машины через VNC.

- Быстрый поиск виртуальных машин

  Отображается список виртуальных машин по кластерам, что позволяет администраторам платформы быстро находить namespace виртуальной машины.

- Мониторинг и оповещения

  Осуществляется мониторинг и оповещения по использованию CPU, памяти, хранилища и сети виртуальных машин.

Virtual Machine Networking

- Использование контейнерных сетей

  Поддерживается настройка контейнерных сетей для виртуальных машин, включая режимы NAT и bridge.

- Настройка внутренней маршрутизации

Позволяет открывать виртуальные машины внутри или за пределами кластера путем создания объектов Service.

- Использование SR-IOV

  Поддерживается SR-IOV (Single Root I/O Virtualization) для высокопроизводительных сетевых интерфейсов.

Backup and Recovery

- Снимки виртуальных машин

  Поддерживается создание снимков виртуальных машин и восстановление из них.

Установка

Для того чтобы сотрудники проекта могли полноценно использовать функции виртуализации в контейнерной платформе, администратору платформы необходимо выполнить следующие операции для подготовки среды виртуализации.

Содержание

Предварительные требования

Процедура

Включение виртуализации на узле

Порядок действий

Развёртывание оператора

Создание экземпляра HyperConverged

Настройка коэффициента overcommit виртуальных машин (необязательно)

Важные замечания

Объяснение квоты ресурсов

Предварительные требования

- **Скачать** пакет установки **ACP Virtualization with KubeVirt**, соответствующий архитектуре вашей платформы.
- **Загрузить** пакет установки **ACP Virtualization with KubeVirt** с помощью механизма Upload Packages.
- При использовании функций виртуализации необходимо заранее спланировать и подготовить сетевую и хранилищную инфраструктуру.

Примечание:

- Если требуется подключение к виртуальной машине напрямую по IP, кластер должен использовать сетевой режим Kube-OVN Underlay. Вы можете ознакомиться с лучшими практиками в разделе Preparing Kube-OVN Underlay Physical Network.
- Рекомендуется использовать TopoLVM совместно с Kubevirt, так как он обеспечивает производительность, близкую к аппаратному уровню. Если требования к производительности не высоки, можно использовать распределённое хранилище Ceph.

Storage Product	Описание
TopoLVM	**Преимущества**: Относительно лёгкий и обладает хорошей производительностью. **Недостатки**: Не поддерживает работу между узлами, низкая надёжность, отсутствие избыточности.
Ceph Distributed Storage	**Преимущества**: Работает между узлами, высокая доступность, наличие избыточности. **Недостатки**: Избыточные копии дисков снижают эффективность использования; производительность ниже.

- Если используется TopoLVM и настроено несколько дисков, убедитесь, что оставшаяся ёмкость на узлах с включённой виртуализацией способна покрыть суммарный объём всех дисков, иначе создание виртуальной машины завершится неудачей.
- Если используется распределённое хранилище Ceph, убедитесь, что сеть, в которой находится хранилище, и сеть, в которой находятся виртуальные машины, могут взаимодействовать друг с другом.

Процедура

1 Включение виртуализации на узле

Если узлы собственного кластера являются **физическими машинами**, вы можете управлять разрешением Kubernetes планировать Virtual Machine Instances (VMI) на этом узле, включая или отключая переключатель виртуализации узла.

- При включённом переключателе новые виртуальные машины могут планироваться на физическом узле; физические узлы Windows не поддерживают включение виртуализации.
- При отключённом переключателе новые виртуальные машины не будут планироваться на физическом узле, но это не влияет на уже запущенные на нём виртуальные машины.

Порядок действий

1. Перейдите в **Platform Management**.
2. В левой навигационной панели выберите **Cluster Management** > **Clusters**.
3. Нажмите на ***Self-Built Cluster Name***.
4. На вкладке **Nodes** нажмите ⋮ справа от узла, для которого хотите настроить переключатель виртуализации > **Enable Virtualization**.
5. Нажмите **Confirm**.

2 Развёртывание оператора

1. Войдите в систему, перейдите на страницу **Platform Management**.
2. Нажмите **Marketplace** > **OperatorHub** для перехода на страницу **OperatorHub**.
3. Найдите **ACP Virtualization with KubeVirt**, нажмите **Install** и перейдите на страницу **Install ACP Virtualization with KubeVirt**.

   Параметры конфигурации:

Параметр	Рекомендуемая конфигурация
Channel	Канал по умолчанию — `alpha`.

Параметр	Рекомендуемая конфигурация
Installation Mode	`Cluster` : Все пространства имён в кластере используют один экземпляр оператора для создания и управления, что снижает использование ресурсов.
Installation Place	Выберите `Recommended` , Namespace поддерживает только **kubevirt**.
Upgrade Strategy	`Manual` : При появлении новой версии в Operator Hub требуется ручное подтверждение для обновления оператора до последней версии.

3 Создание экземпляра HyperConverged

1. Перейдите в **Platform Management**.
2. Нажмите **Marketplace** > **OperatorHub**.
3. Найдите **ACP Virtualization with KubeVirt**, нажмите на него для перехода на страницу с подробной информацией.
4. Нажмите **All Instances**.
5. На карточке экземпляра **HyperConverged** нажмите **Create Instance**.

   Примечание: В каждом кластере необходимо создать только один экземпляр **HyperConverged**.
6. Переключитесь в режим просмотра YAML и замените только ***placeholder*** в поле `spec.storageImport.insecureRegistries` в примере на правильный **адрес репозитория образов виртуальных машин**, например: `192.168.16.214:60080` , остальные параметры оставьте по умолчанию.

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

Результат замены:

```
spec:
  storageImport:
    insecureRegistries:
    - "192.168.16.214:60080"
```

7. Нажмите **Create** и дождитесь автоматического создания экземпляров типов CDI и KubeVirt в списке ресурсов, при этом убедитесь, что в YAML поле **status.phase** отображается как `deployed`, что означает успешное создание экземпляра HyperConverged.

4 Настройка коэффициента overcommit виртуальных машин (необязательно)

- Настройка коэффициента overcommit для кластера, в котором находятся виртуальные машины, в разделе **Cluster Management** > **Clusters**.
- Или настройка коэффициента overcommit для пространства имён, в котором расположены виртуальные машины, в разделе **Project Management** > **Namespaces**.

Важные замечания

- Виртуальные машины поддерживают только коэффициент overcommit по CPU, рекомендуемое значение — от 2 до 4.
- После включения коэффициента overcommit для виртуальных машин при создании виртуальной машины значение запроса контейнера (requests) фиксируется как **указанное значение лимита (limits) / коэффициент overcommit виртуальной машины**, что делает пользовательские настройки запроса через YAML неэффективными.

  Например: если коэффициент overcommit CPU для виртуальной машины установлен в 4, а пользователь при создании виртуальной машины указывает лимит CPU равный 4c, то значение запроса CPU будет 4c/4 = 1c.

Объяснение квоты ресурсов

Квота по памяти для виртуальных машин ограничивается квотой памяти пространства имён, в котором они находятся. Поскольку память Pod, хостящего виртуальную машину, обычно больше фактически доступной памяти виртуальной машины, рекомендуется резервировать 20% ресурсов. Если оставшиеся доступные ресурсы в пространстве имён опускаются ниже 20%, необходимо своевременно масштабировать ресурсы.

Menu

Образы

Введение

Введение

Преимущества

Руководства

Добавление образов виртуальных машин

Процедура

Обновление/Удаление образов виртуальных машин

Обновление/удаление учетных данных образа

Как сделать

Создание образов Windows на основе ISO с использованием KubeVirt

Предварительные требования

Ограничения и условия

Процедура

Удаленный доступ

Создание образов Linux на основе ISO с использованием KubeVirt

Предварительные требования

Ограничения и особенности

Процедура

Экспорт образов виртуальных машин

Процедура

Разрешения

Разрешения

Введение

Виртуализация Alauda Container Platform с помощью KubeVirt использует расширенные возможности API Kubernetes для абстрагирования образов виртуальных машин в виде **Custom Resource Definition** (CRD). Она предоставляет пользовательский интерфейс (UI), который позволяет пользователям легко импортировать образы виртуальных машин, хранящиеся в удалённых репозиториях, в ACP для дальнейшего использования.

Содержание

Преимущества

Преимущества

- Поддержка основных операционных систем

  Поддерживает различные широко используемые дистрибутивы Linux и операционные системы Windows.

- Поддержка нескольких архитектур

  Совместима как с архитектурами **X86_64**, так и **ARM64**.

- Поддержка нескольких источников

  Позволяет импортировать образы виртуальных машин из:

  - реестров образов
  - файловых серверов
  - объектного хранилища, совместимого с S3

- Поддержка нескольких форматов

  Поддерживает образы виртуальных машин в форматах **QCOW2** и **RAW**.

Menu

Руководства

Добавление образов виртуальных машин

Процедура

Обновление/Удаление образов виртуальных машин

Обновление/удаление учетных данных образа

Добавление образов виртуальных машин

Платформа поддерживает добавление образов виртуальных машин архитектур **X86_64** и **ARM64 (Alpha)**, что позволяет разработчикам быстро создавать виртуальные машины для существующих сервисов и облегчает миграцию бизнес-систем.

Содержание

Процедура

Процедура

1. Перейдите в **Platform Management**.
2. В левой навигационной панели нажмите **Virtualization Management** > **Virtual Machine Images**.
3. Нажмите **Add Virtual Machine Image**.
4. Следуйте инструкциям ниже для настройки соответствующих параметров.

Параметр	Описание
Provisioning Method	В настоящее время поддерживается только метод **Public Image**, то есть добавленный образ может использоваться в назначенных проектах.
Operating System	Поддерживаемые операционные системы: **CentOS/Ubuntu/RedHat/Debian/TLinux/Other Linux/Windows (Alpha)**.

Параметр	Описание
	Поддерживаемые архитектуры систем: **X86_64** и **ARM64 (Alpha)**.
Source	• **Image Repository**: образы виртуальных машин, хранящиеся в репозитории контейнерных образов. • **HTTP**: образы виртуальных машин, хранящиеся на файловом сервере с использованием протокола HTTP. • **Object Storage (S3)**: образы виртуальных машин, доступные через протокол Object Storage (S3). Если аутентификация не требуется, рекомендуется использовать HTTP в качестве источника.
CPU Architecture	Укажите архитектуру CPU. Для источников из репозитория образов поддерживается множественный выбор; для остальных источников — только одиночный выбор.
Image Address	Поддерживаются образы виртуальных машин KVM, включая форматы qcow2/raw. • Если источник — репозиторий образов, введите `repository_address:image_version`, например, `index.docker.io/library/ubuntu:latest`. • Если источник — HTTP, введите URL файла образа, который должен начинаться с `http://` или `https://`, например, `http://192.168.0.1/vm_image/centos_7.8.qcow2`. • Если источник — Object Storage (S3), введите адрес образа, доступный через протокол Object Storage (S3), например, `https://endpoint/bucket/centos.qcow2`.
Authentication	В зависимости от необходимости аутентификации в репозитории образов, переключатель можно включить или выключить. Если включено, можно выбрать из существующих учетных данных образа или нажать **Add Credentials**, поддерживаются только учетные данные типа

Параметр	Описание
	Username/Password. **Примечание**: при источнике Object Storage (S3) отключить аутентификацию нельзя.
Assigned Project	Назначьте права использования этого образа проектам. • All Projects: назначить права использования образа всем проектам. • Specific Project: назначить права использования образа конкретному проекту. • No Assignment: пока не назначать ни одному проекту. После создания образа можно назначить через операцию **Update Image**.

5. Нажмите **Add**.

Обновление/Удаление образов виртуальных машин

1. Перейдите в **Platform Management**.
2. В левой боковой панели нажмите **Virtualization Management** > **Virtual Machine Images**.
3. Нажмите ⋮ > **Update/Delete**.
4. После подтверждения нажмите **Update/Delete**.

Menu

Обновление/удаление учетных данных образа

1. Перейдите в **Platform Management**.
2. В левой навигационной панели нажмите **Virtualization Management** > **Virtual Machine Images**.
3. На вкладке **Image Credentials** нажмите ⋮ > **Update/Delete**.
4. После подтверждения нажмите **Update/Delete**.

Menu

Как сделать

Создание образов Windows на основе ISO с использованием KubeVirt

Предварительные требования

Ограничения и условия

Процедура

Удаленный доступ

Создание образов Linux на основе ISO с использованием KubeVirt

Предварительные требования

Ограничения и особенности

Процедура

Экспорт образов виртуальных машин

Процедура

Создание образов Windows на основе ISO с использованием KubeVirt

В данном документе рассматривается решение виртуальной машины на базе открытого компонента KubeVirt, использующее технологию виртуализации KubeVirt для создания образа операционной системы Windows через ISO-образ.

Содержание

Предварительные требования

Ограничения и условия

Процедура

- Создание образа
- Создание виртуальной машины
- Установка операционной системы Windows
- Установка virtio-win-tools
- Экспорт пользовательского образа Windows
- Использование образа Windows
- Добавление внутреннего маршрута

Удаленный доступ

Предварительные требования

- Все компоненты в кластере работают корректно.
- Пожалуйста, заранее подготовьте образ Windows и последние virtio-win-tools↗.

- Подготовьте репозиторий для хранения образа. В данном документе в качестве примера используется репозиторий build-harbor.example.cn, замените его в соответствии с вашей реальной средой.

Ограничения и условия

- При запуске KubeVirt размер файловой системы пользовательского образа влияет на скорость записи образа на диск в PVC. Если файловая система слишком велика, это может привести к увеличению времени создания.
- Рекомендуется держать корневой раздел Linux или диск C Windows менее 100 ГБ для минимизации начального размера. Последующее расширение можно выполнить через cloud-init (для Windows систем расширение необходимо делать вручную после создания).

Процедура

1 Создание образа

Создайте Docker-образ из подготовленных ISO-образов Windows и virtio-win и отправьте его в репозиторий. В данном документе в качестве примера используется Windows Server 2019.

Создание Docker-образа из ISO Windows

1. Перейдите в каталог, где хранится ISO-образ, и выполните в терминале команду для переименования ISO-образа в win.iso.

```
mv <ISO image name> win.iso # Замените <ISO image name> на фактическое имя образа, например, mv en_windows_server_2019_x64_dvd_4cb967d8.iso win.iso
```

2. Выполните команду для создания Dockerfile.

```
touch Dockerfile
```

3. Отредактируйте Dockerfile, добавьте следующий контент и сохраните файл.

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

4. Выполните команду для сборки Docker-образа.

```
docker build -t build-harbor.example.cn/3rdparty/vmdisks/winiso:2019 . #
Замените репозиторий в соответствии с вашей средой
```

5. Выполните команду для отправки образа в репозиторий.

```
docker push  build-harbor.example.cn/3rdparty/vmdisks/winiso:2019 # Замените
репозиторий в соответствии с вашей средой
```

Создание Docker-образа из ISO virtio-win

1. Перейдите в каталог, где хранится ISO-образ, и выполните команду для создания Dockerfile.

```
touch Dockerfile
```

2. Отредактируйте Dockerfile, добавьте следующий контент и сохраните файл.

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

3. Выполните команду для сборки Docker-образа.

```
docker build -t build-harbor.example.cn/3rdparty/vmdisks/win-virtio:latest . #
Замените репозиторий в соответствии с вашей средой
```

4. Выполните команду для отправки образа в репозиторий.

```
docker push  build-harbor.example.cn/3rdparty/vmdisks/win-virtio:latest # Замените репозиторий в соответствии с вашей средой
```

2 Создание виртуальной машины

1. Зайдите в **Container Platform**.
2. В левой навигационной панели выберите **Virtualization** > **Virtual Machines**.
3. Нажмите **Create Virtual Machine**.
4. Заполните необходимые параметры, такие как **Name**, **Image** и др. Для подробных параметров и конфигурации смотрите Create Virtual Machine.
5. Переключитесь в режим YAML.
6. Замените конфигурацию в поле spec.template.spec.domain.devices.disks следующим содержимым.

```
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. Добавьте следующее содержимое в поле spec.template.spec.volumes.

```
        - containerDisk:
            image: registry.example.cn:60070/3rdparty/vmdisks/winiso:2019 #
Замените образ в соответствии с вашей средой
          name: containerdisk
        - containerDisk:
            image: registry.example.cn:60070/3rdparty/vmdisks/win-virtio #
Замените образ в соответствии с вашей средой
          name: virtio
```

8. Проверьте YAML-файл. Полный YAML после завершения конфигурации выглядит следующим образом.

