

# Виртуализация

---

## Обзор

### Введение

Решение виртуальных машин с оркестрацией контейнеров

Особенности

Функциональные возможности продукта

Ограничения и условия

---

## Установка

### Установка

Предварительные требования

Процедура

Объяснение квоты ресурсов

---

## Образы

---

[Введение](#)

Преимущества

[Руководства](#)

[Разрешения](#)

[Как сделать](#)

---

## Виртуальная машина

[Введение](#)

[Устранение неполадок](#)

[Руководства](#)

[Как сделать](#)

---

## Сеть

[Введение](#)

Преимущества

[Руководства](#)

[Как сделать](#)

---

## Хранение

[Введение](#)

Преимущества

[Руководства](#)

# Резервное копирование и восстановление

## Введение

Сценарии применения

Ограничения использования

## Руководства

# Обзор

## Введение

Решение виртуальных машин с оркестрацией контейнеров

Особенности

Функциональные возможности продукта

Ограничения и условия

# Введение

Для предприятий, использующих архитектуру на базе виртуальных машин, переход к архитектуре на базе 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 конфигурировать, 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

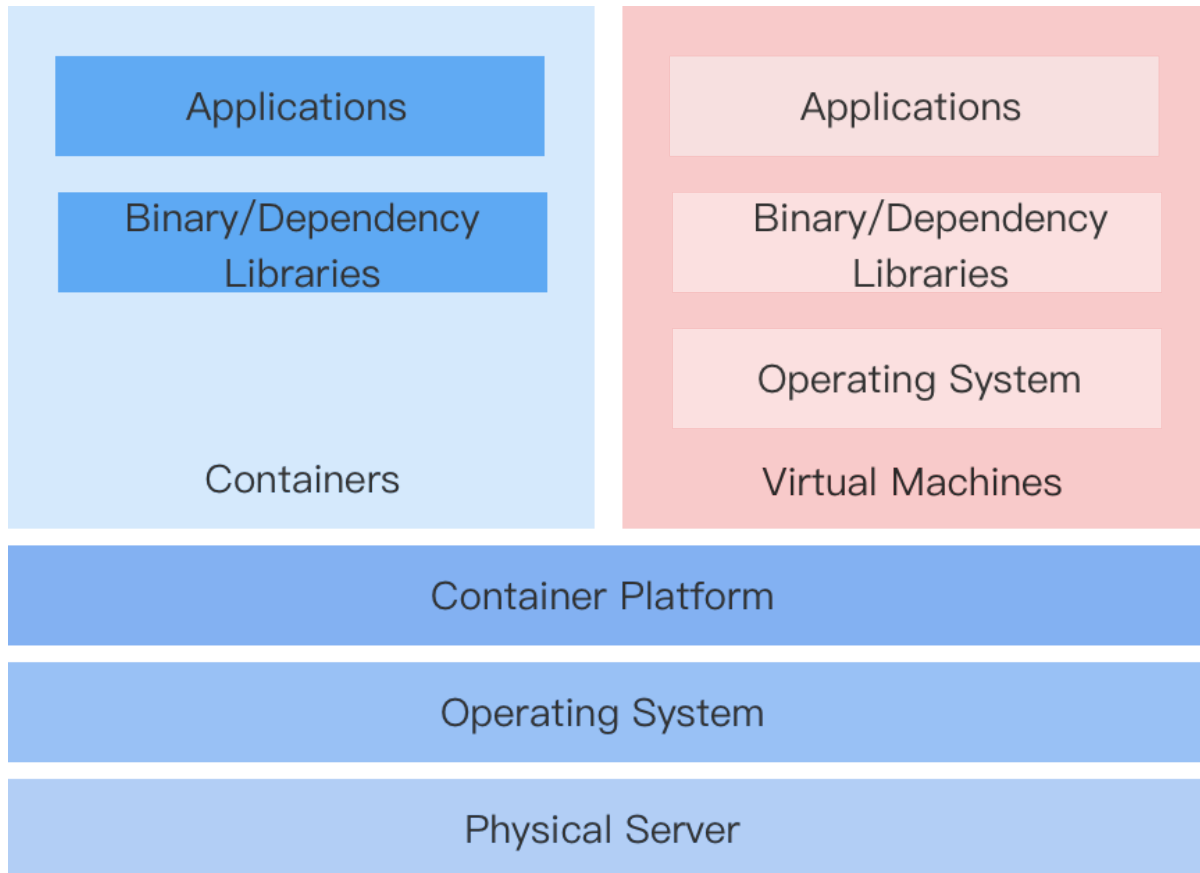
Ограничения и условия

---

## Решение виртуальных машин с оркестрацией контейнеров

Данная платформа реализует решение виртуальной машины (VMI, VirtualMachineInstance) на основе open-source 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 с этим образом на платформе, что позволит развернуть приложение на контейнерной платформе.

### Сохранение привычек поведения

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, одновременно обеспечивая планирование и управление контейнерами на базе Kubernetes.
- При сохранении рабочих нагрузок виртуальных машин платформа 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 их управления, 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 выполняются администратором платформы.

# Установка

Для того чтобы персонал проекта мог полноценно использовать возможности 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 Operator

Создание экземпляра HyperConverged

Настройка коэффициента overcommit виртуальных машин (необязательно)

Важные замечания

Объяснение квоты ресурсов

## Предварительные требования

- **Скачать** установочный пакет **ACP Virtualization with KubeVirt**, соответствующий архитектуре вашей платформы.
- **Загрузить** установочный пакет **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 ToroLVM 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
<b>ToroLVM</b>	<p><b>Преимущества:</b> Относительно лёгкий и с хорошей производительностью.</p> <p><b>Недостатки:</b> Не поддерживает использование между узлами, низкая надёжность, отсутствие избыточности.</p>
<b>Ceph Distributed Storage</b>	<p><b>Преимущества:</b> Поддерживает использование между узлами, высокая доступность и избыточность.</p> <p><b>Недостатки:</b> Избыточные копии дисков снижают эффективность использования; производительность ниже.</p>

- Если используется ToroLVM и настроено несколько дисков, убедитесь, что оставшаяся ёмкость на узлах с включённой виртуализацией может покрыть суммарную ёмкость 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 в самосозданном кластере 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 этом узле виртуальные машины.

## Порядок действий

1. Войдите в **Administrator**.
2. В левой навигационной панели нажмите **Cluster Management > Clusters**.
3. Нажмите на **Self-Built Cluster Name**.
4. На вкладке **Nodes** нажмите  $\vdots$  справа от узла, где нужно установить переключатель виртуализации > **Enable Virtualization**.
5. Нажмите **Confirm**.

2

## Развёртывание Operator

1. Войдите в систему, перейдите на страницу **Administrator**.
2. Нажмите **Marketplace > OperatorHub**, чтобы перейти 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
<b>Channel</b>	Канал по умолчанию — <code>alpha</code> .
<b>Installation Mode</b>	<code>Cluster</code> : Все пространства имён в кластере используют один экземпляр Operator для создания и управления, что снижает использование ресурсов.

Параметр	Рекомендуемая конфигурация
Installation Place	Выберите <code>Recommended</code> , Namespace поддерживает только <code>kubevirt</code> .
Upgrade Strategy	<code>Manual</code> : При появлении новой версии в Operator Hub требуется ручное подтверждение для обновления Operator до последней версии.