```yaml
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$0rawYwaeu9jL6hBf3XP9lk6XXaMUS9/W6LPbWRinUoXujo39lP3l98VOcOObtr.LDoAv/ylm
          name: cloudinitdisk
        - containerDisk:
            image: registry.example.cn:60070/3rdparty/vmdisks/winiso:2019 # Заменит
соответствии с вашей средой
          name: containerdisk
          - containerDisk
             image: registry.example.cn:60070/3rdparty/vmdisks/win-virtio # Заменит
соответствии с вашей средой
           name: virtio
         - dataVolume:
            name: aa-test-rootfs
          name: rootfs
```

9. Нажмите **Create**.

10. Нажмите **Actions** > **VNC Login**.

11. Когда появится подсказка **press any key boot from CD or DVD**, нажмите любую клавишу для входа в программу установки Windows; если подсказка не отображается, нажмите **Send Remote Command** в левом верхнем углу страницы, затем выберите **Ctrl-Alt-Delete** из выпадающего меню для перезагрузки сервера.

    Примечание: Если в верхней части страницы с деталями виртуальной машины появится сообщение **The current virtual machine has configuration changes**

that require a restart to take effect, please restart, это сообщение можно игнорировать; перезагрузка не требуется.

3 Установка операционной системы Windows

1. Следуйте инструкциям установки после входа на страницу установки.

   Примечание: На этапе выбора раздела шина должна быть sata, чтобы диск корректно распознавался. Поэтому необходимо последовательно выбрать каждый раздел и нажать **Delete** для удаления всех разделов, чтобы система могла обработать их автоматически.

2. После настройки пароля администратора нажмите **Send Remote Command** в левом верхнем углу страницы, затем выберите **Ctrl-Alt-Delete** из выпадающего меню.

3. При появлении запроса **The Ctrl+Alt+Delete combination will restart the server, confirm to restart** нажмите **OK**.

4. Введите пароль для доступа к рабочему столу Windows; на этом установка операционной системы Windows завершена.

4 Установка virtio-win-tools

Этот инструмент содержит необходимые драйверы.

1. Откройте Проводник.
2. Дважды щелкните **CD Drive(E:) virtio-win-<version>**, запустите каталог **virtio-win-guest-tools** для входа на страницу установки и следуйте инструкциям. Часть <version> должна соответствовать фактической версии.
3. После завершения установки выключите систему Windows.

5 Экспорт пользовательского образа Windows

Пожалуйста, обратитесь к Export Virtual Machine Image для конкретных действий.

6 Использование образа Windows

1. Зайдите в **Container Platform**.
2. В левой навигационной панели выберите **Virtualization** > **Virtual Machines**.
3. Нажмите **Create Virtual Machine**.
4. Заполните необходимые параметры на странице формы. Для образа выберите экспортированный образ Windows. Для подробных параметров и конфигурации смотрите Create Virtual Machine.
5. (Опционально) Если используется более новая операционная система, например Windows 11, включите такие функции, как часы, UEFI, TPM и др. Переключитесь в YAML и замените исходный YAML следующим файлом.

```
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. Нажмите **Create**.

7 Добавление внутреннего маршрута

Настройте внутренний маршрут типа NodePort для открытия порта для подключения по удаленному рабочему столу.

1. Зайдите в **Container Platform**.
2. В левой навигационной панели выберите **Virtualization** > **Virtual Machines**.
3. В списке нажмите на имя виртуальной машины, созданной с образом Windows, чтобы перейти на страницу деталей.
4. Нажмите на иконку **Add** рядом с **Internal Route** в области **Login Information**.
5. Настройте параметры согласно следующим указаниям.

Параметр	Описание
Type	Выберите **NodePort**.

Параметр	Описание
Port	• Протокол: выберите TCP. • Service Port: используйте 3389. • Virtual Machine Port: используйте 3389. • Service Port Name: используйте rdp.

6. Нажмите **OK** для возврата на страницу деталей.
7. Нажмите на ссылку **Internal Route** в области **Login Information**.
8. Сохраните информацию **Virtual IP** из области базовой информации и **Host Port** из области портов.

Удаленный доступ

В данном документе в качестве примера рассматривается удаленное подключение к операционной системе Windows. Для других ОС можно использовать программное обеспечение, поддерживающее протокол RDP.

1. Откройте **Remote Desktop Connection**.
2. Введите сохранённые Virtual IP и Host Port из шага **Add Internal Route** в формате **Virtual IP:Host Port**, например: 192.1.1.1:3389 .
3. Нажмите **Connect**.

Создание образов Linux на основе ISO с использованием KubeVirt

В данном документе описывается решение виртуальной машины, реализованное на базе открытого компонента KubeVirt. Оно использует технологию виртуализации KubeVirt для создания образа операционной системы Linux из ISO-образа.

Содержание

Предварительные требования

Ограничения и особенности

Процедура

Конвертация ISO-образа Linux в Docker-образ

Создание виртуальной машины

Установка операционной системы Linux

Изменение YAML-файла

Установка необходимого ПО и изменение конфигурации

Экспорт и использование пользовательского образа Linux

Предварительные требования

- Все компоненты в кластере работают корректно.
- Необходимо заранее подготовить образ Linux. В данном документе в качестве примера используется операционная система Ubuntu↗.

- Необходимо заранее подготовить репозиторий для хранения образов. В данном документе используется репозиторий build-harbor.example.cn в качестве примера; замените его согласно вашей реальной среде.

Ограничения и особенности

- При запуске KubeVirt размер файловой системы пользовательского образа влияет на скорость записи образа на диск PVC. Если файловая система слишком большая, это может привести к длительному времени создания.
- Рекомендуется держать размер корневого раздела Linux менее 100G для минимизации начального размера. После настройки cloud-init при создании виртуальной машины выделяйте больший объем хранилища для корневого раздела, и система автоматически его расширит.

Процедура

1 Конвертация ISO-образа Linux в Docker-образ

1. Перейдите в каталог, где хранится ISO-образ, и выполните в терминале команду для переименования ISO-образа в ubuntu.iso.

```
mv <ISO image name> ubuntu.iso # Замените <ISO image name> на фактическое имя образа, например, mv ubuntu-24.04-live-server-amd64.iso ubuntu.iso
```

2. Создайте Dockerfile, выполнив следующую команду.

```
touch Dockerfile
```

3. Отредактируйте Dockerfile, добавьте следующий контент и сохраните файл.

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

4. Постройте Docker-образ, выполнив следующую команду.

```
docker build -t build-harbor.example.cn/3rdparty/vmdisks/ubuntu-iso:24.04 . #
Замените репозиторий согласно вашей реальной среде
```

5. Отправьте образ в репозиторий, выполнив следующую команду.

```
docker push build-harbor.example.cn/3rdparty/vmdisks/ubuntu-iso:24.04 #
Замените репозиторий согласно вашей реальной среде
```

2 Создание виртуальной машины

1. Войдите в **Container Platform**.
2. В левом навигационном меню выберите **Virtualization** > **Virtual Machines**.
3. Нажмите **Create Virtual Machine**.
4. Заполните параметры на странице формы следующим образом. Для конкретных параметров и настроек смотрите Create Virtual Machine.

Параметр	Описание
Select Image	Выберите шаблонный образ для виртуальной машины.
IP Address	Оставьте по умолчанию, IP будет получен через **DHCP**.
Network Mode	Используйте режим **NAT**; не используйте режим **bridged**.

5. Переключитесь в режим YAML.
6. Замените конфигурацию в поле spec.template.spec.domain.devices.disks следующим содержимым.

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

7. Добавьте следующее содержимое в поле spec.template.spec.volumes.

```
        - containerDisk:
            image: registry.example.cn:60070/3rdparty/vmdisks/ubuntu-iso:24.04 #
Замените образ согласно вашей реальной среде
          name: containerdisk
```

8. Проверьте YAML-файл; полный YAML после завершения выглядит следующим образом.

```yaml
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 # Замените образ согласно вашей реальной среде
          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. Нажмите **Create**.

10. Нажмите **Actions** > **VNC Login**.

11. При появлении сообщения **press any key boot from CD or DVD** нажмите любую клавишу, чтобы войти в программу установки Windows; если сообщение не отображается, нажмите **Send Remote Command** в левом верхнем углу страницы, затем выберите **Ctrl-Alt-Delete** из выпадающего меню для перезагрузки сервера.

    Примечание: Если в верхней части страницы с деталями виртуальной машины появляется сообщение **Current virtual machine has configuration changes that require a restart to take effect. Please restart.**, его можно игнорировать; перезагрузка не требуется.

3 Установка операционной системы Linux

1. После входа на страницу установки следуйте инструкциям мастера установки. В данном документе приведен пример установки операционной системы Ubuntu; параметры конфигурации в процессе установки разных ОС обычно схожи, поэтому подробности не приводятся. Ниже приведены некоторые пояснения по конфигурации.

Конфигурация	**Описание**
Тип установки	Рекомендуется использовать минимальную установку для минимизации размера образа.
Конфигурация хранилища	Выберите пользовательское хранилище. Отформатируйте диск в формате ext4 или xfs и смонтируйте в корневой раздел (/). **Примечание**: Не используйте LVM для разметки диска (Create volume group (LVM)).
Конфигурация SSH	Выберите установку OpenSSH для доступа по SSH.

2. Дождитесь завершения установки.

4 Изменение YAML-файла

1. Войдите в **Container Platform**.
2. В левом навигационном меню выберите **Virtualization** > **Virtual Machines**.
3. Нажмите на **Имя виртуальной машины** в списке, чтобы перейти на страницу деталей.
4. Нажмите **Stop**.
5. В правом верхнем углу нажмите **Actions** > **Update**.
6. Переключитесь в режим YAML.
7. Убедитесь, что диск с именем **rootfs** в spec.template.spec.domain.devices.disks имеет bootOrder равный 1. Если нет, измените на 1.
8. Удалите соответствующий блок для диска с именем **containerdisk** в spec.template.spec.domain.devices.disks; конкретно удалите следующий контент.

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

9. Удалите соответствующий блок для диска с именем **containerdisk** в spec.template.spec.volumes; конкретно удалите следующий контент.

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

10. Нажмите **Update**.

11. Нажмите **Start**.

5 Установка необходимого ПО и изменение конфигурации

Примечание: Следующие команды и файлы конфигурации могут незначительно отличаться в разных операционных системах; корректируйте их согласно вашей реальной среде.

1. Войдите в операционную систему, используя имя пользователя и пароль.

2. Переключитесь на пользователя root.

3. Установите необходимые пакеты.

   - Для серии CentOS выполните команду:

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

   - Для серии Debian выполните команду:

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

4. Отредактируйте конфигурационный файл SSHD.

1. Выполните команду для редактирования sshd_config.

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

2. Добавьте следующие настройки.

```
PermitRootLogin yes  # Разрешить вход root-пользователю с паролем
PubkeyAuthentication yes  # Разрешить вход по ключу
```

3. Сохраните изменения.

5. Выполните команду для удаления пароля root-пользователя.

```
passwd -d root
```

6. Измените файл источников пакетов.

1. Выполните команду для редактирования файла источников и замените адрес на подходящий зеркальный сайт.

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

2. Сохраните изменения.

7. Измените конфигурацию cloud-init для автоматического расширения корневого раздела.

1. Выполните команду для редактирования файла cloud.cfg.

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

2. Добавьте следующий контент.

```
runcmd:
  - [growpart, /dev/vda, 1]  # Команда growpart расширяет раздел на диске, расширяя раздел /dev/vda1.
  - [xfs_growfs, /dev/vda1]  # Команда xfs_growfs расширяет файловую систему XFS, чтобы занять все доступное пространство раздела. /dev/vda1 — раздел с расширяемой файловой системой. После расширения раздела xfs_growfs гарантирует расширение файловой системы до нового размера раздела.
```

3. Сохраните изменения.

8. После завершения конфигурации выключите операционную систему.

6 Экспорт и использование пользовательского образа Linux

Для конкретных операций смотрите Export Virtual Machine Image.

Экспорт образов виртуальных машин

Эта функция используется для экспорта системного образа виртуальной машины и загрузки его в объектное хранилище, что позволяет добавлять файлы из объектного хранилища в качестве источников для образов виртуальных машин платформы.

Содержание

Процедура

- Остановка виртуальной машины
- Создание ресурса vmexport
- Загрузка файла образа виртуальной машины
- Загрузка файла образа виртуальной машины в объектное хранилище
- Создание образа виртуальной машины

Процедура

Примечание: Все операции ниже необходимо выполнять на управляющем узле кластера, где находится виртуальная машина.

1 Остановка виртуальной машины

1. Перейдите в **Управление платформой**.
2. В левой навигационной панели нажмите **Управление виртуализацией** > **Виртуальные машины**.
3. Нажмите на имя виртуальной машины, системный образ которой необходимо экспортировать, после чего вы перейдёте на страницу с деталями виртуальной

машины в Container Platform.

4. Нажмите **Остановить**.

2 Создание ресурса vmexport

1. Откройте CLI инструмент.

2. Выполните следующую команду для установки переменных.

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

Объяснение параметров:

- **NAMESPACE**: имя namespace, в котором находится виртуальная машина; замените часть *<namespace>* на это имя.
- **VM_NAME**: имя виртуальной машины, системный образ которой нужно экспортировать; замените часть *<vm_name>* на это имя.
- **TTL_DURATION**: время жизни задачи экспорта, по умолчанию 2 часа, но при необходимости можно увеличить.

3. Выполните следующую команду для создания ресурса vmexport.

```
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
```

Если появится подобное сообщение, значит создание прошло успешно.

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

4. Выполните следующую команду для проверки статуса ресурса vmexport.

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

Вывод:

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

5. Когда в поле PHASE в выводе появится значение Ready, нажмите ctrl (control) + c для остановки операции watch.

6. Выполните следующую команду для получения TOKEN.

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

3 Загрузка файла образа виртуальной машины

1. Выполните следующую команду для получения IP-адреса Pod'a виртуального экспорта машины в указанном namespace и сохраните его в переменную окружения EXPORT_SERVER_IP.

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

2. Выполните следующую команду для установки переменной окружения URL, указывающей на файл дискового образа виртуальной машины.

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

3. Выполните следующую команду для загрузки файла образа, при этом загруженный файл будет называться disk.img.gz.

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

4 Загрузка файла образа виртуальной машины в объектное хранилище

Загрузите скачанный файл образа в объектное хранилище. Для загрузки можно использовать любой S3-инструмент, в данном документе в качестве примера используется инструмент mc (minio-client).

1. Выполните следующую команду для настройки инструмента mc и подключения к указанному S3-сервису хранения.

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

   Объяснение параметров:

   - ENDPOINT: адрес S3-сервиса хранения; замените часть `<ENDPOINT>` на этот адрес.
   - ACCESSKEY, SECRETKEY: ak и sk пользователя S3-сервиса хранения, используемые для аутентификации; по связанным вопросам смотрите MinIO Object Storage ↗.

2. Выполните следующую команду для создания бакета для хранения файлов образов виртуальных машин.