3

## Создание экземпляра HyperConverged

1. Войдите в **Administrator**.
2. Нажмите **Marketplace > OperatorHub**.
3. Найдите **ACP Virtualization with KubeVirt**, нажмите на него, чтобы перейти 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 корректный **адрес репозитория образов виртуальных машин**, например: `192.168.16.214:60080`, остальные параметры оставьте по умолчанию.

```
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)

- Настройте коэффициент `overcommit` для кластера, в котором находятся виртуальные машины, в разделе **Cluster Management > Clusters**.
- Или настройте коэффициент `overcommit` для пространства имён, в котором расположены виртуальные машины, в разделе **Project Management > Namespaces**.

### Важные замечания

- Виртуальные машины поддерживают только коэффициент `overcommit 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 VM`**, что делает пользовательские запросы, заданные через YAML, неэффективными.

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

## Объяснение квоты ресурсов

Квота памяти для виртуальных машин ограничивается квотой памяти пространства имён, в котором они находятся. Поскольку память Pod, hostящего 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 увеличить ресурсы.



# Образы

## Введение

### Введение

Преимущества

## Руководства

### Добавление образов виртуал Обновление/Удаление образо Обновлени

Процедура

## Как сделать

### Создание образов Windows н Создание образов Linux на ос Экспорт об

Предварительные требования

Ограничения и особенности

Процедура

Предварительные требования

Ограничения и особенности

Процедура

Процедура

Удаленный доступ

---

## 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), позволяющий пользователям легко импортировать образы виртуальных машин, хранящиеся в удалённых репозиториях, в ACP для использования.

## Содержание

[Преимущества](#)

## Преимущества

- **Поддержка основных операционных систем**  
Поддерживает различные широко используемые дистрибутивы Linux и операционные системы Windows.
- **Поддержка нескольких архитектур**  
Совместима с архитектурами **X86\_64** и **ARM64**.
- **Поддержка нескольких источников**  
Позволяет импортировать образы виртуальных машин из:
  - регистров образов
  - файловых серверов
  - объектного хранилища, совместимого с S3

- Поддержка нескольких форматов

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

# Руководства

[Добавление образов виртуал](#) [Обновление/Удаление образо](#) [Обновлени](#)

Процедура

# 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 раздел **Administrator**.
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
<b>Provisioning Method</b>	В настоящее время поддерживается только метод <b>Public Image</b> , то есть добавленный образ может использоваться в назначенных проектах.

Параметр	Описание
<b>Operating System</b>	<p>Поддерживаемые операционные системы:  <b>CentOS/Ubuntu/RedHat/Debian/TLinux/Other Linux/Windows (Alpha).</b></p> <p>Поддерживаемые архитектуры систем: <b>X86_64</b> и <b>ARM64 (Alpha).</b></p>
<b>Source</b>	<ul style="list-style-type: none"> <li>• <b>Image Repository:</b> Образы виртуальных машин, хранящиеся в репозитории контейнерных образов.</li> <li>• <b>HTTP:</b> Образы виртуальных машин, хранящиеся на файловом сервере с использованием протокола HTTP.</li> <li>• <b>Object Storage (S3):</b> Образы виртуальных машин, доступные через протокол Object Storage (S3). Если аутентификация не требуется, рекомендуется использовать HTTP в качестве источника.</li> </ul>
<b>CPU Architecture</b>	<p>Укажите архитектуру CPU. Для источников из репозитория образов поддерживается множественный выбор; для других источников — только одиночный выбор.</p>
<b>Image Address</b>	<p>Поддерживаются образы виртуальных машин KVM, включая форматы qcow2/raw.</p> <ul style="list-style-type: none"> <li>• Если источник — репозиторий образов, введите <code>repository_address:image_version</code>, например, <code>registry.example.com/library/ubuntu:latest</code>.</li> <li>• Если источник — HTTP, введите URL файла образа, который должен начинаться с <code>http://</code> или <code>https://</code>, например, <code>http://192.168.0.1/vm_image/centos_7.8.qcow2</code>.</li> <li>• Если источник — Object Storage (S3), введите адрес образа, доступный через протокол Object Storage (S3), например, <code>https://endpoint/bucket/centos.qcow2</code>.</li> </ul>

Параметр	Описание
<b>Authentication</b>	<p>В зависимости от необходимости аутентификации в репозитории образов можно включить или отключить переключатель. Если включено, можно выбрать из существующих учетных данных образа или нажать <b>Add Credentials</b>, поддерживаются только учетные данные типа <b>Username/Password</b>.</p> <p><b>Примечание:</b> При источнике Object Storage (S3) отключить аутентификацию нельзя.</p>
<b>Assigned Project</b>	<p>Назначьте права использования этого образа проектам.</p> <ul style="list-style-type: none"><li>• All Projects: Назначить права использования образа всем проектам.</li><li>• Specific Project: Назначить права использования образа конкретному проекту.</li><li>• No Assignment: Пока не назначать ни одному проекту. После создания образа можно назначить через операцию <b>Update Image</b>.</li></ul>

5. Нажмите **Add**.

# Обновление/Удаление образов виртуальных машин

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

# Обновление/удаление учетных данных образа

1. Перейдите в **Administrator**.
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**.

# Как сделать

## Создание образов Windows н

Предварительные требования

Ограничения и особенности

Процедура

Удаленный доступ

## Создание образов Linux на ос

Предварительные требования

Ограничения и особенности

Процедура

## Экспорт об

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 особенности

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 согласно вашей реальной среде.

## Ограничения и особенности

- При запуске KubeVirt размер файловой системы пользовательского образа повлияет 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 контейнерный 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 контейнерного 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 файла `Containerfile`.

```
touch Containerfile
```

3. Отредактируйте Containerfile, добавьте следующий контент и сохраните.

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

4. Выполните команду для сборки контейнерного образа.

```
podman build -t build-harbor.example.cn/3rdparty/vmdisks/winiso:2019 -f Containerfile . # Замените репозиторий согласно вашей реальной среде
```

5. Выполните команду для отправки образа в репозиторий.

```
podman push build-harbor.example.cn/3rdparty/vmdisks/winiso:2019 #  
Замените репозиторий согласно вашей реальной среде
```

## Создание контейнерного образа из ISO virtio-win

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

```
touch Containerfile
```

2. Отредактируйте Containerfile, добавьте следующий контент и сохраните.

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

3. Выполните команду для сборки контейнерного образа.

```
podman build -t build-harbor.example.cn/3rdparty/vmdisks/win-virtio:latest -f Containerfile . # Замените репозиторий согласно вашей реальной среде
```

4. Выполните команду для отправки образа в репозиторий.

```
podman push build-harbor.example.cn/3rdparty/vmdisks/win-virtio:la
test # Замените репозиторий согласно вашей реальной среде
```

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/win-  
so:2019 # Замените образ согласно вашей реальной среде  
  name: containerdisk  
- containerDisk:  
  image: registry.example.cn:60070/3rdparty/vmdisks/win-  
virtio # Замените образ согласно вашей реальной среде  
  name: virtio
```

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



```
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$0v1hl57e$0rawYwaeu9jL6hBf3XP9L
k6XXaMUS9/W6LPbWRinUoXujo39lP3l98V0c00btr.LDoAv/y1m85FLQmxwNlWFe/
      name: cloudinitdisk
  - containerDisk:
      image: registry.example.cn:60070/3rdparty/vmdisks/winiso:2019 # Замените образ согласно вашей реальной среде
      name: containerdisk
  - containerDisk:
      image: registry.example.cn:60070/3rdparty/vmdisks/win-virtio # Замените образ согласно вашей реальной среде
      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 появляется 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 поочередно выбрать каждый раздел и нажать **Delete** для удаления всех разделов, чтобы система могла обработать это 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 установки. Часть <version> должна соответствовать вашей версии.

3. После завершения установки выключите систему Windows.

5

## Экспорт пользовательского образа Windows

Пожалуйста, ознакомьтесь с [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 ОС, например Windows 11, включите такие функции, как часы, UEFI, TPM и т.д. 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
        vpic: {}
      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 с деталями.
4. Нажмите на иконку **Add** рядом с **Internal Route** в области **Login Information**.
5. Настройте параметры согласно следующим инструкциям.

Параметр	Описание
Type	Выберите <b>NodePort</b> .

Параметр	Описание
Port	<ul style="list-style-type: none"><li>• Протокол: выберите TCP.</li><li>• Порт сервиса: используйте 3389.</li><li>• Порт виртуальной машины: используйте 3389.</li><li>• Имя порта сервиса: rdp.</li></ul>

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

## Удаленный доступ

В данном документе рассматривается пример удаленного подключения 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 основе открытого компонента KubeVirt. Используется технология виртуализации KubeVirt для создания образа операционной системы Linux из ISO-образа.

## Содержание

### [Предварительные требования](#)

Ограничения и особенности

Процедура

Конвертация Linux ISO-образа в контейнерный 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 в качестве примера используется репозиторий `build-harbor.example.cn`; замените его в соответствии с вашей реальной средой.

## 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 меньше 100G для минимизации начального размера. После настройки `cloud-init` при создании виртуальной машины выделяйте больший объем хранилища для корневого раздела, система автоматически расширит его.

## Процедура

### 1 Конвертация Linux ISO-образа в контейнерный 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 файл `Containerfile`, выполнив команду:

```
touch Containerfile
```

3. Отредактируйте `Containerfile`, добавьте следующий контент и сохраните файл.

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

4. Постройте контейнерный образ, выполнив команду:

```
podman build -t build-harbor.example.cn/3rdparty/vmdisks/ubuntu-iso:24.04 . # Замените репозиторий в соответствии с вашей средой
```

5. Отправьте образ в репозиторий, выполнив команду:

```
podman push build-harbor.example.cn/3rdparty/vmdisks/ubuntu-iso:24.04 # Замените репозиторий в соответствии с вашей средой
```

2

## Создание виртуальной машины

1. Войдите в **Container Platform**.
2. В левой навигационной панели выберите **Virtualization > Virtual Machines**.
3. Нажмите **Create Virtual Machine**.
4. Заполните параметры на странице формы следующим образом. Для подробного описания параметров и настроек смотрите [Create Virtual Machine](#).

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

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 следующий контент в поле `spec.template.spec.volumes`.

```
- containerDisk:
  image: registry.example.cn:60070/3rdparty/vmdisks/ubuntu-iso:24.04 # Замените образ в соответствии с вашей средой
  name: containerdisk
```

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



```
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 в соответствии с вашей средой
      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 для входа в программу установки Windows; если сообщения не видно, 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.