```
mc mb minio/vmdisks
```

3. Выполните следующую команду для загрузки экспортированного файла образа виртуальной машины disk.img.gz в созданный бакет.

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

5 Создание образа виртуальной машины

1. Перейдите в **Управление платформой**.
2. В левой навигационной панели нажмите **Управление виртуализацией** > **Образы виртуальных машин**.
3. Нажмите **Добавить образ виртуальной машины**.
4. В поле адреса образа укажите <*ENDPOINT*>/vmdisks/disk.img.gz, заменив часть <ENDPOINT> на адрес S3-сервиса хранения. По другим параметрам смотрите Добавление образов виртуальных машин.
5. Нажмите **Добавить**.

Menu

Разрешения

Функция	Действие	Platform Administrator	Platform auditors	Project Manage
virtualmachineimagetemplates `acp-virtualmachineimagetemplates`	Просмотр	✓	✓	✓
	Создать	✓	×	×
	Обновить	✓	×	×
	Удалить	✓	×	×

Menu

Виртуальная машина

Введение

Введение

Преимущества

Руководства

Создание виртуальных машин/групп виртуальных машин

Предварительные требования

Примечания

Create Virtual Machine

Create Virtual Machine Group

Пакетные операции с виртуальными машинами

Процедура

Вход в виртуальную машину с использованием VNC

Процедура

Управление ключевыми парами

Создание ключевых пар

Обновление ключевых пар

Удаление ключевых пар

Управление виртуальными машинами

Сброс пароля

Обновление ключа

Обновление спецификаций

Живая миграция

Обновление конфигурации NAT-сети

Обновление тегов и аннотаций

Добавление сервиса

Переустановка операционной системы

Настройка IP

Мониторинг и оповещения

Мониторинг

Оповещения

Быстрый поиск виртуальных машин

Предварительные требования

Процедура

Как сделать

Настройка проброса USB-хоста

Обзор функции

Сценарии использования

Предварительные требования

Шаги

Результат выполнения

Узнайте больше

Горячая миграция виртуальной машины

Overview

Ограничения и условия

Предварительные условия

Шаги выполнения

Восстановление виртуальной машины

Шаги для выполнения

Клонирование виртуальной машины

Предварительные требования

Шаги для выполнения

Связанные операции

Подготовка среды для физического GPU Passthrough

Ограничения и лимитации

Предварительные требования

Пошаговые действия

Проверка результата

Связанные операции

Устранение неполадок

Миграция Pod виртуальных машин и восстановление после аварийного завершения работы узлов виртуальных машин

Описание проблемы

Анализ причины

Решения

Сообщения об ошибках горячей миграции и решения

Введение

KubeVirt предоставляет CRD (Custom Resource Definitions), такие как **VirtualMachine** и **VirtualMachineInstance**, для абстрагирования ресурсов виртуальных машин (VM). На основе этих CRD пользователи получают комплексные возможности управления виртуальными машинами. Опираясь на эту основу, **ACP Virtualization With KubeVirt** дополнительно повышает удобство использования, предлагая **Web Console**, которая позволяет пользователям выполнять различные операции с большей лёгкостью.

Содержание

Преимущества

Преимущества

- **Комплексное управление виртуальными машинами**
  - Создание/удаление VM, переустановка ОС, сброс паролей, обновление SSH-ключей, настройка выделенных ресурсов и доступ к VNC-консолям.
  - Экспорт/клонирование VM, выполнение live migration и многое другое.
- **Удобство использования**
  - Большинство операций с VM можно выполнить легко через интуитивно понятный Web UI.
- **Визуальный мониторинг и оповещения**
  - Мониторинг ключевых метрик VM (например, CPU, память, использование диска) через Web UI.

- Настройка оповещений для проактивного выявления потенциальных проблем во время работы (например, исчерпание ресурсов, снижение производительности).

Menu

Руководства

Создание виртуальных машин/групп виртуальных машин

Предварительные требования

Примечания

Create Virtual Machine

Create Virtual Machine Group

Пакетные операции с виртуальными машинами

Процедура

Вход в виртуальную машину с использованием VNC

Процедура

Управление ключевыми парами

Создание ключевых пар

Обновление ключевых пар

Удаление ключевых пар

Управление виртуальными машинами

Сброс пароля

Обновление ключа

Обновление спецификаций

Живая миграция

Обновление конфигурации NAT-сети

Обновление тегов и аннотаций

Добавление сервиса

Переустановка операционной системы

Настройка IP

Мониторинг и оповещения

Мониторинг

Оповещения

Быстрый поиск виртуальных машин

Предварительные требования

Процедура

Создание виртуальных машин/групп виртуальных машин

Создайте виртуальную машину (VirtualMachineInstance) с использованием образа и запланируйте виртуальную машину на физические узлы с установленными компонентами Kubevirt и включённой виртуализацией.

Вы можете создать одну виртуальную машину через Create Virtual Machine или быстро создать несколько виртуальных машин (VirtualMachineInstance) с одинаковой конфигурацией, используя Create Virtual Machine Group (virtualMachinePool).

Содержание

Предварительные требования

Примечания

Create Virtual Machine

Процедура

Связанные операции

Create Virtual Machine Group

Процедура

Предварительные требования

- Перед созданием виртуальной машины с использованием образа, пожалуйста, подтвердите у администратора платформы следующее:
  - Целевой кластер является самосозданным кластером, и компоненты Kubevirt развернуты.

- Целевой узел должен быть физическим узлом с включённой виртуализацией.
- Образ виртуальной машины добавлен на платформу.

- Если необходимо использовать функцию passthrough физического GPU виртуальной машины, обратитесь к администратору платформы для следующей настройки:
  1. Получите план подготовки среды passthrough GPU и подготовьте необходимую среду.
  2. Подготовьте требуемый физический GPU и включите соответствующие функции для passthrough физического GPU виртуальной машины.

Примечания

При использовании виртуальных машин Windows поддерживается только вход по **имени пользователя/паролю**, заданным в образе виртуальной машины. Пожалуйста, заранее свяжитесь с администратором платформы для получения этой информации.

Create Virtual Machine

Процедура

Примечание: Ниже приведён пример создания виртуальной машины с использованием формы, также вы можете переключиться на формат YAML для выполнения операции.

1. Войдите в **Container Platform**.
2. В левой навигационной панели нажмите **Virtualization** > **Virtual Machines**.
3. Нажмите **Create Virtual Machine**.
4. В области **Basic Information** заполните имя и отображаемое имя виртуальной машины, а также задайте теги или аннотации.

Параметр	Описание
Tags	Используются для выбора объектов и поиска коллекций объектов, соответствующих определённым критериям. Должны быть парой ключ-значение, например: *app.kubernetes.io/name: hello-app*.
Annotations	Используются для предоставления любой информации командам разработки и эксплуатации. Должны быть парой ключ-значение, например: *cpaas.io/maintainer: kim*.

5. Установите тип машины и выберите образ виртуальной машины.

Параметр	Описание
Specifications	Вы можете выбрать рекомендованные сценарии использования или задать собственные ограничения ресурсов в зависимости от ваших потребностей.
Physical GPU (Alpha)	Выберите модель физического GPU; каждому виртуальному компьютеру может быть выделен только один физический GPU. **Примечание**: Passthrough физического GPU для виртуальной машины означает прямое выделение реального графического процессора (GPU) виртуальной машине в виртуализированной среде, что позволяет ей напрямую обращаться к физическому GPU и использовать его для достижения графической производительности, эквивалентной работе на физической машине, избегая узких мест производительности, вызванных виртуальными графическими адаптерами, и повышая общую производительность.
Image	Выберите публичный образ, назначенный проекту платформы администратором платформы. **Примечание**: Поддерживается только выбор образов с той же **CPU архитектурой**, что и архитектура кластера.

6. В области **Storage** настройте соответствующую информацию согласно следующим инструкциям.

Параметр	Описание
Disk Name	Имя диска хранения; имя системного диска изменить нельзя.
Type	• **Root Disk**: Система автоматически создаёт системный диск rootfs типа VirtIO для хранения операционной системы и данных. • **Data Disk**: Нажмите для добавления нескольких дисков данных для постоянного хранения данных. По умолчанию устройство VirtIO. **Примечание**: Имена дисков данных не должны дублировать существующие имена дисков.
Volume Mode	• **File System**: Монтировать диск как смонтированный файловый каталог. • **Block Device**: Монтировать диск как блочное устройство.
Storage Class	Платформа поддерживает диски виртуальных машин, создавая и управляя persistent volume claims. Необходимо указать storage class, требуемый для динамического создания persistent volume claims. Разные storage class поддерживают разные volume mode; если для выбранного volume mode нет доступного storage class, обратитесь к администратору для добавления.
Capacity	Требуемая ёмкость для хранения виртуальной машины; минимум для системного диска — 20 ГБ.
Delete with VM	По умолчанию включено и не может быть изменено, что означает удаление данных диска при удалении виртуальной машины.

7. В области **Network** настройте соответствующую информацию согласно следующим инструкциям.

Параметр	Описание
IP Address	• По умолчанию **Dynamic (DHCP)**; IP-адрес динамически назначается при запуске виртуальной машины и освобождается при её остановке. • Если привязан **Static IP**, виртуальная машина всегда будет использовать этот IP-адрес даже после перезапуска. Если в текущем проекте нет доступных IP, сначала освободите IP.
Network Mode	• **Bridged**: Виртуальная машина использует тот же IP-адрес, что и контейнерная группа, и взаимодействует с внешним миром через этот IP. • **NAT**: Виртуальной машине назначается внутренний IP, который транслируется в IP контейнерной группы для внешнего взаимодействия. Открытые порты указывают порты виртуальной машины, например порт SSH 22; если **Open Ports** не заполнены, считаются открытыми все порты.
Auxiliary Network Card	Добавляйте вспомогательные сетевые карты по необходимости. **Примечание**: • Если требуется функция вспомогательных сетевых карт или нет доступных типов вспомогательных сетей, обратитесь к администратору платформы для настройки. • Типы SR-IOV поддерживаются только для Linux на архитектуре x86_64. • По умолчанию IP-адреса получаются через DHCP. • После нескольких перезагрузок SR-IOV виртуальные машины могут иметь два разных VF с одинаковым MAC-адресом.

8. В области **Initialization Settings** настройте соответствующую информацию согласно следующим инструкциям.

Параметр	Описание
Keys	Всегда используйте SSH-ключи для проверки удалённого входа. Этот метод не требует проверки пароля; рекомендуется входить в виртуальную машину с помощью ключей. • Вы можете использовать уже имеющиеся ключи на платформе или создать новые; все ключи доступны на странице **Virtualization > Key Pairs**. • Доступ по SSH к виртуальной машине возможен только у лиц, имеющих приватный ключ. Если несколько человек обслуживают виртуальную машину, можно связать несколько ключей и распределить приватные ключи между пользователями. В случае утечки ключа соответствующий ключ можно быстро отозвать для минимизации ущерба. • Публичный ключ SSH хранится на платформе в конфиденциальном виде; приватный ключ не хранится, поэтому храните его в безопасности самостоятельно. • Пожалуйста, обратитесь к документации соответствующей операционной системы для пароля пользователя root.
Password	Используйте пользователя операционной системы и пароль для проверки входа, который впоследствии можно обновить на метод с ключами. • Пользователь — это только начальная учётная запись; после успешного создания виртуальной машины можно создавать других пользователей ОС для входа. • Платформа шифрует и хранит пароль пользователя root, и вы не увидите его в открытом виде, поэтому храните его в безопасности самостоятельно.
Start Immediately	По умолчанию включено. При включении виртуальная машина запускается сразу после создания, иначе создаётся только виртуальная машина без запуска.

9. (Необязательно) В области **Advanced Configuration** настройте соответствующую информацию согласно следующим инструкциям.

Параметр	Описание
Health Check	• **Liveness Check**: Проверяет, находится ли виртуальная машина в здоровом состоянии; при обнаружении аномалий определяется необходимость перезапуска экземпляра согласно настройкам проверки здоровья. • **Availability Check**: Проверяет, завершён ли запуск виртуальной машины и находится ли она в нормальном рабочем состоянии; при обнаружении аномалий состояние виртуальной машины обновляется. Для описания связанных параметров.
Node Affinity	• **Preferred**: Виртуальная машина будет по возможности запланирована на узлы, соответствующие требованиям affinity. Система определяет узлы, способные запустить виртуальную машину, комбинируя веса affinity и другие требования планирования (например, требования к вычислительным ресурсам). • **Required**: Виртуальная машина будет запланирована только на узлы, полностью соответствующие требованиям affinity.

10. После проверки правильности информации нажмите **Create**.

    Дождитесь, пока статус виртуальной машины изменится с **Creating** на **Running**.

Связанные операции

Вы можете нажать на значок ⋮ справа на странице списка или выбрать **Actions** в правом верхнем углу страницы деталей для обновления или удаления виртуальной машины по необходимости. Для других связанных операций, таких как сброс пароля или обновление ключей, пожалуйста, обратитесь к Manage Virtual Machines.

Примечание:

- Обновление возможно только при статусах виртуальной машины **Abnormal**, **Unknown** или **Stopped**.
- Обновление не поддерживает отображение дисков, которые были отдельно присоединены или созданы после создания виртуальной машины.
- По умолчанию при обновлении опция **Start Immediately** отключена; вы можете включить её при необходимости.

Create Virtual Machine Group

Процедура

Примечание: Ниже приведён пример создания группы виртуальных машин с использованием формы, также вы можете переключиться на формат YAML для выполнения операции.

1. Войдите в **Container Platform**.
2. В левой навигационной панели нажмите **Virtualization** > **Virtual Machine Groups**.
3. Нажмите **Create Virtual Machine Group**.
4. В области **Basic Information** настройте информацию для группы виртуальных машин согласно следующим инструкциям.

Параметр	Описание
Number of Instances	Количество виртуальных машин, создаваемых группой виртуальных машин.
Anti-Affinity between Instances	Если включено, при планировании нескольких виртуальных машин на узлы будет предпринята попытка распределить виртуальные машины по разным узлам, что повышает высокую доступность группы виртуальных машин.

Параметр	Описание
Tags	Для группы виртуальных машин можно добавить теги. Теги используются для выбора объектов и поиска коллекций объектов, соответствующих определённым критериям. Должны быть парой ключ-значение, например: *app.kubernetes.io/name: hello-app*.

5. В области **Virtual Machine Template** настройте единые теги, аннотации, спецификации, образы, хранилище и другую информацию для всех виртуальных машин группы согласно Create Virtual Machine.

6. После проверки правильности информации нажмите **Create**.

   Совет: После успешного создания вы можете перейти на страницу списка **Virtual Machines**, чтобы просмотреть информацию о виртуальных машинах, созданных через группу виртуальных машин.

Пакетные операции с виртуальными машинами

Выполняйте пакетные операции, такие как запуск, остановка, перезапуск и удаление виртуальных машин.

Содержание

Процедура

Процедура

1. Перейдите в **Container Platform**.
2. В левой навигационной панели выберите **Virtualization** > **Virtual Machines**.
3. Найдите нужную виртуальную машину, нажмите ⋮ для выполнения операций с одной виртуальной машиной или воспользуйтесь изображением ниже для пакетных операций с виртуальными машинами.

   Примечание:

   - Операция **Start/Batch Start** доступна, когда виртуальная машина находится в состоянии приостановки или остановки; операция **Stop/Batch Stop** доступна, когда виртуальная машина находится в состоянии Preparing, Starting, Running, Suspended, Unknown или Exception; операция **Restart/Batch Restart** доступна, когда виртуальная машина находится в состоянии Running.
   - Принудительное выполнение операций **Restart/Stop** эквивалентно отключению питания виртуальной машины, что может привести к потере данных, не

записанных на диск.

4. Выполните операции согласно подсказкам на интерфейсе. Когда виртуальная машина перейдет в одно из указанных ниже состояний, операция считается успешной.

Операция	**Статус**
Запуск виртуальной машины	Running
Остановка виртуальной машины	Stopped
Перезапуск виртуальной машины	Running

Вход в виртуальную машину с использованием VNC

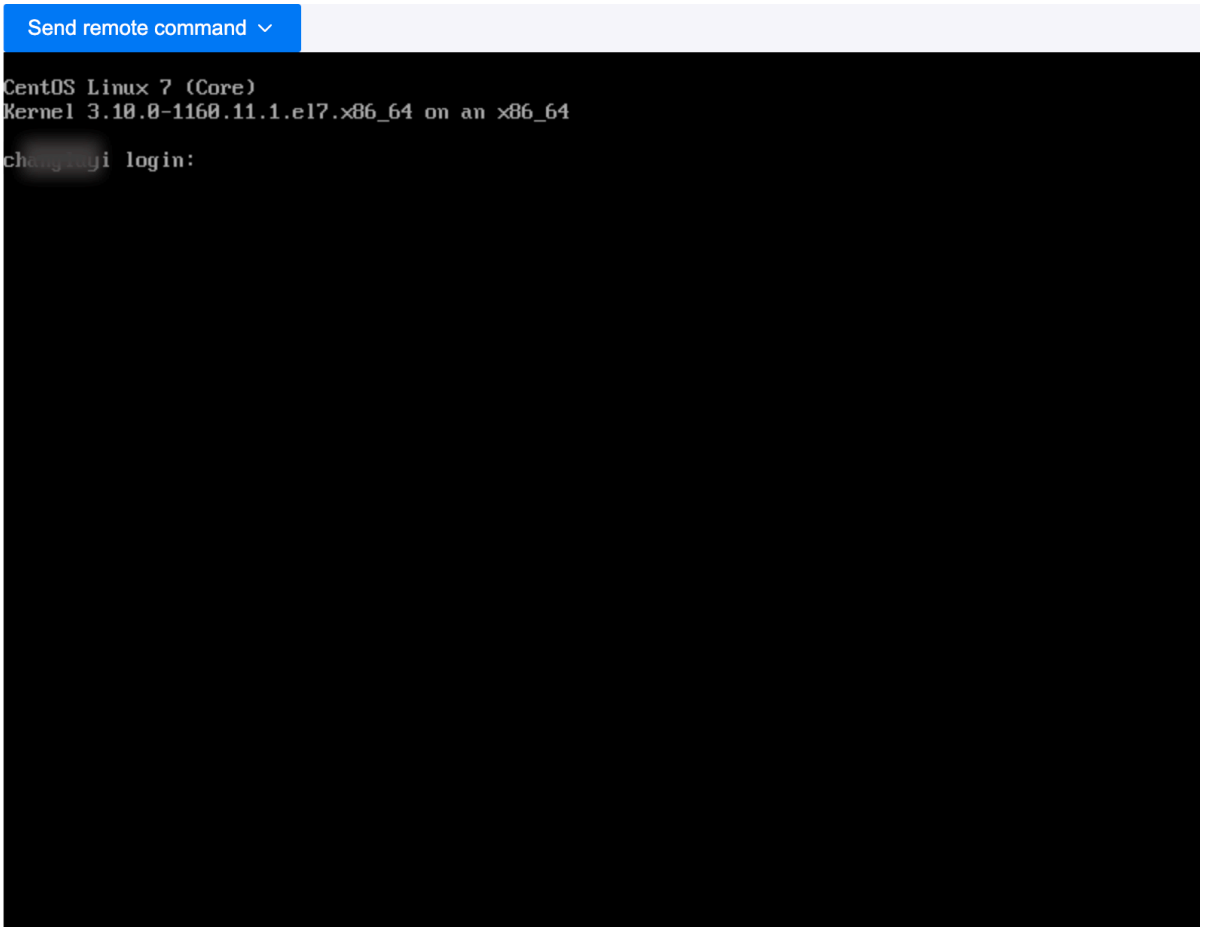
Выполните вход в виртуальную машину через Web Console (VNC) в качестве аварийного способа управления.

Содержание

Процедура

Процедура

1. Перейдите в **Container Platform**.
2. В левой навигационной панели нажмите **Virtualization** > **Virtual Machines**.
3. Нажмите ⋮ > **VNC Login**.
4. Окно консоли откроется автоматически; для входа необходимо ввести имя пользователя и пароль.



Примечание:

- Поддерживается отправка стандартных команд с клавиатуры.
- Поддерживается копирование и вставка команд и параметров.

Управление ключевыми парами

Создание, обновление или удаление ключевых пар.

Содержание

Создание ключевых пар

Обновление ключевых пар

Удаление ключевых пар

Создание ключевых пар

1. Перейдите в **Container Platform**.
2. В левой навигационной панели выберите **Virtualization** > **Key Pairs**.
3. Нажмите **Create Key Pair**.

   В настоящее время поддерживаются только ключевые пары типа SSH. Вы можете вручную импортировать ключи или позволить системе автоматически сгенерировать ключевую пару. При использовании системно сгенерированной ключевой пары платформа поддерживает автоматическую загрузку приватного ключа на ваш локальный компьютер. Платформа не сохраняет приватный ключ.
4. Нажмите **Create**.

Обновление ключевых пар

1. Перейдите в **Container Platform**.
2. В левой навигационной панели выберите **Virtualization** > **Key Pairs**.
3. Найдите ***Key Pair Name***, нажмите ⋮ > **Update**.
4. После повторного импорта или генерации новой ключевой пары системой нажмите **Update**.

Удаление ключевых пар

1. Перейдите в **Container Platform**.
2. В левой навигационной панели выберите **Virtualization** > **Key Pairs**.
3. Найдите ***Key Pair Name***, нажмите ⋮ > **Delete** и подтвердите.

Управление виртуальными машинами

Содержание

Сброс пароля
- Процедура

Обновление ключа
- Процедура

Обновление спецификаций

Живая миграция

Обновление конфигурации NAT-сети
- Процедура

Обновление тегов и аннотаций

Добавление сервиса

Переустановка операционной системы
- Процедура

Настройка IP
- Процедура

Сброс пароля

Сбросьте пароль пользователя **root**. Этот пароль также служит паролем для входа в виртуальную машину при использовании входа по паролю.

Процедура

1. Зайдите в **Container Platform**.

2. В левой навигационной панели нажмите **Virtualization** > **Virtual Machines**.
3. Найдите виртуальную машину и выберите ⋮ > **Reset Password**.
4. Установите пароль.
5. Нажмите **Reset**.

   Примечание: Пожалуйста, храните пароль в безопасности. Для обеспечения безопасности среды платформа шифрует и сохраняет ваш пароль, и вы не сможете увидеть пароль в открытом виде повторно.

Обновление ключа

Обновите SSH-ключи.

Процедура

1. Зайдите в **Container Platform**.
2. В левой навигационной панели нажмите **Virtualization** > **Virtual Machines**.
3. Найдите виртуальную машину и выберите ⋮ > **Update Key**.
4. Выберите один или несколько связанных ключей либо **Create Key**.
5. Выберите, нужно ли перезапускать сразу; для применения обновления ключей требуется перезапуск виртуальной машины.
6. Нажмите **Update**.

Обновление спецификаций

1. Зайдите в **Container Platform**.
2. В левой навигационной панели нажмите **Virtualization** > **Virtual Machines**.

3. Найдите нужную виртуальную машину и выберите ⋮ > **Update Specifications**.
4. Измените соответствующие ресурсы в соответствии с рекомендованными платформой сценариями или пользовательскими требованиями.
5. Выберите, нужно ли **Restart Immediately**; конфигурация вступит в силу после перезапуска.
6. Нажмите **Update**.

Живая миграция

Примечание: Если вам нужна документация по операциям живой миграции, обратитесь к администратору за помощью.

1. Зайдите в **Container Platform**.
2. В левой навигационной панели нажмите **Virtualization** > **Virtual Machines**.
3. Найдите нужную виртуальную машину и выберите ⋮ > **Live Migration**.
4. Нажмите **Confirm**.

Обновление конфигурации NAT-сети

При использовании режима NAT-сети платформа по умолчанию открывает порт 22 для SSH-сервисов, а также вы можете открыть другие порты по необходимости.

Процедура

1. Зайдите в **Container Platform**.
2. В левой навигационной панели нажмите **Virtualization** > **Virtual Machines**.
3. Нажмите на ***Virtual Machine Name***.

4. В разделе **Basic Information** нажмите на иконку справа от **Open Port**.
5. Введите номер порта и нажмите клавишу Enter для подтверждения.
6. Выберите, нужно ли **Restart Immediately**; конфигурация вступит в силу после перезапуска.
7. Нажмите **Update**.

Обновление тегов и аннотаций

1. Зайдите в **Container Platform**.
2. В левой навигационной панели нажмите **Virtualization** > **Virtual Machines**.
3. Нажмите на ***Virtual Machine Name***.
4. В разделе **Basic Information** нажмите на иконку справа от **Tags** или **Annotations**.
5. Настройте по необходимости и нажмите **Update**.

Добавление сервиса

1. Зайдите в **Container Platform**.
2. В левой навигационной панели нажмите **Virtualization** > **Virtual Machines**.
3. Нажмите на ***Virtual Machine Name***.
4. В разделе **Login Information** нажмите на иконку справа от Internal Route.
5. Ознакомьтесь со страницей Create Service для быстрого добавления внутренних маршрутов для виртуальной машины.
6. Нажмите **Confirm**.

Переустановка операционной системы

Настоятельно рекомендуется сделать резервную копию данных перед переустановкой операционной системы, чтобы избежать потери данных.

Примечание: Эта операция очистит все данные на **системном диске** виртуальной машины, а также все **снимки**, и является необратимой. Пожалуйста, будьте осторожны!

Процедура

1. Зайдите в **Container Platform**.
2. В левой навигационной панели нажмите **Virtualization** > **Virtual Machines**.
3. Найдите виртуальную машину и выберите ⋮ > **Reinstall Operating System**.
4. В окне **Reinstall Operating System** настройте следующие параметры.
   - **Provisioning Method**: в настоящее время поддерживаются публичные образы.
   - **Select Image**: по умолчанию для переустановки будет использоваться текущий образ операционной системы. Если вы хотите переустановить новую ОС, сначала выберите операционную систему образа виртуальной машины, затем выберите образ виртуальной машины, принадлежащий этой ОС.
5. Нажмите **Reinstall**.

Настройка IP

Назначьте виртуальной машине IP с помощью динамического выделения (DHCP) или привяжите фиксированный IP; новый IP вступит в силу после перезапуска виртуальной машины.

Процедура

1. Зайдите в **Container Platform**.

2. В левой навигационной панели нажмите **Virtualization** > **Virtual Machines**.

3. Найдите нужную виртуальную машину и выберите ⋮ > **Configure IP**.

4. Настройте **IP Address**.

   - Заполните доступный IP: привязка фиксированного IP означает, что даже после перезапуска виртуальная машина будет постоянно использовать этот IP-адрес.
   - Оставьте поле пустым: будет использоваться динамическое выделение (DHCP), при котором IP назначается при запуске виртуальной машины и освобождается при её остановке.

5. Выберите, нужно ли **Restart Immediately**; конфигурация вступит в силу после перезапуска.

6. Нажмите **Configure**.

Мониторинг и оповещения

Мониторинг и оповещения виртуальных машин по параметрам CPU, памяти, хранилища и сети. Для своевременного информирования также можно настроить политики уведомлений.

Интуитивно представленные данные мониторинга могут использоваться для поддержки принятия решений при проверке работы или настройке производительности, а комплексный механизм оповещений и уведомлений поможет обеспечить стабильную работу виртуальных машин.

Содержание

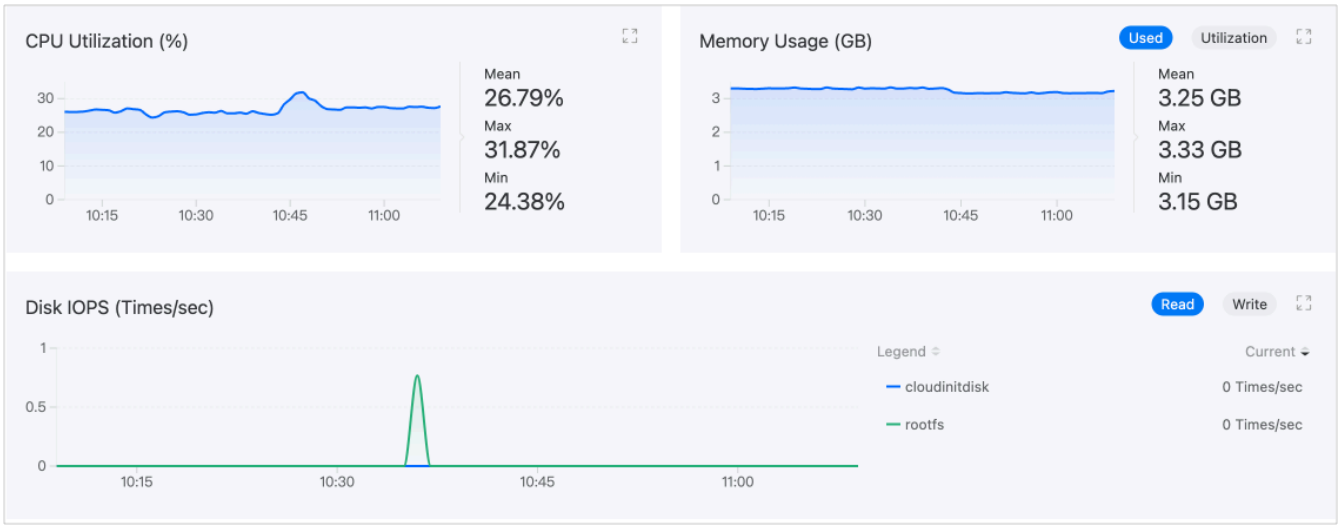
Мониторинг

Оповещения

- Настройка политик оповещений
- Обработка оповещений
- Привязка политик уведомлений

Мониторинг

По умолчанию платформа собирает часто используемые метрики производительности виртуальных машин, включая CPU, память, хранилище и сеть. Перейдите в **Virtualization** > **Virtual Machines**, и на вкладке **Monitoring** в деталях виртуальной машины вы сможете просмотреть данные мониторинга в реальном времени по этим метрикам.



Оповещения

Настройка политик оповещений

Для включения оповещений необходимо сначала создать политику оповещений. Политика оповещений описывает объекты, которые вы хотите контролировать, условия, при которых необходимо отправлять оповещения, и способ уведомления о соответствующих оповещениях. Перейдите в **Container Platform** > **Virtualization** > **Virtual Machines**, и в деталях виртуальной машины на вкладке **Alerts** нажмите **Create Alert Policy** для завершения настройки.

Параметр	Описание
Тип оповещения	- Metric Alert: Объект мониторинга — предопределённая метрика платформы, например *Memory Usage Rate*. - Event Alert: Объект мониторинга — причина события, то есть причина перехода виртуальной машины в текущее состояние, например BackOff, Pulling, Failed.
Условие срабатывания	Состоит из операторов сравнения, порогов оповещения и длительности. Путём сравнения результатов мониторинга в реальном времени с установленными порогами определяется необходимость оповещения. Если задана длительность, платформа также сравнивает

Параметр	Описание
	время, в течение которого объект мониторинга находился в состоянии оповещения.
Уровень оповещения	- Hint: У объекта мониторинга есть ожидаемые проблемы, которые не влияют немедленно на бизнес-процессы, но представляют потенциальные риски. Например, если использование CPU превышает 70% в течение 3 минут. - Warning: У объекта мониторинга есть операционные риски, которые при отсутствии своевременного реагирования могут повлиять на нормальную работу бизнеса. Например, если использование CPU превышает 80% в течение 3 минут. - Serious: У объекта мониторинга выявлены известные проблемы, которые могут привести к сбоям в работе платформы, влияющим на нормальное функционирование бизнеса. - Disaster: Объект мониторинга вышел из строя, что привело к прерыванию сервисов платформы, потере данных и значительному влиянию.

Совет: Функция оповещений виртуальной машины схожа с общей функцией оповещений платформы. Для более подробных инструкций по настройке обратитесь к общей документации по Alerts.

Обработка оповещений

Перейдите на вкладку **Alerts**, и если указаны стратегии состояния оповещений, пожалуйста, оперативно примите меры.

Привязка политик уведомлений

Помимо оповещений в реальном времени на вкладке **Alerts**, платформа также поддерживает отправку информации об оповещениях по электронной почте, SMS и другим способам соответствующим сотрудникам, уведомляя их о необходимости принять меры для устранения проблем или предотвращения сбоев. Политика уведомлений настраивается через администратора.

Быстрый поиск виртуальных машин

Платформа поддерживает отображение списка виртуальных машин по кластерам, что позволяет администраторам платформы быстро находить namespace виртуальной машины и выполнять операции, такие как масштабирование или устранение неполадок, тем самым повышая эффективность работы.

Содержание

Предварительные требования

Процедура

Предварительные требования

Убедитесь, что функция виртуализации включена для текущего кластера перед использованием. Пожалуйста, обратитесь к разделу Install.

Процедура

1. Перейдите в **Platform Management**.
2. В левой навигационной панели нажмите **Virtualization Management** > **Virtual Machines**.
3. Выберите ***Cluster***, чтобы просмотреть список виртуальных машин в этом кластере.

4. Вы можете быстро найти виртуальную машину по её имени, IP-адресу или создателю.

5. Нажмите на ссылку с ***Name*** виртуальной машины, чтобы перейти на страницу с подробной информацией о ней, где можно выполнить операции, такие как масштабирование или устранение неполадок.

Menu

Как сделать

Настройка проброса USB-хоста

Обзор функции

Сценарии использования

Предварительные требования

Шаги

Результат выполнения

Узнайте больше

Горячая миграция виртуальной машины

Overview

Ограничения и условия

Предварительные условия

Шаги выполнения

Восстановление виртуальной машины

Шаги для выполнения

Клонирование виртуальной машины

Предварительные требования

Шаги для выполнения

Связанные операции

Подготовка среды для физического GPU Passthrough

Ограничения и лимитации

Предварительные требования

Пошаговые действия

Проверка результата

Связанные операции

Настройка проброса USB-хоста

Содержание

Обзор функции

Сценарии использования

Предварительные требования

Шаги

- Открытие USB-устройств
- Назначение USB-устройств виртуальной машине

Результат выполнения

Узнайте больше

- Открытие нескольких USB-устройств
- Назначение USB-устройств виртуальной машине

Обзор функции

Функция проброса USB (Universal Serial Bus) позволяет получить доступ и управлять USB-устройствами из виртуальной машины.

Сценарии использования

Некоторые приложения, работающие в виртуальных машинах (VM), имеют требования к шифрованию и нуждаются во взаимодействии с выделенными USB-устройствами. В

таких случаях необходимо пробросить USB-устройства с хост-машины в виртуальную машину.

Предварительные требования

- Версия платформы должна быть не ниже v3.18.

Шаги

1 Открытие USB-устройств

Чтобы назначить USB-устройство виртуальной машине, USB-устройство должно быть открыто через **ResourceName**. Это можно настроить, отредактировав раздел `spec.permittedHostDevices.usbHostDevices` в **HyperConverged CR** в пространстве имён kubevirt.

Ниже приведён пример конфигурации для USB-устройства с **ResourceName** kubevirt.io/storage, где vendor — `0bda`, a product — `8812`:

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

> **Совет**
>
> Идентификаторы vendor и product USB-устройства можно получить с помощью команды `lsusb`. Например:

```
lsusb
Bus 001 Device 007: ID 0bda:8812 Realtek Semiconductor Corp. RTL8812AU
802.11a/b/g/n/ac 2T2R DB WLAN Adapter
```

Эта команда выводит список всех подключенных USB-устройств, где ID отображает пару vendor:product .

2 Назначение USB-устройств виртуальной машине

Теперь в конфигурации VM можно добавить `spec.domain.devices.hostDevices.deviceName`, чтобы сослаться на `ResourceName`, указанный на предыдущем шаге, и присвоить ему локальное имя. Например:

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

Совет

Убедитесь, что виртуальная машина остановлена перед редактированием конфигурации.

Результат выполнения

После завершения настройки выполните команду `lsusb` внутри виртуальной машины. Если в выводе отображается USB-устройство хост-узла, значит проброс прошёл успешно. Например:

```
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 802.11a/b/g/n/ac
2T2R DB WLAN Adapter
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Узнайте больше

Возможно, вам потребуется пробросить несколько USB-устройств в виртуальную машину, например клавиатуру, мышь или устройство смарт-карты. Мы поддерживаем назначение нескольких USB-устройств под одним resourceName. Вот как это настроить:

1 Открытие нескольких USB-устройств

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

Совет

Внимание: Все USB-устройства должны быть физически подключены и обнаружены на хосте для успешного назначения виртуальной машине.