1. После входа на страницу установки следуйте инструкциям мастера установки. В данном документе приведён пример установки Ubuntu; конфигурационные пункты при установке других ОС обычно схожи, поэтому подробно не рассматриваются. Ниже приведены некоторые пояснения к настройкам.

Конфигурация	Описание
Тип установки	Рекомендуется использовать минимальную установку для минимизации размера образа.
Конфигурация хранилища	<p>Выберите пользовательское хранилище.</p> <p>Отформатируйте диск в формат ext4 или xfs и смонтируйте его в корневой раздел (/).</p> <p><b>Примечание:</b> не используйте LVM для разметки диска (Create volume group (LVM)).</p>
Настройка SSH	Выберите установку инструментов OpenSSH для доступа по SSH.

2. Дождитесь завершения установки.

4

## Изменение YAML-файла

1. Войдите в **Container Platform**.
2. В левой навигационной панели выберите **Virtualization > Virtual Machines**.
3. Нажмите на **Имя виртуальной машины** в списке для перехода на страницу деталей.
4. Нажмите **Stop**.
5. В правом верхнем углу нажмите **Actions > Update**.
6. Переключитесь в режим YAML.
7. Убедитесь, что у диска с именем **rootfs** в `spec.template.spec.domain.devices.disks` значение `bootOrder` равно 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 конфигурационные файлы могут незначительно отличаться в разных операционных системах; настройте их в соответствии с вашей средой.

1. Войдите в операционную систему, используя имя пользователя и пароль.
2. Переключитесь на пользователя `root`.
3. Установите необходимые пакеты.

- Для CentOS-серии выполните команду:

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

- Для Debian-серии 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 по паролю  
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 `/dev/vda1`.
- `[xfs_growfs, /dev/vda1]` # Команда `xfs_growfs` расширяет файловую систему XFS до полного размера раздела. `/dev/vda1` – раздел, который расширяется.

3. Сохраните изменения.

8. После завершения настроек 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 должны выполняться на управляющем узле кластера, где находится виртуальная машина.

### 1 Остановка виртуальной машины

1. Перейдите в **Administrator**.
2. В левой навигационной панели нажмите **Virtualization Management > Virtual Machines**.

3. Нажмите на имя виртуальной машины, системный образ которой необходимо экспортировать, после чего вы будете перенаправлены на страницу с деталями виртуальной машины в Container Platform.
4. Нажмите **Stop**.

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 может быть увеличено при необходимости.

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 похожее сообщение, это означает успешное создание.

```
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
```

Сообщение:

NAME	SOURCEKIND	SOURCENAME	PHASE
export-k1	VirtualMachine	k1	Ready

5. Когда поле PHASE в сообщении изменится на Ready, нажмите ctrl (control) + c, чтобы остановить операцию наблюдения.

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 экспорта виртуальной 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 образа диска 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 скачанный 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`: пользовательские `ak` и `sk` S3-сервиса хранения, используемые для аутентификации; для получения дополнительной информации обратитесь к [MiniO Object Storage](#) <sup>↗</sup>.

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 **Administrator**.
2. В левой навигационной панели нажмите **Virtualization Management > Virtual Machine Images**.
3. Нажмите **Add Virtual Machine Image**.
4. В поле адреса образа укажите `<ENDPOINT>/vmdisks/disk.img.gz`, заменив часть `<ENDPOINT>` на адрес S3-сервиса хранения. Для объяснения других параметров обратитесь к [Adding Virtual Machine Images](#).
5. Нажмите **Add**.

# Разрешения

Функция	Действие	Platform Administrator	Platform auditors	Project Manager
	Просмотр	✓	✓	✓
virtualmachineimagetemplates	Создать	✓	✗	✗
асп- virtualmachineimagetemplates	Обновить	✓	✗	✗
	Удалить	✓	✗	✗

# Виртуальная машина

## Введение

[Введение](#)

## Руководства

### Создание виртуальных маши

Предварительные требования

Примечания

Create Virtual Machine

Create Virtual Machine Group

### Пакетные операции с виртуал

Процедура

### Вход в вир

Процедура

### Управление виртуальными машинами

Сброс пароля

### Мониторинг и оповещения

Мониторинг

Оповещения

### Быстрый п

Добавление сервиса

Переустановка операционной системы

Требования

Процедура

## Настройка IP

## Как сделать

### Настройка проброса USB-устройств

Обзор функции

Сценарии использования

Требования

Шаги

Результат работы

Дополнительная информация

### Горячая миграция виртуальной машины

Overview

Ограничения и условия

Предварительные условия

Шаги выполнения

### Восстановление виртуальной машины

Шаги для вып...

### Клонирование виртуальной машины

Убедитесь в ве...

### Подготовка среды для физического GPU Раздел 1

### Настройка высокой доступности для виртуальных машин

Overview

Glossary

Component Overview

Flow of events during fencing and remediation

Procedure

### Создание и настройка виртуальной машины

Предварительные условия

Процедура

## Устранение неполадок

## Миграция Pod и восстановление работы виртуальных машин

## Сообщения об ошибках горячей миграции

Описание проблемы

Анализ причины

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 полноценные 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.

# Руководства

## Создание виртуальных маши

Предварительные требования

Примечания

Create Virtual Machine

Create Virtual Machine Group

## Пакетные операции с виртуал

Процедура

## Вход в вир

Процедура

## Управление виртуальными машинами

Сброс пароля

## Мониторинг и оповещения

Мониторинг

Оповещения

## Быстрый п

Добавление сервиса

Переустановка операционной системы

Настройка IP

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, 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 через **имя пользователя/пароль**, заданные 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 помощью формы, также вы можете переключиться на YAML-формат 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
<b>Tags</b>	Используются для выбора объектов и поиска коллекций объектов, соответствующих определённым критериям. Должны быть парой ключ-значение, например: <code>app.kubernetes.io/name: hello-app</code> .
<b>Annotations</b>	Используются для предоставления любой информации командам разработки и эксплуатации. Должны быть парой ключ-значение, например: <code>cpaas.io/maintainer: kim</code> .

##### 5. Установите тип машины и выберите образ виртуальной машины.

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

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

Параметр	Описание
<b>Disk Name</b>	Имя диска хранения; имя системного диска изменить нельзя.
<b>Type</b>	<ul style="list-style-type: none"> <li>• <b>Root Disk:</b> система автоматически создаёт системный диск rootfs типа VirtIO для хранения ОС и данных.</li> <li>• <b>Data Disk:</b> нажмите для добавления нескольких дисков данных для постоянного хранения данных. По умолчанию устройство VirtIO.</li> </ul> <p><b>Примечание:</b> имена дисков данных не должны дублировать существующие имена дисков.</p>
<b>Volume Mode</b>	<ul style="list-style-type: none"> <li>• <b>File System:</b> монтировать диск как файловую систему.</li> <li>• <b>Block Device:</b> монтировать диск как блочное устройство.</li> </ul>
<b>Storage Class</b>	<p>Платформа управляет дисками виртуальных машин, создавая и управляя persistent volume claims. Необходимо указать storage class для динамического создания persistent volume claims.</p> <p>Разные storage class поддерживают разные volume mode; если для выбранного volume mode нет доступного storage class, обратитесь к администратору для добавления.</p>
<b>Capacity</b>	Требуемая ёмкость для хранения виртуальной машины; минимальный размер системного диска — 20 ГБ.
<b>Delete with VM</b>	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
<b>IP Address</b>	<ul style="list-style-type: none"> <li>• По умолчанию <b>Dynamic (DHCP)</b>; IP-адрес динамически назначается при запуске виртуальной машины и освобождается при её</li> </ul>

Параметр	Описание
	<p>остановке.</p> <ul style="list-style-type: none"> <li>Если привязан <b>Static IP</b>, виртуальная машина всегда будет использовать этот IP-адрес даже после перезапуска. Если в текущем проекте нет доступных IP, сначала освободите IP.</li> </ul>
<b>Network Mode</b>	<ul style="list-style-type: none"> <li><b>Bridged</b>: виртуальная машина использует тот же IP-адрес, что и контейнерная группа, и общается с внешним миром через этот IP.</li> <li><b>NAT</b>: виртуальной машине назначается внутренний IP, но для внешней связи он транслируется в IP контейнерной группы.</li> </ul> <p>Открытые порты указывают порты виртуальной машины, например, порт SSH 22; если <b>Open Ports</b> не заполнены, считаются открытыми все порты.</p>
<b>Auxiliary Network Card</b>	<p>Добавьте вспомогательные сетевые карты по необходимости.</p> <p><b>Примечание:</b></p> <ul style="list-style-type: none"> <li>Если требуется функция вспомогательных сетевых карт или нет доступных типов сетей для вспомогательных карт, обратитесь к администратору платформы для настройки.</li> <li>Типы SR-IOV поддерживаются только для Linux на архитектуре x86_64.</li> <li>По умолчанию IP-адреса получают через DHCP.</li> <li>После нескольких перезагрузок SR-IOV виртуальные машины могут иметь два разных VF с одинаковым MAC-адресом.</li> </ul>