2 Назначение USB-устройств виртуальной машине

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

Горячая миграция виртуальной машины

Содержание

Overview

ProCopy

Ограничения и условия

Предварительные условия

Шаги выполнения

Развертывание kubevirt-operator

Создание экземпляра HyperConverged

Подготовка виртуальной машины

Запуск горячей миграции

Overview

Технология горячей миграции виртуальной машины позволяет перемещать виртуальную машину с одного физического сервера на другой без выключения или прерывания работы виртуальной машины. Решение платформы для виртуальных машин реализовано на основе open-source компонента KubeVirt, который по умолчанию использует режим ProCopy для горячей миграции.

ProCopy

ProCopy (Pre-Copy Memory Migration) — широко используемая технология миграции виртуальных машин, обеспечивающая непрерывность сервиса во время миграции за

счёт предварительного копирования данных памяти виртуальной машины. Конкретный процесс следующий:

1. **Начальная фаза**: В начале миграции исходный хост копирует страницы памяти виртуальной машины на целевой хост, при этом виртуальная машина продолжает работать. Поскольку виртуальная машина продолжает работу, некоторые страницы памяти могут изменяться в процессе копирования.
2. **Итеративное копирование**: Исходный хост многократно копирует изменённые страницы памяти на целевой хост, пока количество изменённых страниц не уменьшится до приемлемого уровня. Каждый цикл копирования называется итерацией, и количество неизменённых страниц памяти постепенно уменьшается после каждой итерации.
3. **Остановка и копирование**: Когда оставшихся для копирования страниц памяти становится достаточно мало, виртуальная машина приостанавливается на короткое время (обычно от нескольких секунд до десятка секунд), в течение которого последние страницы памяти копируются на целевой хост, а состояние CPU и устройств виртуальной машины синхронизируется с целевым хостом.
4. **Возобновление работы**: Виртуальная машина возобновляет работу на целевом хосте.

Ограничения и условия

Рекомендуется, чтобы два физических сервера, участвующих в операции горячей миграции, имели одинаковую аппаратную конфигурацию. Если конфигурации отличаются (например, разные модели CPU), миграция может завершиться неудачей.

Предварительные условия

Пожалуйста, заранее включите соответствующие функции горячей миграции виртуальной машины.

Шаги выполнения

Развертывание kubevirt-operator

Примечание: Подробные шаги и описание параметров см. в Deploy Operator.

1. Перейдите в **Platform Management**.
2. В левой навигационной панели нажмите **App Store Management** > **Operators**.
3. В верхней части страницы нажмите **Cluster**, чтобы переключиться на кластер, в котором необходимо развернуть Operator.
4. На вкладке OperatorHub нажмите **Deploy** на карточке **KubeVirt HyperConverged Cluster Operator**.
5. Настройте параметры по необходимости и нажмите **Deploy**. Статус развертывания Operator можно проверить на вкладке **Deployed**.

Создание экземпляра HyperConverged

Для конкретных шагов создания см. Create HyperConverged Instance.

Подготовка виртуальной машины

Примечание: Рекомендуется использовать сеть Kube-OVN Underlay. По связанным настройкам см. Create Subnet (Kube-OVN Underlay Network).

1. Перейдите в **Container Platform**.
2. В левой навигационной панели нажмите **Virtualization** > **Virtual Machine**.
3. Нажмите **Create Virtual Machine**.
4. В области **Basic Information** нажмите **More**, чтобы развернуть дополнительные параметры конфигурации, затем нажмите **Add** напротив **Annotations** и добавьте аннотации согласно приведённым ниже ключам и значениям. Если используется сетевой плагин Kube-OVN, вручную заполнять эту аннотацию не нужно.

Примечание: Из-за ограничений формы сначала введите **значение** аннотации, затем **ключ**.

	Аннотация
Значение	true
Ключ	kubevirt.io/allow-pod-bridge-network-live-migration

5. Настройте остальные параметры виртуальной машины по необходимости. Для описания параметров см. соответствующую документацию продукта.

Параметр	**Описание**
Volume Mode	Обязательно использовать **Block Mode**.
Storage Class	Обязательно использовать класс хранения типа **CephRBD block storage**.
Network Mode	Рекомендуется использовать **Bridge**.

6. Нажмите **Create**.

Запуск горячей миграции

Примечание: Горячая миграция может быть запущена только при статусе виртуальной машины **Running**.

1. Перейдите в **Container Platform**.
2. В левой навигационной панели нажмите **Virtualization** > **Virtual Machine**.
3. Запустите горячую миграцию одним из двух способов:
   - Нажмите ⋮ > **Hot Migration** справа от виртуальной машины, которую нужно мигрировать, в списке.
   - Нажмите на имя виртуальной машины в списке, чтобы перейти на страницу с подробной информацией, затем нажмите **Actions** > **Hot Migration**.

4. Нажмите **Confirm**. Прогресс миграции можно отслеживать через **Virtual Machine Status** или **Real-Time Events**. Когда статус изменится с **Migrating** на **Running**, или в реальном времени появится событие с информацией вроде **Migrated: The VirtualMachineInstance migrated to node 10.1.1.1.**, это означает успешное завершение миграции.

Восстановление виртуальной машины

В некоторых случаях, например, при неправильных изменениях в fstab или ошибках файловой системы, требующих fsck, виртуальные машины могут не запускаться корректно. В таких ситуациях можно использовать режим восстановления для ремонта корневой файловой системы (rootfs) или извлечения данных из системы.

Содержание

Шаги для выполнения

- Получение адреса образа
- Изменение YAML-файла виртуальной машины
- Монтирование оригинального rootfs и выполнение ремонта
- Восстановление YAML-файла виртуальной машины

Шаги для выполнения

Получение адреса образа

1. В левой навигационной панели нажмите **Virtualization Management** > **Virtual Machine Images**.
2. Выберите предоставленный платформой **Source** как **Image Repository**, а **Operating System** — либо **CentOS**, либо **Ubuntu**. Нажмите ⋮ > **Update** справа.
3. Скопируйте и сохраните **Image Address**. В данном документе в качестве примера используется `192.168.1.1:11443/3rdparty/vmdisks/centos:7.9`.
4. Нажмите **Cancel**.

Изменение YAML-файла виртуальной машины

1. Зайдите в **Container Platform**.
2. В левой навигационной панели нажмите **Virtualization** > **Virtual Machines**.
3. Нажмите ⋮ > **Stop** справа у виртуальной машины, которую нужно отремонтировать, чтобы **Остановить** или **Принудительно остановить** её.
4. Нажмите ⋮ > **Update** справа у виртуальной машины.
5. Переключитесь на вкладку **YAML** и измените следующие поля.

   - Добавьте следующий блок под `spec.template.spec.domain.devices.disks`. Параметр bootOrder позволяет контролировать приоритет загрузки дисков виртуальной машины; чем меньше значение bootOrder, тем выше приоритет.

     Примечание: Если в исходном поле `spec.template.spec.domain.devices.disks` уже присутствует `bootOrder: 1`, увеличьте исходное значение, чтобы новое добавленное значение bootOrder было меньше исходного.

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

     Пример изменённого YAML:

```
domain:
  devices:
    disks:
      - bootOrder: 1  # Добавленное поле
        disk:
          bus: virtio
        name: containerdisk
      - disk:
          bus: virtio
        name: cloudinitdisk
      - disk:      # Увеличьте исходное значение bootOrder: 1
          bus: virtio
        name: rootfs
        bootOrder: 10
      - disk:
          bus: virtio
        name: "1"
```

- Добавьте следующий блок под `spec.template.spec.volumes`.

  Примечание: Замените адрес образа в поле `image` на тот, который был получен в разделе Получение адреса образа.

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

  Пример изменённого YAML:

```
volumes:
  - containerDisk:  # Добавленное поле
      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. Нажмите **Update**.

   Примечание: После изменения YAML-файла не переключайтесь на вкладку **Form**, просто нажмите **Update**.

7. Нажмите ⋮ > **Start** справа у виртуальной машины.

Монтирование оригинального rootfs и выполнение ремонта

1. Войдите в виртуальную машину, используя исходный пароль или ключ, и выполните команду `df -h /`, чтобы убедиться, что файловая система rootfs была заменена. Можно использовать команды, связанные с mount, чтобы смонтировать её, или команды fsck для проверки и ремонта оригинальной файловой системы.

2. По завершении выключите виртуальную машину.

Восстановление YAML-файла виртуальной машины

Следуйте шагам из раздела Изменение YAML-файла виртуальной машины, чтобы вернуть YAML-файл виртуальной машины в исходное состояние. После этого виртуальная машина сможет запускаться в обычном режиме.

Клонирование виртуальной машины

Клонирование виртуальной машины означает создание копии виртуальной машины на определённый момент времени. Клонированная виртуальная машина содержит все настройки, операционные системы, приложения и данные исходной виртуальной машины.

Содержание

Предварительные требования

Шаги для выполнения

Связанные операции

Просмотр и запуск клонированной виртуальной машины

Предварительные требования

- Для использования функции клонирования виртуальной машины её диск должен использовать класс хранилища, поддерживающий функции создания снимков. Снимки виртуальной машины сохраняют текущее состояние виртуальной машины, которое можно использовать для восстановления виртуальной машины на этот момент времени в случае непредвиденного сбоя.
- На узле, где будет выполняться скрипт, должен быть установлен пакет jq. jq — это лёгкий инструмент командной строки для обработки JSON, используемый для парсинга и обработки JSON-данных.

Шаги для выполнения

Примечание: Все операции ниже должны выполняться на мастер-узле кластера, где находится виртуальная машина.

1. Откройте CLI-инструмент.
2. Выполните следующую команду для создания и открытия файла vm-clone.sh.

```
vi vm-clone.sh
```

3. Нажмите `i` и скопируйте следующий содержимое в файл vm-clone.sh.

```
#!/bin/bash

vm_clone() {
  NAMESPACE=$1
  VM_NAME=$2
  VM_CLONE_NAME=$3

  cat <<EOF | kubectl create -f -
kind: VirtualMachineClone
apiVersion: clone.kubevirt.io/v1alpha1
metadata:
  name: clone-$VM_NAME
  namespace: $NAMESPACE
spec:
  source:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: $VM_NAME
  target:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: $VM_CLONE_NAME
  labelFilters:
    - "*"
    - "!ovn.kubernetes.io/*"
  annotationFilters:
    - "*"
    - "!ovn.kubernetes.io/*"
  template:
    labelFilters:
      - "*"
      - "!ovn.kubernetes.io/*"
    annotationFilters:
      - "*"
      - "!ovn.kubernetes.io/*"
EOF

  if [ $? -eq 0 ]; then
    echo "Create vmclone resource Succeeded"
  else
    echo "Create vmclone resource failed"
    exit 1
  fi
```

```
  echo "Waiting for vm clone completion"
  while true; do
    phase=$(kubectl -n $NAMESPACE get vmclone clone-$VM_NAME -o
jsonpath='{.status.phase}')
    if [ "$phase" == "Succeeded" ]; then
      break
    elif [ "$phase" == "Failed" ]; then
      echo "VirtualMachineClone resource phase is Failed"
      exit 1
    fi
    sleep 5
  done
  echo "vm clone completion"

  dvList=$(kubectl -n $NAMESPACE get vm $VM_CLONE_NAME -o
jsonpath='{.spec.template.spec.volumes}' | jq . | grep restore- | grep name | awk
'{print $2}')

  for dv in $dvList; do
    kubectl -n $NAMESPACE label --overwrite dv $(echo $dv | sed 's/"//g')
vm.cpaas.io/used-by=$VM_CLONE_NAME
    if [ $? -ne 0 ]; then
      echo "update DV label failed"
      exit 1
    fi
  done

  pvcList=$(kubectl -n $NAMESPACE get vm $VM_CLONE_NAME -o
jsonpath='{.spec.template.spec.volumes}' | jq . | grep restore- | grep claimName | awk
'{print $2}')

  for pvc in $pvcList; do
    kubectl -n $NAMESPACE label --overwrite pvc $(echo $pvc | sed 's/"//g')
vm.cpaas.io/used-by=$VM_CLONE_NAME
    if [ $? -ne 0 ]; then
      echo "update PVC label failed"
      exit 1
    fi
  done
}

if [ $# -ne 3 ]; then
  echo "error: parameters error"
```

```
  echo "Usage: ./vm-clone.sh NAMESPACE VM_NAME VM_CLONE_NAME"
  exit 1
fi

# exec vm clone
vm_clone "$1" "$2" "$3"
```

4. Нажмите `shift+:wq` для сохранения файла.

5. Выполните следующую команду, чтобы добавить права на выполнение для файла vm-clone.sh.

```
chmod +x vm-clone.sh
```

6. Выполните следующую команду для запуска скрипта.{#clone}

```
./vm-clone.sh <NAMESPACE> <VM_NAME> <VM_CLONE_NAME>
```

   Описание параметров:

   - **NAMESPACE**: Укажите namespace виртуальной машины, которую нужно клонировать, заменив часть <NAMESPACE> на этот namespace.
   - **VM_NAME**: Укажите имя виртуальной машины, которую нужно клонировать, заменив часть <VM_NAME> на это имя.
   - **VM_CLONE_NAME**: Укажите имя клонированной виртуальной машины, заменив часть <VM_CLONE_NAME> на это имя.

7. Когда появится сообщение, похожее на следующее, это означает, что клонирование завершено.

```
virtualmachineclone.clone.kubevirt.io/clone-k1 created
Create vmclone resource Succeeded
Waiting for vm clone completion
vm clone completion
datavolume.cdi.kubevirt.io/restore-e8ff0e7b-dc7e-4140-aec7-8556cfcf4533-rootfs labeled
datavolume.cdi.kubevirt.io/restore-e8ff0e7b-dc7e-4140-aec7-8556cfcf4533-1 labeled
```

Связанные операции

Просмотр и запуск клонированной виртуальной машины

1. Перейдите в **Platform Management**.
2. В левой навигационной панели нажмите **Virtualization Management** > **Virtual Machine Images**.
3. Вы увидите клонированную виртуальную машину с указанным именем из шага Run Script; состояние клонированной виртуальной машины по умолчанию — **Stopped**.
4. Нажмите на имя этой виртуальной машины, и страница перенаправит вас на страницу с деталями виртуальной машины в Container Platform.
5. Нажмите **Start**, чтобы успешно запустить виртуальную машину.

Подготовка среды для физического GPU Passthrough

Физический GPU passthrough в виртуальных машинах — это процесс прямого выделения реального графического процессора (GPU) виртуальной машине в среде виртуализации. Это позволяет виртуальной машине напрямую получать доступ к физическому GPU и использовать его, достигая графической производительности, эквивалентной работе на физической машине. Такой подход избегает узких мест производительности, вызванных виртуальными графическими адаптерами, тем самым повышая общую производительность.

Содержание

- Ограничения и лимитации
- Предварительные требования
  - Подготовка Chart и образов
  - Включение IOMMU
- Пошаговые действия
  - Создание Namespace
  - Развёртывание gpu-operator
  - Настройка Kubevirt
- Проверка результата
- Связанные операции
  - Удаление виртуальной машины с passthrough GPU
  - Удаление конфигурации, связанной с GPU, из KubeVirt
  - Удаление gpu-operator

Ограничения и лимитации

Функциональность физического GPU passthrough требует использования kubevirt-gpu-device-plugin; однако в настоящее время отсутствует ARM64-образ для kubevirt-gpu-device-plugin, что означает, что данная функциональность не может быть использована в операционной системе с архитектурой CPU ARM64.

Предварительные требования

Подготовка Chart и образов

Получите следующие Chart и образы и загрузите их в репозиторий образов. В этом документе в качестве примера используется адрес репозитория `build-harbor.example.cn`. Для конкретного способа получения Chart и образов обратитесь к соответствующим специалистам.

Chart

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

Образы

- 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

Включение IOMMU

Процедура включения IOMMU различается в зависимости от операционной системы. Пожалуйста, обратитесь к документации соответствующей ОС. В данном документе в

качестве примера используется CentOS, все команды необходимо выполнять в терминале.

1. Отредактируйте файл `/etc/default/grub` и добавьте `intel_iommu=on iommu=pt` в опцию конфигурации `GRUB_CMDLINE_LINUX`.

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

2. Выполните следующую команду для генерации файла `grub.cfg`.

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

3. Перезагрузите сервер.

4. Выполните команду для проверки успешного включения IOMMU. Если в выводе содержится `IOMMU enabled`, значит IOMMU успешно включён.

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

Пошаговые действия

Примечание: Все команды ниже необходимо выполнять в CLI на соответствующем Master-узле кластера, если не указано иное.

Создание Namespace

Выполните команду для создания namespace с именем `gpu-system`. Если в выводе появится `namespace/gpu-system created`, значит создание прошло успешно.

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

Развёртывание gpu-operator

1. Выполните команду для развёртывания gpu-operator.

```
export REGISTRY=<registry> # Замените <registry> на адрес репозитория, где
находится образ gpu-operator, например: export REGISTRY=build-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> # Замените <chartName> на реальный путь к chart, например:
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. Выполните команду для проверки синхронизации gpu-operator. Если в столбце `SYNC` отображается `Synced`, значит синхронизация прошла успешно.

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

Пример вывода:

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

3. Выполните команду для получения списка всех узлов и определения имени GPU-узла.

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

4. Выполните команду для проверки наличия GPU, поддерживающего passthrough, на GPU-узле. Если в выводе содержится информация о GPU, например `nvidia.com/GK210GL_TESLA_K80`, значит есть GPU, поддерживающие passthrough.

```
kubectl get node <gpu-node-name> -o jsonpath='{.status.allocatable}' # Замените <gpu-node-name> на имя GPU-узла, полученное на шаге 3
```

Пример вывода:

```
{"cpu":"39","devices.kubevirt.io/kvm":"1k","devices.kubevirt.io/tun":"1k","devices.kubevirtnet":"1k","ephemeral-storage":"426562784165","hugepages-1Gi":"0","hugepages-2Mi":"0","memory":"122915848Ki","nvidia.com/GK210GL_TESLA_K80":"8","pods":"256"}
```

5. На этом этапе gpu-operator успешно развернут.

Настройка Kubevirt

1. Выполните команду для включения функции DisableMDEVConfiguration. Если возвращается сообщение, похожее на `hyperconverged.hco.kubevirt.io/kubevirt-hyperconverged patched`, значит функция успешно включена.

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

2. В терминале GPU-узла выполните команду для получения pciDeviceSelector. Часть `10de:102d` в выводе — это значение pciDeviceSelector. {#pciDeviceSelector}

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

   Пример вывода:

   ```
   04:00.0 3D controller [0302]: NVIDIA Corporation GK210GL [Tesla K80] [10de:102d] (rev
   a1)
   05:00.0 3D controller [0302]: NVIDIA Corporation GK210GL [Tesla K80] [10de:102d] (rev
   a1)
   08:00.0 3D controller [0302]: NVIDIA Corporation GK210GL [Tesla K80] [10de:102d] (rev
   a1)
   09:00.0 3D controller [0302]: NVIDIA Corporation GK210GL [Tesla K80] [10de:102d] (rev
   a1)
   85:00.0 3D controller [0302]: NVIDIA Corporation GK210GL [Tesla K80] [10de:102d] (rev
   a1)
   86:00.0 3D controller [0302]: NVIDIA Corporation GK210GL [Tesla K80] [10de:102d] (rev
   a1)
   89:00.0 3D controller [0302]: NVIDIA Corporation GK210GL [Tesla K80] [10de:102d] (rev
   a1)
   8a:00.0 3D controller [0302]: NVIDIA Corporation GK210GL [Tesla K80] [10de:102d] (rev
   a1)
   ```

3. Выполните команду для получения списка всех узлов и определения имени GPU-узла.

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

4. Выполните команду для получения resourceName. Часть `nvidia.com/GK210GL_TESLA_K80` в выводе — это значение resourceName.

```
kubectl get node <gpu-node-name> -o jsonpath='{.status.allocatable}' # Замените <gpu-node-name> на имя GPU-узла, полученное на шаге 3
```

Пример вывода:

```
{"cpu":"39","devices.kubevirt.io/kvm":"1k","devices.kubevirt.io/tun":"1k","devices.kubevirtnet":"1k","ephemeral-storage":"426562784165","hugepages-1Gi":"0","hugepages-2Mi":"0","memory":"122915848Ki","nvidia.com/GK210GL_TESLA_K80":"8","pods":"256"}
```

5. Выполните команду для добавления passthrough GPU.

   Примечание: При замене части <pci-devices-id> в команде ниже на значение pciDeviceSelector, полученное в Шаге 2, **все буквы pciDeviceSelector должны быть преобразованы в верхний регистр**. Например, если pciDeviceSelector равен `10de:102d`, его нужно заменить на `export DEVICE=10DE:102D`.

   - Добавление одной GPU-карты

```
export DEVICE=<pci-devices-id> # Замените <pci-devices-id> на pciDeviceSelector, полученный на шаге 2, например: export DEVICE=10DE:102D
export RESOURCE=<resource-name> # Замените <resource-name> на resourceName, полученный на шаге 4, например: export RESOURCE=nvidia.com/GK210GL_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"'"
        }
      ]
    }
  }
]'
```

- Добавление нескольких GPU-карт

  Примечание: При добавлении нескольких GPU-карт каждое значение pciDeviceSelector, используемое для замены <pci-devices-id>, должно быть уникальным.

```
export DEVICE1=<pci-devices-id1> # Замените <pci-devices-id1> на
pciDeviceSelector, полученный на шаге 2
export RESOURCE1=<resource-name1> # Замените <resource-name1> на resourceName,
полученный на шаге 4
export DEVICE2=<pci-devices-id2> # Замените <pci-devices-id2> на
pciDeviceSelector, полученный на шаге 2
export RESOURCE2=<resource-name2> # Замените <resource-name2> на resourceName,
полученный на шаге 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"'"
        }
      ]
    }
  }
]'
```

- Добавление новых GPU-карт после уже добавленных

```
export DEVICE=<pci-devices-id> # Замените <pci-devices-id> на pciDeviceSelector, полученный на шаге 2
export RESOURCE=<resource-name> # Замените <resource-name> на resourceName, полученный на шаге 4
export INDEX=<index> # index – это индекс массива с нуля, используйте число для замены <index>, например: если уже добавлена одна GPU-карта, и нужно добавить ещё одну, индекс должен быть 1, т.е. 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"'"
    }
  }
]'
```

Проверка результата

После выполнения вышеуказанных шагов настройки, если при создании виртуальной машины можно выбрать соответствующий физический GPU, значит среда физического GPU passthrough успешно подготовлена.

Примечание: Если требуется настроить физический GPU passthrough, пожалуйста, заранее включите соответствующие функции.

1. Перейдите в **Container Platform**.
2. В левой панели навигации выберите **Virtualization** > **Virtual Machines**.
3. Нажмите **Create Virtual Machine**.
4. Настройте параметр **Physical GPU (Alpha)** для виртуальной машины.

Параметр	Описание
Physical GPU (Alpha)	Выберите модель настроенного физического GPU. Каждой виртуальной машине может быть назначен только один физический GPU.

5. На этом этапе среда физического GPU passthrough успешно подготовлена.

Связанные операции

Удаление виртуальной машины с passthrough GPU

1. Перейдите в **Container Platform**.
2. В левой панели навигации выберите **Virtualization** > **Virtual Machines**.
3. На странице списка нажмите ⋮ справа от виртуальной машины, которую нужно удалить, и выберите **Delete**, либо перейдите на страницу деталей виртуальной машины и выберите **Actions** > **Delete**.
4. Введите подтверждающую информацию для удаления виртуальной машины с passthrough GPU.

Удаление конфигурации, связанной с GPU, из KubeVirt

1. На соответствующем Master-узле кластера для GPU выполните в CLI следующую команду для удаления конфигурации, связанной с GPU, из KubeVirt.

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

2. После удаления, если при создании виртуальной машины через **Container Platform** невозможно выбрать соответствующую модель физического GPU, значит удаление прошло успешно. Для конкретных шагов создания виртуальной машины обратитесь к разделу Select Physical GPU Model.

Удаление gpu-operator

1. На соответствующем Master-узле кластера для GPU выполните в CLI следующую команду для удаления gpu-operator.

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

   Пример вывода:

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

2. Выполните команду, и если получите ответ, похожий на приведённый ниже, значит gpu-operator успешно удалён.

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

   Пример вывода:

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

Menu

Устранение неполадок

Миграция Pod виртуальных машин и восстановление после аварийного завершения работы узлов виртуальных машин

Описание проблемы

Анализ причины

Решения

Сообщения об ошибках горячей миграции и решения

Миграция Pod виртуальных машин и восстановление после аварийного завершения работы узлов виртуальных машин

Содержание

Описание проблемы

Анализ причины

Решения

Миграция Pod виртуальных машин при корректном завершении работы

Восстановление после аварийного завершения работы

Описание проблемы

Независимо от того, происходит ли **корректное завершение работы** узла или **аварийный сбой**, Pod виртуальных машин, запущенных на этом узле, не будут автоматически мигрировать на другие здоровые узлы.

Анализ причины

Платформа реализует решение для виртуальных машин на основе открытого компонента KubeVirt. Однако с точки зрения KubeVirt невозможно отличить реальный сбой виртуальной машины от потери соединения, вызванной сетевыми или другими

проблемами. Если виртуальные машины мигрировать на другие узлы без разбора, это может привести к одновременному существованию нескольких экземпляров одной и той же виртуальной машины.

Решения

При обслуживании узлов виртуальных машин необходимо выполнять ручные действия согласно данной документации. В случаях как **корректного завершения работы**, так и **аварийного сбоя** Pod виртуальных машин должны быть вручную эвакуированы или принудительно удалены.

Примечание: Следующие команды необходимо выполнять на Master-узле соответствующего кластера.

Миграция Pod виртуальных машин при корректном завершении работы

1. В CLI выполните следующую команду для получения информации об узлах. Поле `NAME` в выводе — это `Node-Name`.

   ```
   kubectl get nodes
   ```

   Вывод:

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

2. (Опционально) Выполните следующую команду для просмотра экземпляров виртуальных машин на узле.

   ```
   kubectl get vmis --all-namespaces -o wide | grep <Node-Name>  # Замените <Node-Name>
   в команде на Node-Name, полученный на шаге 1
   ```

Вывод:

```
test-test         vm-t-export-clone   13d     Running      1.1.1.1   1.1.1.211   True
False
```

3. Перед корректным завершением работы выполните следующую команду для эвакуации всех Pod виртуальных машин на узле, который будет выключен. Если вывод выглядит следующим образом, это означает успешную эвакуацию.

```
kubectl drain <Node-Name> --delete-local-data --ignore-daemonsets=true --force --pod-
selector=kubevirt.io=virt-launcher   # Замените <Node-Name> в команде на Node-Name
узла, который будет выключен
```

Вывод:

```
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. После того как все виртуальные машины запущены на других узлах, выключите узел.

5. После выключения и перезагрузки узла выполните следующую команду, чтобы снять с него запрет на планирование Pod.

```
kubectl uncordon <Node-Name> # Замените <Node-Name> в команде на Node-Name
выключенного и перезагруженного узла
```

Вывод:

```
node/1.1.1.211 uncordoned
```

6. На этом этапе исходные экземпляры виртуальных машин с этого узла были мигрированы на другие здоровые узлы, а данный узел после перезагрузки доступен для планирования новых Pod.

Восстановление после аварийного завершения работы

1. В CLI выполните следующую команду для получения информации об узлах. Поле `NAME` в выводе — это `Node-Name`.

   ```
   kubectl get nodes
   ```

   Вывод:

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

2. Выполните следующую команду для принудительного удаления всех Pod виртуальных машин на узле.

   ```
   kubectl get po -A -l kubevirt.io=virt-launcher -o wide | grep <Node-Name> | awk '{print "kubectl delete pod --force -n " $1, $2}'  | bash  # Замените <Node-Name> в команде на Node-Name узла, который аварийно завершил работу.
   ```

3. Выполните следующую команду для удаления volume attachments на этом узле.

   ```
   kubectl get volumeattachments.storage.k8s.io | grep <Node-Name> | awk '{print $1}' | xargs kubectl delete volumeattachments.storage.k8s.io  # Замените <Node-Name> в команде на Node-Name узла, который аварийно завершил работу.
   ```

4. Выполните следующую команду для проверки наличия Pod с меткой kubevirt.io=virt-api на узле, который аварийно завершил работу.

```
kubectl -n kubevirt get po -l kubevirt.io=virt-api -o wide | grep <Node-Name> # Замените <Node-Name> в команде на Node-Name узла, который аварийно завершил работу.
```

Если такие Pod существуют, выполните следующую команду для их удаления.

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

5. Выполните следующую команду для проверки наличия Pod с меткой kubevirt.io=virt-controller на узле, который аварийно завершил работу.