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

Параметр	Описание
<b>Keys</b>	<p>Всегда используйте SSH-ключи для проверки удалённого входа. Этот метод не требует проверки пароля; рекомендуется входить в</p>

Параметр	Описание
	<p>виртуальную машину с помощью ключей.</p> <ul style="list-style-type: none"> <li>• Вы можете использовать уже имеющиеся ключи на платформе или создать новые; все ключи доступны на странице <b>Virtualization &gt; Key Pairs</b>.</li> <li>• Доступ по SSH к виртуальной машине возможен только у тех, у кого есть приватный ключ. Если несколько человек обслуживают виртуальную машину, можно связать несколько ключей и выдать приватные ключи разным пользователям. При утечке ключа связанный ключ можно быстро отозвать для минимизации ущерба.</li> <li>• Публичный ключ SSH хранится на платформе в конфиденциальном виде; приватный ключ не хранится, поэтому храните его в безопасности самостоятельно.</li> <li>• Пожалуйста, обратитесь к документации соответствующей ОС для пароля пользователя root.</li> </ul>
<b>Password</b>	<p>Используйте пользователя и пароль операционной системы для проверки входа, позже можно переключиться на метод с ключами.</p> <ul style="list-style-type: none"> <li>• Пользователь — это только начальная учётная запись; после успешного создания виртуальной машины можно создавать других пользователей ОС для входа.</li> <li>• Платформа шифрует и хранит пароль пользователя root, и вы не увидите его в открытом виде, поэтому храните его в безопасности самостоятельно.</li> </ul>
<b>Start Immediately</b>	<p>По умолчанию включено. При включении виртуальная машина запускается сразу после создания, иначе создаётся только сама виртуальная машина.</p>


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

Параметр	Описание
<b>Health Check</b>	<ul style="list-style-type: none"> <li>• <b>Liveness Check:</b> проверяет, находится ли виртуальная машина в здоровом состоянии; при обнаружении проблем решается, перезапускать ли экземпляр согласно настройкам проверки здоровья.</li> <li>• <b>Availability Check:</b> проверяет, завершён ли запуск виртуальной машины и находится ли она в нормальном рабочем состоянии; при обнаружении проблем статус виртуальной машины обновляется.</li> </ul> <p>Для описания связанных параметров.</p>
<b>Node Affinity</b>	<ul style="list-style-type: none"> <li>• <b>Preferred:</b> 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).</li> <li>• <b>Required:</b> виртуальная машина будет запланирована только на узлы, полностью соответствующие требованиям affinity.</li> </ul>

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 на **Actions** в правом верхнем углу страницы деталей для обновления или удаления виртуальной машины по необходимости. Для других операций, таких как сброс пароля или обновление ключей, пожалуйста, обратитесь к [Manage Virtual Machines](#).

### Примечание:

- Обновления возможны только при статусах виртуальной машины **Abnormal**, **Unknown** или **Stopped**.

- Обновления не поддерживают 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 помощью формы, также вы можете переключиться на YAML-формат 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
<b>Number of Instances</b>	Количество виртуальных машин, создаваемых группой виртуальных машин.
<b>Anti-Affinity between Instances</b>	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.
<b>Tags</b>	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: <code>app.kubernetes.io/name: hello-app</code> .

5. В области **Virtual Machine Template** настройте единые теги, аннотации, спецификации, образы, хранилище и другую информацию для всех виртуальных машин группы, следуя инструкции в разделе [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** для виртуальной машины 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 состояний ниже, операция считается успешной.

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

# Вход в виртуальную машину с помощью 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.

```
Send remote command v
CentOS Linux 7 (Core)
Kernel 3.10.0-1160.11.1.el7.x86_64 on an x86_64
changaji login:
```

**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 нажмите **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 нажмите **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 нажмите **Virtualization > Key Pairs**.
3. Найдите **Key Pair Name**, нажмите **:** > **Delete** и подтвердите действие.

# Управление виртуальными машинами

---

## Содержание

### Сброс пароля

Процедура

Обновление ключа

Процедура

Обновление характеристик

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 используется для входа в виртуальную машину при входе с использованием пароля.

## 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 или **Create Key**.
5. Выберите, нужно ли перезапустить виртуальную машину сразу; обновление ключей требует перезапуска для вступления изменений в силу.
6. Нажмите **Update**.

## Обновление характеристик

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 сценариями платформы или индивидуальными потребностями.
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, пожалуйста, 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, и вы можете открывать 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 использован текущий образ операционной системы. Если вы хотите переустановить новую ОС, сначала выберите ОС образа виртуальной машины, затем выберите образ виртуальной машины, принадлежащий этой ОС.

5. Нажмите **Reinstall**.

## Настройка IP

Назначьте IP виртуальной машине с помощью динамического выделения (DHCP) или привяжите фиксированный IP к виртуальной машине; 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 оповещение виртуальных машин по показателям CPU, памяти, хранилища и сети. Для своевременного информирования также можно настроить политики уведомлений.

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

---

## Содержание

### Мониторинг

#### Оповещения

- Настройка политик оповещений

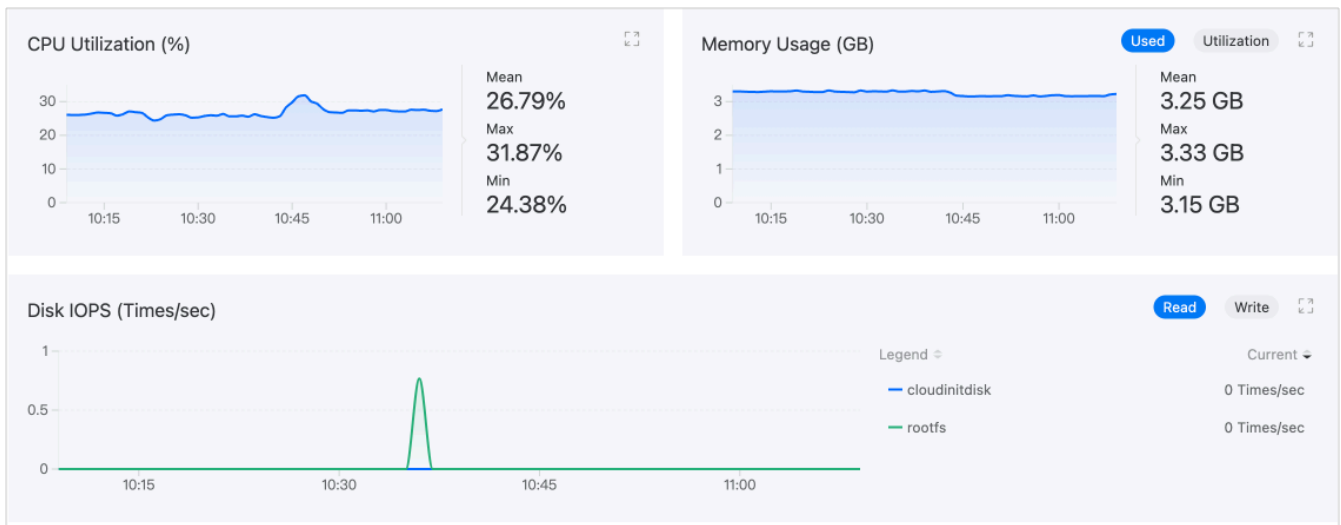
- Обработка оповещений

- Привязка политик уведомлений

---

## Мониторинг

По умолчанию платформа собирает часто используемые метрики мониторинга производительности виртуальных машин, включая CPU, память, хранилище и сеть. Перейдите в **Virtualization > Virtual Machines**, и на вкладке **Monitoring** в деталях виртуальной машины вы можете 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 мониторить, условия, при которых необходимо оповещать, 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	<ul style="list-style-type: none"> <li>- Metric Alert: Объект мониторинга — предопределённая платформой метрика, например <i>Memory Usage Rate</i>.</li> <li>- Event Alert: Объект мониторинга — причина события, то есть причина перехода виртуальной машины в текущее состояние, например BackOff, Pulling, Failed.</li> </ul>
Условие срабатывания	<p>Состоит из операторов сравнения, порогов оповещения и длительности. Сравнивая результаты мониторинга в реальном времени с установленными порогом, определяется необходимость оповещения.</p> <p>Если задана длительность, платформа также сравнивает время, в течение которого объект мониторинга находится в состоянии оповещения.</p>

Параметр	Описание
<b>Уровень оповещения</b>	<ul style="list-style-type: none"> <li>- 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.</li> <li>- Warning: У объекта мониторинга есть операционные риски, которые при отсутствии своевременного решения могут повлиять на нормальную работу бизнеса. Например, если использование CPU превышает 80% в течение 3 минут.</li> <li>- Serious: У объекта мониторинга выявлены известные проблемы, которые могут привести к сбоям функциональности платформы и повлиять на нормальную работу бизнеса.</li> <li>- Disaster: Объект мониторинга вышел из строя, что привело к прерыванию сервисов платформы, потере данных и значительному воздействию.</li> </ul>

**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 статуса оповещений, пожалуйста, оперативно их обработайте.

## Привязка политик уведомлений

Помимо оповещений в реальном времени на вкладке **Alerts**, платформа также поддерживает отправку информации об оповещениях по электронной почте, SMS и другим каналам 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 такие операции, как масштабирование или устранение неполадок, тем самым повышая эффективность работы.