```
kubectl -n kubevirt get po -l kubevirt.io=virt-controller -o wide | grep <Node-Name> # Замените <Node-Name> в команде на Node-Name узла, который аварийно завершил работу.
```

Если такие Pod существуют, выполните следующую команду для их удаления.

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

6. На этом этапе экземпляры виртуальных машин будут мигрированы на другие здоровые узлы после аварийного завершения работы узла.

Menu

Сообщения об ошибках горячей миграции и решения

Сообщение об ошибке	Причина	Решение
cannot migrate VMI which does not use masquerade, bridge with <annotation> VM annotation or a migratable plugin to connect to the pod network	Сетевая конфигурация виртуальной машины не поддерживает горячую миграцию.	Пожалуйста, проверьте следующие настройки: • Проверьте CNI сетевой плагин, используемый в текущем кластере; рекомендуется Kube-OVN. • Проверьте наличие аннотации "kubevirt.io/allow-pod-bridge-network-live-migration": "true" в полях `metadata.annotations` и `spec.template.metadata.annotations` соответствующего YAML-файла виртуальной машины; если отсутствует, добавьте её вручную.
• 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)	Тип хранилища виртуальной машины не поддерживает многозадачный режим чтения-записи (RWX).	Параметры, связанные с виртуальной машиной, нельзя изменить после создания. Поэтому, пожалуйста, пересоздайте виртуальную машину, выбрав тип хранилища, поддерживающий многозадачный режим чтения-записи (RWX); рекомендуется блочное хранилище CephRBD. Если проблемы сохраняются после пересоздания, обратитесь к соответствующим специалистам за помощью.

Сообщение об ошибке	Причина	Решение
• 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		
Другие сообщения об ошибках	Виртуальная машина не поддерживает горячую миграцию.	Пожалуйста, обратитесь к соответствующим специалистам за помощью.

Menu

Сеть

Введение

Введение

Преимущества

Руководства

Настройка сети

Настройка IP

Прямое подключение к виртуальной машине по IP

Добавление внутренних маршрутов

Как сделать

Контроль сетевых запросов виртуальной машины через Network Policy

Процедура

Проверка результата

Настройка SR-IOV

Терминология

Ограничения и лимиты

Предварительные условия

Процедуры

Проверка результата

Связанные заметки

Настройка виртуальных машин для использования режима сетевого биндинга с поддержкой IPv6

Требования

Процедура

Введение

ACP Virtualization With KubeVirt тесно интегрирована с Kube-OVN, расширяя поддержку традиционных требований к сетям виртуальных машин (VM) и оптимизируя производительность для конкретных сценариев.

Содержание

Преимущества

Преимущества

- Поддержка IPv6

  Полная поддержка IPv6.

- Сохранение статического IP

  Обеспечивает сохранение у виртуальных машин одного и того же IP-адреса после перезапусков, что соответствует устоявшимся паттернам использования VM.

- Поддержка мультисетевого режима

  Поддерживает несколько сетевых режимов, таких как контейнерные сети и SR-IOV, что позволяет удовлетворять разнообразные пользовательские сценарии.

Menu

Руководства

Настройка сети

Настройка IP

Прямое подключение к виртуальной машине по IP

Добавление внутренних маршрутов

Настройка сети

Содержание

Настройка IP

Прямое подключение к виртуальной машине по IP

Добавление внутренних маршрутов

Настройка IP

Ссылки на Configure IP

Прямое подключение к виртуальной машине по IP

Ссылки на Preparing Kube-OVN Underlay Physical Network

Добавление внутренних маршрутов

Ссылки на Add Internal Routes

Menu

Практическое руководство

Контроль сетевых запросов виртуальной машины через Network Policy

Процедура

Проверка результата

Настройка SR-IOV

Терминология

Ограничения и лимиты

Предварительные условия

Процедуры

Проверка результата

Связанные заметки

Настройка виртуальных машин для использования режима сетевого биндинга с поддержкой IPv6

Требования

Процедура

Контроль сетевых запросов виртуальной машины через Network Policy

Решение платформы для виртуальных машин реализовано на базе open-source компонента KubeVirt, который фактически работает внутри Pod'ов. Используя функциональность Network Policies, можно контролировать входящие и исходящие запросы виртуальных машин.

Содержание

Процедура

Проверка результата

Шаг первый: Создание виртуальной машины и Network Policy, разрешающей весь трафик

Шаг второй: Обновление Network Policy для исключения www.example.com из белого списка

Процедура

1. Войдите в **Container Platform**.
2. В левой навигационной панели нажмите **Network** > **Network Policies**.
3. Нажмите **Create Network Policy**.
4. Настройте следующие параметры по необходимости.

Параметр	Описание
Association Method	• **Compute Component**: Выберите целевой вычислительный компонент по необходимости; рекомендуется выбрать **All** в

Параметр	Описание
	качестве целевого вычислительного компонента. • **Label Selector**: Соответствие Pod'ов по меткам.
Direction	• **Ingress**: Запросы, направленные извне к Pod. • **Egress**: Запросы, направленные из Pod во внешний мир; выберите этот вариант, если необходимо запретить виртуальной машине делать запросы к определённому внешнему адресу.
Protocol	Выберите TCP или UDP. **Примечание**: • При использовании доменных имён в виртуальной машине для запросов к внешним сервисам необходимо добавить белый список для протокола UDP, так как DNS использует UDP. • Форма не поддерживает настройку протокола ICMP; при включении правил белого списка протокол ICMP будет отключён, что приведёт к невозможности выполнять операции Ping.
Access Ports	Укажите, трафик по каким портам разрешён для входящих или исходящих запросов. Если поле оставить пустым, по умолчанию разрешён трафик по всем портам. **Примечание**: Необходимо разрешить порты 1053 и 53 для протоколов UDP и TCP, чтобы разрешить исходящий DNS-трафик; иначе разрешение доменных имён не будет работать.
Remote Type	Укажите разрешённый тип удалённого доступа. Варианты: вычислительный компонент, namespace, IP-сегменты.

Параметр	Описание
Exclude Remote	Если тип удалённого доступа — **IP Segment**, исключите указанный IP из белого списка (то есть запретите доступ). Один IP можно исключить, указав его в формате `IP/32`. **Примечание**: В это поле можно вводить только IP; если соответствующий IP доменного имени неизвестен, используйте команду `curl -vvv <domain>`, чтобы запросить домен и получить IP из возвращаемой информации.

5. Нажмите **Create**.

Проверка результата

В данном документе проверяется настройка с использованием виртуальной машины для доступа к www.example.com↗.

Шаг первый: Создание виртуальной машины и Network Policy, разрешающей весь трафик

1. Создайте виртуальную машину, подробные шаги смотрите в разделе Create Virtual Machine.

2. Настройте Network Policy в namespace виртуальной машины, добавив правила белого списка для протоколов TCP и UDP со следующими параметрами:

   - Правила белого списка для протокола TCP:

Параметр	Описание
Association Method	Выберите **Compute Component**.
Target Compute Component	Выберите **All**.
Direction	Выберите **Egress**.

Параметр	Описание
Protocol	Выберите **TCP**.
Remote Type	Выберите **IP Segment**
Remote	Введите **0.0.0.0/0**, что означает разрешение всего исходящего трафика.

- Правила белого списка для протокола UDP:

Параметр	Описание
Direction	Выберите **Egress**.
Protocol	Выберите **UDP**.
Remote Type	Выберите **IP Segment**
Remote	Введите **0.0.0.0/0**, что означает разрешение всего исходящего трафика.

3. После создания Network Policy войдите в виртуальную машину и выполните команду для запроса www.example.com↗:

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

4. Запрос выполнен успешно.

Шаг второй: Обновление Network Policy для исключения www.example.com↗ из белого списка

1. Выполните команду для получения IP-адреса www.example.com↗, в результате будет получен IP 93.184.215.14.

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

2. Обновите Network Policy, созданную на Шаге первом, с обновлёнными параметрами:

Параметр	Описание
Exclude Remote	В правилах белого списка для протокола TCP заполните параметр exclude remote значением 93.184.215.14/32, что означает исключение IP 93.184.215.14 из белого списка.

3. После обновления Network Policy войдите в виртуальную машину и выполните команду для запроса www.example.com ↗:

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

4. Запрос завершается по таймауту, что свидетельствует о корректной работе функции исключения удалённого адреса.

Настройка SR-IOV

Настраивая физические серверные узлы для поддержки создания виртуальных машин с сетевыми картами SR-IOV (Single Root I/O Virtualization), достигается снижение задержек для виртуальных машин, а также поддержка автономного IPv6 и функциональности двойного стека IPv4/IPv6.

Содержание

- Терминология
- Ограничения и лимиты
- Предварительные условия
  - Чарт
  - Образы
- Процедуры
  - Включение SR-IOV в BIOS физической машины
  - Включение IOMMU
  - Загрузка модуля VFIO в ядро системы
  - Создание устройств VF
  - Привязка драйвера VFIO
  - Развёртывание плагина Multus CNI
  - Развёртывание sriov-network-operator
  - Установка меток идентификаторов ролей узлов для физических узлов
  - Проверка успешного создания ресурсов
  - Установка меток функций узлов SR-IOV для физических узлов
  - Проверка поддержки устройства NIC
  - Настройка IP-адреса
- Проверка результата

Связанные заметки

Конфигурация параметров ядра для виртуальных машин CentOS

Терминология

Термин	Определение
Multus CNI	Выступает в роли промежуточного слоя для других CNI-плагинов, позволяя Kubernetes поддерживать несколько сетевых интерфейсов для Pod’ов.
SR-IOV	Позволяет виртуализировать физическую сетевую карту (NIC) на узле, разделяя её на несколько виртуальных функций (VF) для использования Pod’ами или виртуальными машинами, обеспечивая превосходную производительность сети.
VF	Виртуальное устройство, созданное из физического PCI-устройства; VF могут быть выделены напрямую виртуальным машинам или контейнерам, имитируя независимые физические PCI-устройства, что значительно улучшает производительность ввода-вывода.

Ограничения и лимиты

Функция SR-IOV зависит от glibc и поддерживает только версии glibc 2.34 и выше. Однако операционные системы Kylin V10 и CentOS 7.x не поддерживают эту версию, поэтому функциональность SR-IOV не может быть использована на этих ОС.

Предварительные условия

Получите следующие чарты и образы и загрузите их в репозиторий образов. В этом документе в качестве примера используется адрес репозитория `build-harbor.example.cn`. Для получения конкретных способов получения чартов и образов обратитесь к соответствующим специалистам.

Чарт

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

Образы

- `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`

Процедуры

Примечание: Все команды, упомянутые ниже, выполняются в терминале.

1 Включение SR-IOV в BIOS физической машины

Перед настройкой выполните следующую команду для проверки информации о материнской плате.

```
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
```

Операция включения SR-IOV в BIOS различается у разных производителей серверов. Пожалуйста, обратитесь к документации соответствующего производителя. Обычно шаги следующие:

1. Перезагрузите сервер.
2. При появлении логотипа бренда на экране во время POST BIOS нажмите клавишу F2 для входа в настройки системы.
3. Перейдите в **Processor Settings** > **Virtualization Technology** и измените параметр **Virtualization Technology** на `Enabled`.
4. Перейдите в **Settings** > **Integrated devices** и измените параметр **SR-IOV Global Enable** на `Enabled`.
5. Сохраните конфигурацию и перезагрузите сервер.

2 Включение IOMMU

Операция включения IOMMU может отличаться в разных операционных системах. Пожалуйста, обратитесь к документации вашей ОС. В этом документе используется CentOS в качестве примера.

1. Отредактируйте файл `/etc/default/grub`, добавив `intel_iommu=on iommu=pt` в параметр `GRUB_CMDLINE_LINUX`.

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

2. Выполните команду для генерации файла `grub.cfg`.

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

3. Перезагрузите сервер.

4. Выполните команду, и если в выводе будет `IOMMU enabled`, значит включение прошло успешно.

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

3 Загрузка модуля VFIO в ядро системы

1. Выполните команду для загрузки модуля vfio-pci.

```
modprobe vfio-pci
```

2. После загрузки выполните следующую команду. Если информация выводится корректно, значит модуль VFIO загружен успешно.

```
# Для CentOS выполните команду для проверки статуса загрузки VFIO
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


# Для Ubuntu выполните команду для проверки статуса загрузки VFIO
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 Создание устройств VF

1. Выполните команду для просмотра поддерживаемых в данный момент устройств VF.

```
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
```

   Вывод означает следующее:

   - **0000:05:00 .1**: PCI-адрес физической SR-IOV сетевой карты enp5s0f1.
   - **0000:05:00 .0**: PCI-адрес физической SR-IOV сетевой карты enp5s0f0.
   - **sriov_totalvfs**: Количество поддерживаемых VF.
   - **sriov_numvfs**: Текущее количество VF.

2. Выполните команду для получения информации о сетевой карте физической машины.

```
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<global>
        inet6 fe80::a236:9fff:fe29:6c00  prefixlen 64  scopeid 0x20<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 0x20<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. Выполните команду `ethtool -i <имя NIC>`, чтобы получить PCI-адрес соответствующей физической сетевой карты, как показано ниже.

```
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     ## PCI-адрес сетевой карты enp5s0f0
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    ## PCI-адрес сетевой карты enp5s0f1
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-priv-flags: yes
```

4. Выполните команду для создания VF. В этом документе в качестве примера используется настройка NIC enp5s0f1. Если необходимо виртуализировать несколько NIC, настройте все их.

```
cat /sys/devices/pci0000:00/0000:00:03.0/0000:05:00.1/sriov_totalvfs   ##
Проверка количества поддерживаемых VF
63

echo 8 > /sys/devices/pci0000:00/0000:00:03.0/0000:05:00.1/sriov_numvfs  ##
Установка текущего количества VF

cat /sys/devices/pci0000:00/0000:00:03.0/0000:05:00.1/sriov_numvfs   ##
Проверка текущего количества VF
8
```

5. Выполните команду для проверки успешного создания VF.

   Примечание: Вы увидите настроенные 8 адресов VF, например `05:10.1`. Эти адреса VF необходимо дополнить **идентификатором домена**, в результате получится формат: `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 Controller Virtual
Function (rev 01)
05:10.3 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
05:10.5 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
05:10.7 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
05:11.1 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
05:11.3 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
05:11.5 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
05:11.7 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual
Function (rev 01)
```

5 Привязка драйвера VFIO

1. Скачайте скрипт привязки и выполните команду `python3 dpdk-devbind.py -b vfio-pci <адрес VF с идентификатором домена>`, чтобы привязать 8 VF сетевой карты enp5s0f1 к драйверу vfio-pci, как показано ниже.

```
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. После успешной привязки выполните команду для проверки результатов. В выводе найдите уже привязанные VF в разделе **Network devices using DPDK-compatible driver**. Среди них ID устройства VF — `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=eno1 drv=tg3
unused=vfio-pci
0000:01:00.1 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=eno2 drv=tg3
unused=vfio-pci
0000:02:00.0 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=eno3 drv=tg3
unused=vfio-pci
0000:02:00.1 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=eno4 drv=tg3
unused=vfio-pci
0000:05:00.0 'Ethernet 10G 2P X520 Adapter 154d' if=enp5s0f0 drv=ixgbe
unused=vfio-pci *Active*
0000:05:00.1 'Ethernet 10G 2P X520 Adapter 154d' if=enp5s0f1 drv=ixgbe
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 Развёртывание плагина Multus CNI

1. Перейдите в **Platform Management**.
2. В левой навигационной панели выберите **Cluster Management** > **Clusters**.
3. Нажмите на имя кластера виртуальных машин и перейдите на вкладку **Plugins**.
   - Разверните плагин **Multus CNI**.

7 Развёртывание sriov-network-operator

Выполните следующую команду для развёртывания sriov-network-operator.

```
REGISTRY=<$registry>  # Замените <$registry> на адрес репозитория, где находится образ sriov-network-operator, например: REGISTRY=build-harbor.example.cn
NICSELECTOR=["<nics>"] # Замените <nics> на имена NIC, например: NICSELECTOR=["ens802f1","ens802f2"], разделяйте несколько через запятую
NUMVFS=<numVfs> # Замените <numVfs> на количество VF, например: 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> # Замените <chartName> на фактический путь чарта, например: 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 Установка меток идентификаторов ролей узлов для физических узлов

Примечание: Перед выполнением убедитесь, что Pod `sriov-network-operator` работает нормально.

1. Перейдите в **Platform Management**.
2. В левой навигационной панели выберите **Cluster Management** > **Clusters**.
3. Нажмите на имя кластера и перейдите на вкладку **Nodes**.
4. Нажмите на физический узел, поддерживающий SR-IOV ⋮ > **Update Node Labels**.
5. Установите метку узла следующим образом:
   - `node-role.kubernetes.io/worker: ""`
6. Нажмите **Update**.

9 Проверка успешного создания ресурсов

В CLI выполните команду `kubectl -n cpaas-system get sriovnetworknodestates`, чтобы проверить, был ли успешно создан ресурс `sriovnetworknodestates`. Если вы видите вывод, похожий на приведённый ниже, значит создание прошло успешно. Если создание ресурса не удалось, проверьте успешность развёртывания плагина Multus CNI и sriov-network-operator.

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

10 Установка меток функций узлов SR-IOV для физических узлов

Примечание: Перед выполнением убедитесь, что ресурс `sriovnetworknodestates` был успешно создан.

1. Перейдите в **Platform Management**.
2. В левой навигационной панели выберите **Cluster Management** > **Clusters**.
3. Нажмите на имя кластера и перейдите на вкладку **Nodes**.
4. Нажмите на физический узел, поддерживающий SR-IOV ⋮ > **Update Node Labels**.

5. Установите метку узла следующим образом:

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

11 Проверка поддержки устройства NIC

1. Выполните команду `lspci -n -s <адрес VF с идентификатором домена>`, чтобы получить текущие vendor ID и device ID устройства NIC, как показано ниже.

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

   Вывод означает:

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

2. Выполните команду `lspci -s <адрес VF с идентификатором домена> -vvv | grep Ethernet`, чтобы получить текущее имя NIC, как показано ниже.

   ```
   lspci -s 0000:05:00.1 -vvv | grep Ethernet
   05:00.1 Ethernet controller: Intel Corporation Ethernet 10G 2P X520 Adapter (rev
   01)
   ```

3. В пространстве имён cpaas-system найдите конфигурационный файл с именем `supported-nic-ids` типа ConfigMap и проверьте, содержится ли информация о текущем NIC в списке поддержки в разделе data.