## Содержание

[Требования](#)

[Процедура](#)

## Требования

Убедитесь, что функция виртуализации включена для текущего кластера перед использованием. Пожалуйста, обратитесь к разделу [Install](#).

## Процедура

1. Перейдите в раздел **Administrator**.
2. В левой панели навигации нажмите **Virtualization Management > Virtual Machines**.
3. Выберите **Cluster**, чтобы просмотреть список виртуальных машин в этом кластере.
4. Вы можете быстро найти виртуальную машину по 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.

# Как сделать

## Настройка проброса USB-устройств

Обзор функции

Сценарии использования

Требования

Шаги

Результат работы

Дополнительная информация

## Горячая миграция виртуальных машин

Overview

Ограничения и условия

Предварительные условия

Шаги выполнения

## Восстановление виртуальной машины

Шаги для выполнения

## Клонирование виртуальной машины

Убедитесь в выполнении условий

## Подготовка среды для физического GPU Раздел

## Настройка высокой доступности для виртуальных машин

Overview

Glossary

Component Overview

Flow of events during fencing and remediation

Procedure

## Создание и настройка виртуальной машины

Предварительные условия

Процедура

# Настройка проброса USB-устройств хоста

---

## Содержание

### Обзор функции

Сценарии использования

Требования

Шаги

- Открытие USB-устройств

- Назначение USB-устройств виртуальной машине

Результат работы

Дополнительная информация

- 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.

## Требования

- Версия платформы должна быть не ниже v3.18.

## Шаги

### 1 Открытие USB-устройств

Чтобы назначить USB-устройство виртуальной машине, 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`, а `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:

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

### Совет

Убедитесь, что виртуальная машина остановлена перед редактированием конфигурации.

## Результат работы

После завершения настройки выполните команду `lsusb` внутри виртуальной машины. Если в выводе отображается USB-устройство хост-узла, проброс выполнен 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 80
2.11a/b/g/n/ac 2T2R DB WLAN Adapter
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

## Дополнительная информация

Возможно, вы захотите 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

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

#### Совет

Примечание: 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: kubvirt.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) — это широко используемая технология миграции виртуальных машин, обеспечивающая непрерывность сервиса во время миграции за счёт предварительного копирования данных памяти виртуальной машины. Конкретный процесс следующий:

1. **Начальный этап:** 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 до тех пор, 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 не скопированных 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 состояния CPU и устройств виртуальной машины синхронизируются 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 объяснения параметров см. в [Deploy Operator](#).

1. Перейдите в раздел **Administrator**.
2. В левой навигационной панели выберите **App Store Management > Operators**.
3. Нажмите **Cluster** в верхней части страницы, 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. Для соответствующих настроек см. [Create Subnet \(Kube-OVN Underlay Network\)](#).

1. Перейдите в раздел **Container Platform**.
2. В левой навигационной панели выберите **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 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 <b>Block Mode</b> .
Storage Class	Обязательно использовать класс хранения типа <b>CephRBD block storage</b> .
Network Mode	Рекомендуется использовать <b>Bridge</b> .

6. Нажмите **Create**.

## Запуск горячей миграции

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

1. Перейдите в раздел **Container Platform**.
2. В левой навигационной панели выберите **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 выберите **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**, либо

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 спасения 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 в качестве **Source** платформенный репозиторий **Image Repository**, а в поле **Operating System** — **CentOS** или **Ubuntu**. Справа нажмите **> Update**.

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. Перейдите в **Container Platform**.
2. В левой навигационной панели выберите **Virtualization > Virtual Machines**.
3. Справа у виртуальной машины, требующей ремонта, нажмите **:** > **Stop** или **Force Stop** для остановки.
4. Справа у виртуальной машины нажмите **:** > **Update**.
5. Переключитесь на вкладку **YAML** и измените следующие поля.

- Добавьте следующий блок под `spec.template.spec.domain.devices.disks`.

Параметр `bootOrder` позволяет управлять приоритетом загрузки дисков виртуальной машины; чем ниже значение `bootOrder`, тем выше приоритет.

**Примечание:** Если в исходном поле `spec.template.spec.domain.devices.disks` уже есть `bootOrder: 1`, увеличьте его 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 значение поля `image` на адрес образа, полученный на шаге [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. Справа у виртуальной машины нажмите : > **Start**.

## Монтирование оригинального rootfs и выполнение ремонта

1. Войдите в виртуальную машину, используя оригинальный пароль или ключ, и выполните команду `df -h /`, чтобы убедиться, что файловая система rootfs была заменена. Используйте команды монтирования для монтирования оригинальной файловой системы или команды fsck для её проверки и ремонта.
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 виртуальных машин в KubeVirt

В этом документе приведено пошаговое руководство по клонированию виртуальных машин (VM) с использованием API `VirtualMachineClone` в KubeVirt.

## Содержание

[Убедитесь в выполнении предварительных условий](#)

Быстрый старт

Понимание объекта `VirtualMachineClone`

Полный пример `VirtualMachineClone`

Описание полей

Фазы операции клонирования

## Убедитесь в выполнении предварительных условий

Перед началом операции клонирования VM убедитесь, что выполнены следующие требования:

- **Хранилище с поддержкой снимков:** API клонирования опирается на функции Snapshot & Restore. Класс хранилища виртуальной машины должен поддерживать

**volume snapshots**, а функция создания снимков должна быть явно включена для данного бэкенда хранения.

## Быстрый старт

Следуйте этим простым шагам для клонирования VM:

### 1. Подготовьте манифест клона:

Создайте файл с именем `clone.yaml` со следующей структурой:

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

### 2. Выполните операцию клонирования:

Примените манифест:

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

### 3. Отслеживайте статус клонирования:

Дождитесь успешного завершения клонирования:

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

### 4. Проверьте склонированную VM:

Просмотрите конфигурацию склонированной VM:

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

## 5. Исправьте метку DataVolume (метаданные UI):

В UI платформы связь VM с дисками осуществляется через метку

```
vm.cpaas.io/used-by=<vm-name>
```

, которая автоматически добавляется к каждому

DataVolume.

После операции клонирования новый DataVolume наследует метку от *исходной* VM, поэтому UI по-прежнему считает, что он принадлежит старой VM.

Обновите метку у вновь созданного DV, чтобы связь отображалась корректно (функциональность **не** нарушается).

```
# Список DataVolumes в namespace VM; имя клонированного DV обычно начин  
ается с "restore-"
```

```
kubectl get datavolumes -n <ns-where-vm-run>
```

```
# Перезапишите метку, указав склонированную VM
```

```
kubectl label datavolume <new-dv-name> -n <ns-where-vm-run> vm.cpaas.i  
o/used-by=<target-vm> --overwrite
```

# Понимание объекта VirtualMachineClone

## Полный пример VirtualMachineClone

Ниже приведён полный пример ресурса `VirtualMachineClone` с подробными комментариями:



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

  # Данные целевой VM
  target:
    apiGroup: kubevirt.io
    kind: VirtualMachine
    name: vm-target

  # Фильтры для меток и аннотаций, копируемых из исходной VM
  labelFilters:
    - "*"
    - "!exclude-key/*"
  annotationFilters:
    - "include-annotations/*"

  # Фильтры шаблона для управления сетевыми аннотациями
  template:
    labelFilters:
      - "*"
    annotationFilters:
      - "!network-info/*"

  # Явное указание новых MAC-адресов
  newMacAddresses:
    eth0: "02-00-00-aa-bb-cc"

  # Явное указание нового SMBios serial
  newSMBiosSerial: "unique-serial-1234"

  # JSON-патчи для дополнительной настройки клонированной VM
  patches:
    - '{"op": "add", "path": "/metadata/labels/new-label", "value": "new-value"}'
```

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

## Описание полей

- **source и target:**
  - Определяют исходную VM ( `source` ) и клонированную VM ( `target` ).
  - Автоматически генерируются, если имя `target` не указано.
  - Обе VM должны находиться в одном namespace.
- **Фильтры меток и аннотаций:**
  - Управляют копированием или исключением меток/аннотаций из исходной VM с помощью подстановочных знаков ( `*` ) и отрицаний ( `!` ).
- **Фильтры меток и аннотаций шаблона:**
  - Полезны для управления сетевыми аннотациями, особенно при использовании CNI, таких как Kube-OVN.
- **newMacAddresses:**
  - Опционально задаёт новые MAC-адреса для сетевых интерфейсов.
  - Если не указано, MAC-адреса генерируются автоматически.
- **newSMBiosSerial:**
  - Опционально задаёт новый SMBios serial.
  - Если не указан, генерируется на основе имени VM.
- **JSON-патчи:**
  - Позволяют выполнять продвинутую настройку спецификации VM напрямую.

## Фазы операции клонирования

Поле `.status.phase` объекта `VirtualMachineClone` изменяется в зависимости от этапа процесса клонирования. В таблице ниже описаны возможные фазы:

Фаза	Описание
<b>SnapshotInProgress</b>	Создание снимка исходной VM, начальный этап при клонировании запущенной VM.
<b>CreatingTargetVM</b>	Снимок завершён; создаются метаданные и спецификация для целевой VM.
<b>RestoreInProgress</b>	Создание DataVolume и PersistentVolumeClaim, восстановление данных из снимка.
<b>Succeeded</b>	Операция успешно завершена. Целевая VM и хранилище готовы.
<b>Failed</b>	Операция завершилась с ошибкой. Для подробностей смотрите <code>events</code> и <code>status.conditions</code> .
<b>Unknown</b>	Невозможно определить статус операции клонирования, возможно, проблема с контроллером.

# 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 лимиты

- 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 лимиты

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 операционных системах с архитектурой CPU ARM64.

## Предварительные требования

### Подготовка Chart и образов

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

#### 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 разных операционных системах. 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 параметр 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 следующую команду, чтобы подтвердить успешное включение IOMMU. Если в выводе содержится `IOMMU enabled`, значит IOMMU успешно включён.

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

## Пошаговые действия

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

### Создание Namespace

Выполните следующую 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 следующую команду для развёртывания gpu-operator.

```
export REGISTRY=<registry> # Замените <registry> на адрес репозитория,
где находится образ gpu-operator, например: export REGISTRY=build-harbo
r.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 фактический путь к ch
art, 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 следующую команду, чтобы проверить, синхронизировался ли gpu-operator. Если в столбце `SYNC` отображается `Synced`, значит синхронизация прошла успешно.

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

Вывод:

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

3. Выполните следующую команду, чтобы получить имена всех узлов и найти имя GPU-узла.

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

4. Выполните следующую команду, чтобы проверить, есть ли на GPU-узле GPU, поддерживающий passthrough. 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
```

Вывод:

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

5. На этом этапе gpu-operator успешно развернут.

## Настройка Kubevirt

1. Выполните следующую команду для включения функции `DisableMDEVConfiguration`. Если в ответе появится 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 следующую команду для получения `pciDeviceSelector`. Часть `10de:102d` в выводе — это значение `pciDeviceSelector`.
- ```
{#pciDeviceSelector}
```

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

Вывод:

```
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 следующую команду, чтобы получить имена всех узлов и найти имя GPU-узла.

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

4. Выполните следующую 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
```

Вывод:

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

5. Выполните следующую команду для добавления passthrough GPU.

**Примечание:** При замене части <pci-devices-id> в команде ниже на значение pciDeviceSelector, полученное в [Share 2](#), **все буквы в pciDeviceSelector должны быть преобразованы в верхний регистр**. Например, если значение pciDeviceSelector равно `10de:102d`, его следует 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": ""$DEVICE1"",
          "resourceName": ""$RESOURCE1""
        },
        {
          "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 – 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 навигационной панели выберите **Virtualization > Virtual Machines**.
3. Нажмите **Create Virtual Machine**.
4. Настройте параметр **Physical GPU (Alpha)** для виртуальной машины.

| Параметр                    | Описание                                                                                                            |
|-----------------------------|---------------------------------------------------------------------------------------------------------------------|
| <b>Physical GPU (Alpha)</b> | Выберите модель настроенного физического GPU. Каждой виртуальной машине можно назначить 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 навигационной панели выберите **Virtualization > Virtual Machines**.
3. На странице списка нажмите  $\vdots$  справа от виртуальной машины, которую нужно удалить, и выберите **Delete**, либо перейдите на страницу подробной информации 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
```

Вывод:

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

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

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

Вывод:

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

# Настройка высокой доступности для виртуальных машин

---

## Содержание

[Overview](#)

[Glossary](#)

[Component Overview](#)

[Flow of events during fencing and remediation](#)

[Procedure](#)

[Список операторов](#)

[Развертывание Self Node Remediation Operator](#)

[Настройка Self Node Remediation Operator \(опционально\)](#)

[Настройка шаблона Self Node Remediation \(опционально\)](#)

[Развертывание Node Health Check Operator](#)

[Создание экземпляра NodeHealthCheck](#)

[Проверка \(опционально\)](#)

---

## Overview

Аппаратное обеспечение несовершенно, а программное обеспечение содержит ошибки. При сбоях на уровне узла, таких как зависание ядра или отказ сетевых контроллеров (NIC), нагрузка на кластер не уменьшается, и рабочие нагрузки с затронутых узлов

необходимо перезапустить где-то еще. Однако некоторые рабочие нагрузки, например тома ReadWriteOnce (RWO) и StatefulSets, требуют семантики "не более одного".

Сбои, влияющие на эти рабочие нагрузки, могут привести к потере данных, их повреждению или и тому, и другому. Важно обеспечить достижение узлом безопасного состояния, известного как fencing, перед началом восстановления рабочей нагрузки, известного как remediation, и, по возможности, также восстановление самого узла.

Не всегда практично полагаться на вмешательство администратора для подтверждения истинного состояния узлов и рабочих нагрузок. Для упрощения такого вмешательства Alauda Container Platform предоставляет несколько компонентов для автоматизации обнаружения сбоев, fencing и remediation.

## Glossary

| Акроним | Термин                |
|---------|-----------------------|
| SNR     | Self Node Remediation |
| ННС     | Node Health Check     |

## Component Overview

- **Self Node Remediation Operator**

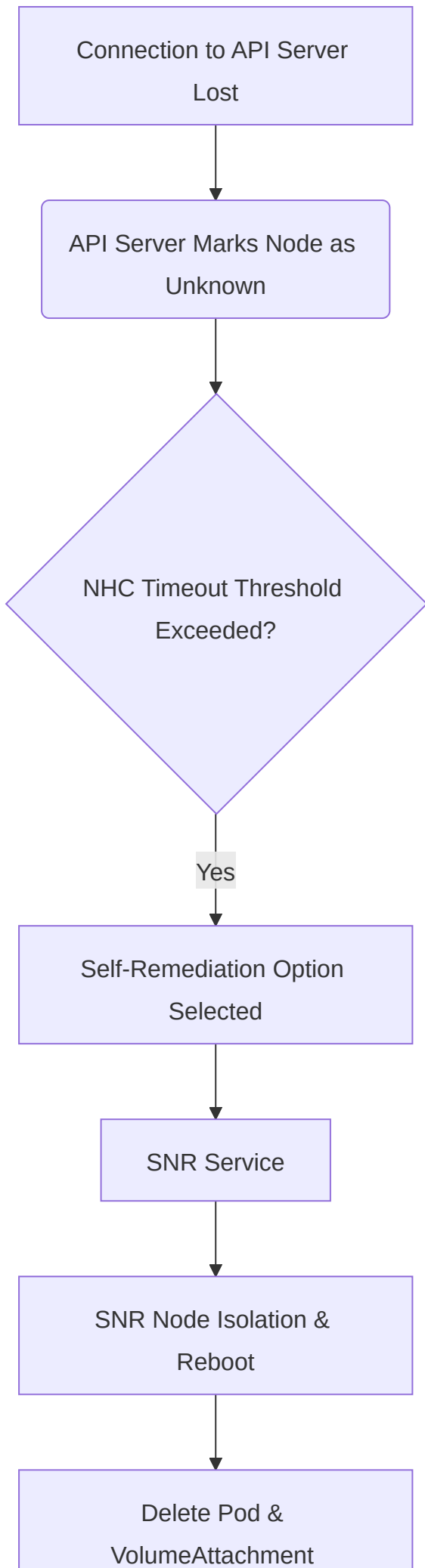
Self Node Remediation Operator — это дополнительный оператор Alauda Container Platform, реализующий внешнюю систему fencing и remediation, которая перезагружает неисправные узлы и удаляет ресурсы, такие как Pods и VolumeAttachments. Перезагрузка гарантирует fencing рабочих нагрузок, а удаление ресурсов ускоряет перепланирование затронутых рабочих нагрузок. В отличие от других внешних систем, Self Node Remediation не требует какого-либо интерфейса управления, например, Intelligent Platform Management Interface (IPMI) или API для управления узлами.