   Примечание: Если текущий NIC отсутствует в списке поддержки, необходимо обратиться к Шагу 4 для добавления NIC в конфигурационный файл. Если NIC уже в списке, пропустите Шаг 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. Добавьте текущий NIC в раздел data списка поддержки в формате `<Имя NIC>: <Vendor ID> <Device ID> <VF Device ID>`, как показано ниже.

```
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            ## Добавление новой
информации о NIC

  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
```

Объяснение параметров:

- **Intel_Corporation_X520**: Имя NIC, можно задать произвольно.
- **8086**: Vendor ID.
- **154d**: Device ID.
- **10ed**: VF Device ID, который можно найти в результатах привязки.

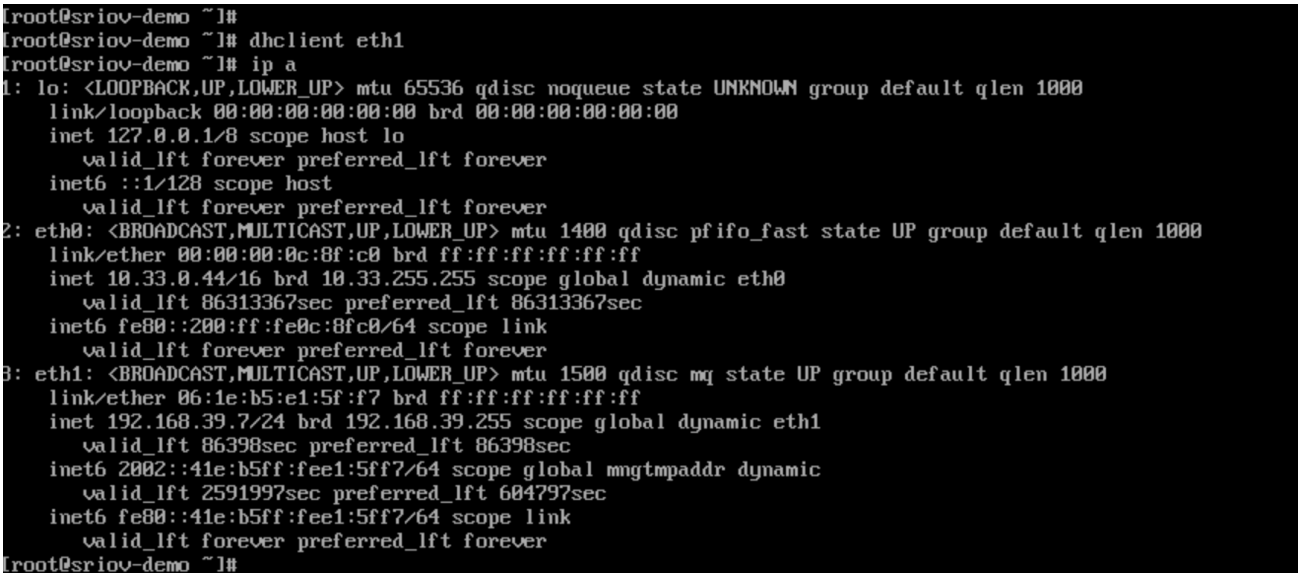
12 Настройка IP-адреса

Войдите в коммутатор для настройки DHCP (Dynamic Host Configuration Protocol).

Примечание: Если использование DHCP невозможно, настройте IP-адрес вручную в виртуальной машине.

Проверка результата

1. Перейдите в **Container Platform**.
2. В левой навигационной панели выберите **Virtualization** > **Virtual Machines**.
3. Нажмите **Create Virtual Machine**, и при добавлении вспомогательной сетевой карты выберите **SR-IOV** в качестве **Network Type**.
4. Завершите создание виртуальной машины.
5. Подключитесь к виртуальной машине через VNC, вы должны увидеть, что eth1 успешно получил IP-адрес, что свидетельствует об успешной настройке.



Связанные заметки

Конфигурация параметров ядра для виртуальных машин CentOS

После использования виртуальной машиной CentOS сетевой карты SR-IOV необходимо изменить параметры ядра для соответствующей сетевой карты. Конкретные шаги следующие.

1. Откройте терминал и выполните команду для изменения параметров ядра для соответствующей сетевой карты. Замените `<NIC Name>` на фактическое имя NIC.

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

2. Выполните команду для загрузки и применения всех параметров ядра из файла /etc/sysctl.conf, чтобы конфигурация ядра вступила в силу. Если в выводе значение равно 2, значит изменение прошло успешно.

```
sysctl -p
```

   Вывод:

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

Настройка виртуальных машин для использования режима сетевого биндинга с поддержкой IPv6

Режим сетевого биндинга — это механизм расширения плагинов для сетевого взаимодействия виртуальных машин. По умолчанию платформа использует плагин ManagedTap для обеспечения поддержки IPv6 в виртуальных машинах. Этот плагин позволяет виртуальным машинам получать IP-адреса через DHCP Server CNI. Следовательно, пока DHCP Server CNI поддерживает IPv6, виртуальные машины также получают возможность работы с IPv6.

В настоящее время в качестве CNI используется Kube-OVN. Поскольку DHCP Server Kube-OVN полностью поддерживает IPv6, виртуальные машины могут получить полноценную функциональность IPv6 благодаря сочетанию ManagedTap и Kube-OVN.

Содержание

Требования

Процедура

Добавление конфигурации IPv6 в подсеть виртуальной машины

Создание виртуальной машины с использованием режима сетевого биндинга в веб-консоли

Доступ к виртуальной машине через VNC и настройка сетевого интерфейса

Настройка маршрута по умолчанию для IPv6

Требования

- Версия ACP должна быть v4.0.0 или выше.

- В качестве CNI используется Kube-OVN, а подсеть виртуальной машины настроена как Underlay.

Процедура

1 Добавление конфигурации IPv6 в подсеть виртуальной машины

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

Добавьте следующие параметры в раздел `spec`:

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

2 Создание виртуальной машины с использованием режима сетевого биндинга в веб-консоли

При создании виртуальной машины выберите **Network Binding** в качестве сетевого режима.

3 Доступ к виртуальной машине через VNC и настройка сетевого интерфейса

Для систем CentOS отредактируйте файл `/etc/sysconfig/network-scripts/ifcfg-enp1s0` и добавьте следующую конфигурацию:

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

Перезапустите сеть

```
systemctl restart network
```

4 Настройка маршрута по умолчанию для IPv6

Если коммутатор настроен на отправку сообщений Router Advertisement (RA), ручная настройка маршрута не требуется. Маршрут по умолчанию может быть автоматически получен через RA-сообщения от коммутатора.

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

Menu

Хранение данных

Введение

Введение

Преимущества

Руководства

Управление виртуальными дисками

Создание виртуального диска

Монтирование виртуального диска

Расширение виртуального диска

Отмонтирование виртуального диска

Удаление виртуального диска

Введение

ACP Virtualization с KubeVirt Storage предоставляет возможности постоянного хранения для виртуальных машин (VM) за счёт бесшовной интеграции с нативными для Kubernetes механизмами хранения. Он использует **PersistentVolumeClaim** (PVC) для хранения данных дисков VM и применяет **Container Storage Interface** (CSI) для интеграции с различными системами хранения. Кроме того, для инициализации данных дисков VM используется **Containerized Data Importer** (CDI). Основываясь на этих компонентах, платформа расширяет функциональность управления дисками VM, обеспечивая комплексный контроль жизненного цикла.

Содержание

Преимущества

Преимущества

- Удобство использования

  Большинство операций с дисками VM можно легко выполнять через **Web UI**, что минимизирует необходимость владения CLI.

- Управление жизненным циклом дисков VM

  Настройка автоматического удаления дисков VM при завершении работы соответствующей VM.

Menu

Руководства

Управление виртуальными дисками

Создание виртуального диска

Монтирование виртуального диска

Расширение виртуального диска

Отмонтирование виртуального диска

Удаление виртуального диска

Управление виртуальными дисками

Дисковые накопители данных могут использоваться для удовлетворения требований бизнеса по сохранению данных.

Содержание

Создание виртуального диска
- Процедуры

Монтирование виртуального диска
- Процедуры

Расширение виртуального диска
- Процедуры

Отмонтирование виртуального диска
- Процедуры

Удаление виртуального диска
- Процедуры

Создание виртуального диска

Создайте **дисковый накопитель данных** для виртуальной машины. За один раз можно добавить только **один** виртуальный диск; если требуется несколько дисков, повторите эту операцию.

Примечание: Виртуальные диски можно монтировать онлайн, когда виртуальная машина находится в состоянии **запущена**.

Процедуры

1. Перейдите в **Container Platform**.
2. В левой навигационной панели нажмите **Virtualization** > **Virtual Disk**.
3. Нажмите **Create Virtual Disk**.
4. Настройте параметры согласно следующим инструкциям.

Параметр	Описание
Volume Mode	- **File System**: Монтирование диска с доступом к файловой директории. - **Block Device**: Монтирование диска как блочного устройства.
Storage Class	Платформа поддерживает диски виртуальных машин, автоматически создавая и управляя persistent volume claims. Необходимо указать класс хранения, требуемый для динамического создания persistent volume claims. Разные классы хранения поддерживают разные режимы томов. Если для выбранного режима тома нет доступных классов хранения, обратитесь к администратору для добавления.
Delete with VM	Если включено, данные диска будут удалены при удалении виртуальной машины.
Mount	- **Do Not Mount**: Только создать виртуальный диск; его можно смонтировать позже при необходимости. - **Mount to VM**: Выберите целевую виртуальную машину, к которой необходимо примонтировать виртуальный диск.

5. Нажмите **Create**.

Монтирование виртуального диска

Примонтируйте **дисковый накопитель данных** к виртуальной машине, присоединив уже созданный виртуальный диск к целевой виртуальной машине.

Примечание: Виртуальные диски можно монтировать онлайн, когда виртуальная машина находится в состоянии **запущена**.

Процедуры

1. Перейдите в **Container Platform**.
2. В левой навигационной панели нажмите **Virtualization** > **Virtual Disk**.
3. Нажмите ⋮ > **Mount** рядом с виртуальным диском, который нужно примонтировать.
4. Выберите целевую виртуальную машину и нажмите **Mount**.

Расширение виртуального диска

Расширьте **системный диск** и **дисковый накопитель данных**, уже примонтированные к виртуальной машине.

Процедуры

1. Перейдите в **Container Platform**.
2. В левой навигационной панели нажмите **Virtualization** > **Virtual Machine**.
3. Нажмите на имя виртуальной машины, чтобы перейти на страницу **Details**.
4. В разделе **Virtual Disk** найдите диск, который нужно расширить, и нажмите ⋮ > **Expand**.
5. Введите новый размер и нажмите **Expand**.

Отмонтирование виртуального диска

Отмонтируйте **дисковый накопитель данных** от виртуальной машины; диски можно отмонтировать только у виртуальных машин в состоянии **stopped**.

Процедуры

1. Перейдите в **Container Platform**.
2. В левой навигационной панели нажмите **Virtualization** > **Virtual Disk**.
3. Нажмите ⋮ > **Unmount** рядом с виртуальным диском, который нужно отмонтировать, и подтвердите действие.

Удаление виртуального диска

Удаление поддерживается только для виртуальных дисков в отмонтированном состоянии.

Примечание: Системные диски удалять нельзя.

Процедуры

1. Перейдите в **Container Platform**.
2. В левой навигационной панели нажмите **Virtualization** > **Virtual Disk**.
3. Нажмите ⋮ > **Delete** рядом с виртуальным диском, который нужно удалить, и подтвердите действие.

Menu

Резервное копирование и восстановление

Введение

Введение

Сценарии применения

Ограничения использования

Руководства

Использование снимков

Требования

Примечания

Создание снимка

Откат к снимку

Удаление снимка

Введение

ACP Virtualization With Kubevirt предоставляет возможности создания снимков виртуальных машин, позволяя пользователям выполнять резервное копирование и восстановление ВМ с помощью снимков.

Содержание

Сценарии применения

Ограничения использования

Сценарии применения

- Восстановление после сбоев и откат при ошибках

  Когда виртуальная машина теряет данные из-за аппаратных сбоев, ошибок пользователя (например, случайного удаления файлов) или вредоносных атак (например, программ-вымогателей), снимки служат последней линией защиты для восстановления работы.

Ограничения использования

- Для создания снимка необходимо сначала остановить виртуальную машину.
- PVC (Persistent Volume Claim), используемый диском виртуальной машины, должен быть настроен с режимом совместного доступа на нескольких узлах.

Menu

Руководства

Использование снимков

Требования

Примечания

Создание снимка

Откат к снимку

Удаление снимка

Использование снимков

Снимок виртуальной машины сохраняет текущее состояние виртуальной машины и может быть использован для восстановления виртуальной машины в этом состоянии в случае неожиданного сбоя.

Содержание

Требования

Примечания

Создание снимка

Процедура

Откат к снимку

Примечания

Процедура

Удаление снимка

Примечания

Процедура

Требования

- **Volume Snapshot** был развернут администратором в управлении платформой.
- Снимки виртуальной машины основаны на volume snapshots. Убедитесь, что хотя бы один диск привязан к storage class, поддерживающему volume snapshots, например, встроенному хранилищу CephFS.

- Поддерживаются только офлайн-снимки виртуальной машины. Пожалуйста, сначала остановите виртуальную машину перед созданием или откатом к снимку.

Примечания

Если в кластере присутствует несколько однотипных типов хранилищ, например, подключено несколько различных источников Ceph RBD, функциональность снимков дисков может работать некорректно при использовании виртуальной машиной такого хранилища.

Создание снимка

В снимок виртуальной машины включается: настройки виртуальной машины и состояние дисков, поддерживающих volume snapshots.

Процедура

1. Зайдите в **Container Platform**.
2. В левой навигационной панели выберите **Virtualization** > **Virtual Machines**.
3. Найдите виртуальную машину и нажмите ⋮ > **Create Snapshot**.
4. Заполните описание снимка. Описание поможет задокументировать текущее состояние виртуальной машины, например `Initial Installation`, `Before Application Upgrade`.
5. Нажмите **Create**. Время создания снимка зависит от сетевых условий и нагрузки, пожалуйста, подождите.
6. Проверьте статус снимка.
   - Когда статус снимка изменится на `Ready`, это означает успешное создание.

- Если снимок долго остается в статусе `Not Ready`, нажмите > Просмотрите причины и устраните неполадки, затем создайте снимок заново.

Откат к снимку

Откатит настройки виртуальной машины и диски, поддерживающие volume snapshots, к состоянию на момент создания снимка. Например, диски, добавленные после создания снимка, будут удалены; изменённые данные дисков будут восстановлены.

Примечания

Если диски привязаны к storage class, поддерживающему механизм LVM (например, TopoLVM), пожалуйста, подтвердите у администратора, что политика восстановления для этого storage class установлена в **Retain** (`reclaimPolicy: Retain`), чтобы корректно использовать функцию отката снимка.

Процедура

1. Зайдите в **Container Platform**.
2. В левой навигационной панели выберите **Virtualization** > **Virtual Machines**.
3. Нажмите на ***Virtual Machine Name***.
4. Во вкладке **Snapshots** найдите нужный снимок и нажмите ⋮ > **Rollback**.
5. Ознакомьтесь с информацией в подсказке и нажмите **Rollback** после подтверждения правильности данных.

   Примечание: Операция отката не может быть прервана или отменена, будьте внимательны.
6. Нажмите на имя снимка, чтобы проверить в разделе «Snapshot Rollback Records», завершён ли откат. Время отката зависит от сетевых условий и нагрузки, пожалуйста, подождите.

Описание

- Если откат не удался, состояние виртуальной машины останется без изменений. Вы можете запустить виртуальную машину в обычном режиме или попытаться выполнить откат снимка снова.
- Если виртуальная машина была запущена во время процесса отката, она вернётся в состояние до остановки, а при следующей остановке продолжит откат к состоянию на момент создания снимка.
- Чтобы избежать конфликтов в операциях, убедитесь, что последний откат завершён, прежде чем выполнять другие действия с этой виртуальной машиной.

Удаление снимка

Удаляйте ненужные снимки виртуальной машины для освобождения ресурсов диска.

Примечания

При удалении снимка виртуальной машины, к которому был выполнен откат, если для диска виртуальной машины требуется копирование данных на основе снимка (например, TopoLVM), необходимо дождаться запуска виртуальной машины на основе версии после отката перед удалением, иначе виртуальная машина не сможет запуститься.

Процедура

1. Зайдите в **Container Platform**.
2. В левой навигационной панели выберите **Virtualization** > **Virtual Machines**.
3. Нажмите на ***Virtual Machine Name***.
4. Во вкладке **Snapshots** найдите нужный снимок и нажмите ⋮ > **Delete**.
5. Ознакомьтесь с информацией в подсказке и нажмите **Delete** после подтверждения правильности данных.