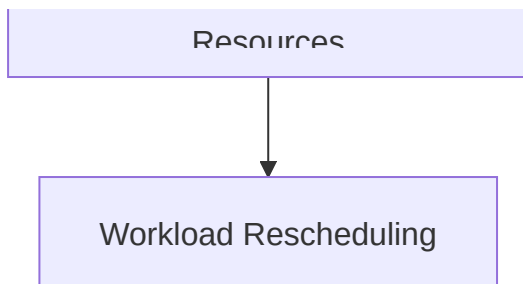
Self Node Remediation может использоваться системами обнаружения сбоев, такими как Machine Health Check или Node Health Check.

- **Node Health Check Operator**

Node Health Check Operator — это дополнительный оператор Alauda Container Platform, реализующий систему обнаружения сбоев, которая мониторит состояние узлов. Он не имеет встроенной системы fencing или remediation, поэтому должен быть настроен с внешней системой, предоставляющей эти функции. По умолчанию он настроен на использование системы Self Node Remediation.

## **Flow of events during fencing and remediation**





## Procedure

### 1 Список операторов

- Скачайте установочный пакет **Alauda Build of SelfNodeRemediation**, соответствующий архитектуре вашей платформы.
- Загрузите установочный пакет **Alauda Build of SelfNodeRemediation** с помощью механизма Upload Packages.
- Скачайте установочный пакет **Alauda Build of NodeHealthCheck**, соответствующий архитектуре вашей платформы.
- Загрузите установочный пакет **Alauda Build of NodeHealthCheck** с помощью механизма Upload Packages.

### 2 Развертывание Self Node Remediation Operator

1. Войдите в систему, перейдите на страницу **Administrator**.
2. Нажмите **Marketplace > OperatorHub**, чтобы перейти на страницу **OperatorHub**.
3. Найдите **Alauda Build of SelfNodeRemediation**, нажмите **Install** и перейдите на страницу **Install Alauda Build of SelfNodeRemediation**.

Параметры конфигурации:

| Параметр          | Рекомендуемая конфигурация                                                                                                                                 |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Channel           | Канал по умолчанию — <code>stable</code> .                                                                                                                 |
| Installation Mode | <code>Cluster</code> : Все пространства имён в кластере используют один экземпляр оператора для создания и управления, что снижает использование ресурсов. |
|                   |                                                                                                                                                            |

| Параметр           | Рекомендуемая конфигурация                                                                                                                 |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| Installation Place | Выберите <code>Recommended</code> , Namespace поддерживает только <code>workload-availability</code> .                                     |
| Upgrade Strategy   | <code>Manual</code> : При наличии новой версии в Operator Hub требуется ручное подтверждение для обновления оператора до последней версии. |

3

## Настройка Self Node Remediation Operator (опционально)

Self Node Remediation Operator создаёт CR `SelfNodeRemediationConfig` с именем `self-node-remediation-config`. CR создаётся в пространстве имён оператора Self Node Remediation.

### Примечание

Изменение CR `SelfNodeRemediationConfig` приводит к пересозданию daemon set Self Node Remediation.

CR `SelfNodeRemediationConfig` выглядит следующим образом:

```

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

```

## Параметры

| Параметр                                   | Описание                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>safeTimeToAssumeNodeRebootedSeconds</b> | Укажите необязательное время ожидания, которое оператор выдерживает перед восстановлением затронутых рабочих нагрузок на неисправном узле. Запуск замещающих подов, пока они ещё работают на сбойном узле, может привести к повреждению данных и нарушению семантики <code>group-once</code> . Оператор рассчитывает минимальную длительность, используя значения полей <code>ApiServerTimeout</code> , <code>ApiCheckInterval</code> , |

| Параметр                | Описание                                                                                                                                                                                                                                                                      |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                         | MaxApiErrorThreshold, PeerDialTimeout и PeerRequestTimeout, а также таймаут watchdog и размер кластера на момент remediation.                                                                                                                                                 |
| watchdogFilePath        | Укажите путь к устройству watchdog на узлах. Если указан неверный путь, Self Node Remediation Operator автоматически обнаружит путь к устройству softdog. Если устройство watchdog недоступно, CR <code>SelfNodeRemediationConfig</code> использует программную перезагрузку. |
| isSoftwareRebootEnabled | Укажите, хотите ли вы включить программную перезагрузку неисправных узлов. По умолчанию значение <code>isSoftwareRebootEnabled</code> установлено в <code>true</code> . Чтобы отключить программную перезагрузку, установите параметр в <code>false</code> .                  |
| apiServerTimeout        | Укажите время ожидания для проверки подключения к каждому API серверу. По истечении этого времени оператор начинает remediation. Время ожидания должно быть не менее 10 миллисекунд.                                                                                          |
| apiCheckInterval        | Укажите частоту проверки подключения к каждому API серверу. Время ожидания должно быть не менее 1 секунды.                                                                                                                                                                    |
| maxApiErrorThreshold    | Укажите пороговое значение. После его достижения узел начинает                                                                                                                                                                                                                |

| Параметр                    | Описание                                                                                                                                                                                      |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                             | связываться с одноранговыми узлами. Значение порога должно быть не менее 1 секунды.                                                                                                           |
| <b>peerApiServerTimeout</b> | Укажите время ожидания подключения однорангового узла к API серверу. Время ожидания должно быть не менее 10 миллисекунд.                                                                      |
| <b>peerDialTimeout</b>      | Укажите время ожидания установления соединения с одноранговым узлом. Время ожидания должно быть не менее 10 миллисекунд.                                                                      |
| <b>peerRequestTimeout</b>   | Укажите время ожидания получения ответа от однорангового узла. Время ожидания должно быть не менее 10 миллисекунд.                                                                            |
| <b>peerUpdateInterval</b>   | Укажите частоту обновления информации об одноранговых узлах, например IP-адреса. Время ожидания должно быть не менее 10 секунд.                                                               |
| <b>hostPort</b>             | Укажите необязательное значение для изменения порта, который агенты Self Node Remediation используют для внутренней связи. Значение должно быть больше 0. Значение по умолчанию — порт 30001. |
| <b>customDsTolerations</b>  | Укажите пользовательские toleration для агентов Self Node Remediation, работающих в DaemonSets, чтобы поддерживать remediation для разных типов узлов.                                        |

### Примечание

- Self Node Remediation Operator по умолчанию создаёт CR в пространстве имён развертывания.
- Имя CR должно быть `self-node-remediation-config`.
- Допускается только один CR `SelfNodeRemediationConfig`.
- Удаление CR `SelfNodeRemediationConfig` отключает Self Node Remediation.

4

## Настройка шаблона Self Node Remediation (опционально)

Self Node Remediation Operator также создаёт Custom Resource Definition (CRD) `SelfNodeRemediationTemplate`. Этот CRD определяет стратегию remediation для узлов, направленную на более быстрое восстановление рабочих нагрузок. Доступны следующие стратегии remediation:

- **Automatic**

Эта стратегия упрощает процесс remediation, позволяя Self Node Remediation Operator выбирать наиболее подходящую стратегию для кластера. Стратегия проверяет наличие стратегии `OutOfServiceTaint` в кластере. Если стратегия `OutOfServiceTaint` доступна, оператор выбирает её. Если нет, оператор выбирает стратегию `ResourceDeletion`. По умолчанию используется стратегия `Automatic`.

- **ResourceDeletion**

Эта стратегия удаляет поды на узле, а не удаляет объект узла.

- **OutOfServiceTaint**

Эта стратегия косвенно приводит к удалению подов и связанных volume attachments на узле, а не удалению объекта узла. Это достигается путём установки стратегии `OutOfServiceTaint` на узле.

Self Node Remediation Operator создаёт CR `SelfNodeRemediationTemplate` для стратегии `self-node-remediation-automatic-strategy-template`, которую использует стратегия `Automatic`.

CR SelfNodeRemediationTemplate выглядит следующим образом:

```

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

```

Параметры

| Параметр             | Описание                                                 |
|----------------------|----------------------------------------------------------|
| remediation_strategy | Значения: Automatic, ResourceDeletion, OutOfServiceTaint |

5

## Развертывание Node Health Check Operator

1. Войдите в систему, перейдите на страницу **Administrator**.
2. Нажмите **Marketplace > OperatorHub**, чтобы перейти на страницу **OperatorHub**.
3. Найдите **Alauda Build of NodeHealthCheck**, нажмите **Install** и перейдите на страницу **Install Alauda Build of NodeHealthCheck**.

Параметры конфигурации:

| Параметр           | Рекомендуемая конфигурация                                                                                                                                 |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Channel            | Канал по умолчанию — <code>stable</code> .                                                                                                                 |
| Installation Mode  | <code>Cluster</code> : Все пространства имён в кластере используют один экземпляр оператора для создания и управления, что снижает использование ресурсов. |
| Installation Place | Выберите <code>Recommended</code> , Namespace поддерживает только <code>workload-availability</code> .                                                     |

| Параметр         | Рекомендуемая конфигурация                                                                                                           |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| Upgrade Strategy | <b>Manual</b> : При наличии новой версии в Operator Hub требуется ручное подтверждение для обновления оператора до последней версии. |

6

## Создание экземпляра NodeHealthCheck

Выполните следующую команду на управляющем узле кластера:

Command

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

Example

```

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

```

Параметры:

| Параметр          | Описание                                                                                                                                                                                    |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>name</b>       | имя ресурса                                                                                                                                                                                 |
| <b>minHealthy</b> | Укажите минимальную долю здоровых узлов. Повреждённые узлы будут ремонтироваться только тогда, когда доля здоровых узлов будет больше или равна этому значению. Значение по умолчанию — 51% |
| <b>selector</b>   | Укажите LabelSelector для выбора узлов, которые будут проверяться и самостоятельно ремонтироваться. Пожалуйста, избегайте                                                                   |

| Параметр | Описание                                                                |
|----------|-------------------------------------------------------------------------|
|          | одновременного указания control-plane и worker узлов в одном экземпляре |

7

## Проверка (опционально)

Смоделируйте сбой работающего узла виртуальной машины и убедитесь, что виртуальная машина автоматически планируется для запуска на других узлах.

# Создание шаблона VM из существующей виртуальной машины

В этом документе описывается, как создать переиспользуемый шаблон виртуальной машины (VM) на основе существующей VM для быстрого развертывания новых VM.

## Содержание

### [Предварительные требования](#)

#### Процедура

Шаг 1: Базовая настройка виртуальной машины

Шаг 2: Создание снимка VM

Шаг 3: Получение полного имени ресурса снимка диска

Шаг 4: Создание ресурса DataSource

Объяснение параметров меток:

Шаг 5: Создание новой VM с использованием шаблона

## Предварительные требования

- Корректно развернутая и настроенная среда KubeVirt.
- Доступ к Web Console и инструменту kubectl.
- Настроенная VM с уже установленным необходимым программным обеспечением.

# Процедура

## Шаг 1: Базовая настройка виртуальной машины

Внутри VM выполните следующие действия:

- Установите [cloud-init](#) ↗.
- Установите `qemu-guest-agent`.
- Установите любое необходимое программное обеспечение.

После завершения установки выполните команды для очистки данных cloud-init и выключения VM:

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

## Шаг 2: Создание снимка VM

Используя KubeVirt Web Console:

1. Перейдите в **Virtualization > Virtual Machines**.
2. Выберите VM, которую планируете использовать в качестве шаблона.
3. Нажмите **Actions**, выберите **Create Snapshot**, задайте имя снимка и подтвердите.

## Шаг 3: Получение полного имени ресурса снимка диска

Получите полное имя ресурса снимка одним из следующих способов:

- **Через Web Console:**
  - Перейдите в **Storage > Volume Snapshots**.
  - Найдите и запишите полное имя ресурса снимка в поле "Data Source".
- **С помощью kubectl:**

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

Запишите полное имя ресурса снимка из вывода команды.

## Шаг 4: Создание ресурса DataSource

Создайте следующий ресурс DataSource в пространстве имён `kube-public`, заменив заполнители на фактическое имя снимка и пространство имён:

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

### Объяснение параметров меток:

| Ключ                                  | Возможные значения | Описание          |
|---------------------------------------|--------------------|-------------------|
| virtualization.cpaas.io/image-os-arch | amd64, arm64       | Архитектура ОС VM |
|                                       |                    |                   |

| Ключ                                  | Возможные значения           | Описание                                                               |
|---------------------------------------|------------------------------|------------------------------------------------------------------------|
| virtualization.cpaas.io/image-os-type | linux, windows               | Тип ОС VM                                                              |
| virtualization.cpaas.io/storage-class | имя класса хранения          | Класс хранения по умолчанию, можно изменить при создании VM            |
| virtualization.cpaas.io/access-mode   | ReadWriteOnce, ReadWriteMany | Режим доступа к диску; для живой миграции VM используйте ReadWriteMany |
| virtualization.cpaas.io/size          | Размер (Gi, Ti и т.д.)       | Размер диска по умолчанию; укажите подходящий размер                   |
| virtualization.cpaas.io/volume-mode   | Block, Filesystem            | Режим тома диска; рекомендуется Block для лучшей производительности    |

**Важно:**

- Убедитесь, что пространство имён — `kube-public`.
- Эти параметры, связанные с диском, можно изменить при создании VM, но указание значений по умолчанию упрощает процесс.

**Шаг 5: Создание новой VM с использованием шаблона**

1. Зайдите в KubeVirt Web Console, перейдите в **Container Platform > Virtualization > Virtual Machines**.
2. Нажмите **Create Virtual Machine**.
3. В разделе **Image** выберите **Image Instance** в качестве метода предоставления.
4. Выберите созданный вами DataSource из выпадающего списка.
5. Настройте дополнительные параметры по необходимости и завершите создание VM.

Теперь вы успешно создали и развернули новые VM с использованием вашего шаблона VM.



# Устранение неполадок

[Миграция Pod и восстановление работы виртуальных машин |](#)

[Сообщения об ошибках горячей миграции |](#)

Описание проблемы

Анализ причины

Решения

# Миграция Pod и восстановление после аномального завершения работы виртуальных машин на узлах

## Содержание

### Описание проблемы

Анализ причины

Решения

Миграция виртуальных машин при корректном завершении работы

Восстановление после аномального завершения работы

## Описание проблемы

Независимо от того, происходит ли **корректное завершение работы** узла или **аномальный сбой**, виртуальные машины (Pods), запущенные 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 **аномального сбоя** виртуальные машины (Pods) должны быть вручную эвакуированы или принудительно удалены.

**Примечание:** Все приведённые ниже команды необходимо выполнять на Master-узле соответствующего кластера.

## Миграция 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 команду для просмотра виртуальных машин на узле.

```
kubectl get vmis --all-namespaces -o wide | grep <Node-Name> # Заменит  
е <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.
1.1.1.211  True    False
```

3. Перед корректным завершением работы выполните команду для эвакуации всех виртуальных машин (Pods) с узла, который будет выключен. Если вывод выглядит следующим образом, эвакуация прошла успешно.

```
kubectl drain <Node-Name> --delete-local-data --ignore-daemonsets=true
--force --pod-selector=kubevirt.io=virt-launcher # Замените <Node-Nam
e> на имя 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 команду, чтобы снять запрет на планирование Pod на этом узле.

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

Вывод:

```
node/1.1.1.211 uncordoned
```

6. Теперь исходные виртуальные машины с этого узла мигрировали на другие здоровые узлы, и данный узел доступен для планирования новых Pod после перезагрузки.

## Восстановление после аномального завершения работы

1. В CLI выполните 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 команду для принудительного удаления всех виртуальных машин (Pods) 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 # 3
Замените <Node-Name> на имя узла, на котором произошёл аномальный сбой.
```

3. Выполните 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> на имя узла, на котором произошёл аномальный сбой.
```

4. Выполните команду для проверки наличия Pod с меткой `kubevirt.io=virt-api` на узле с аномальным сбоем.

```
kubectl -n kubevirt get po -l kubevirt.io=virt-api -o wide | grep <Node-Name> # Замените <Node-Name> на имя узла, на котором произошёл аномальный сбой.
```

Если такие Pod существуют, выполните 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 команду для проверки наличия Pod с меткой `kubevirt.io=virt-controller` на узле с аномальным сбоем.

```
kubectl -n kubevirt get po -l kubevirt.io=virt-controller -o wide | gre  
p <Node-Name> # Замените <Node-Name> на имя узла, на котором произошёл  
аномальный сбой.
```

Если такие Pod существуют, выполните команду для их удаления.

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

6. После этого виртуальные машины будут мигрированы на другие здоровые узлы после аномального завершения работы узла.

# 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.                | <p>Пожалуйста, проверьте следующие настройки:</p> <ul style="list-style-type: none"> <li>Проверьте CNI сетевой плагин, используемый в текущем кластере; рекомендуется Kube-OVN.</li> <li>Проверьте наличие аннотации "kubevirt.io/allow-pod-bridge-network-live-migration": "true" в полях <code>metadata.annotations</code> и <code>spec.template.metadata.annotations</code> соответствующего YAML-файла виртуальной машины; если её нет, добавьте ручную.</li> </ul> |
| <ul style="list-style-type: none"> <li>cannot migrate VMI: Unable to determine if PVC &lt;pvc name&gt; is shared, live migration requires that all PVCs must be shared (using</li> </ul> | Тип хранилища виртуальной машины не поддерживает многозвенный режим чтения-записи (RWX). | <p>Параметры, связанные с виртуальной машиной, нельзя изменить после создания. Поэтому, пожалуйста, пересоздайте виртуальную машину, выбрав тип хранилища, поддерживающий многозвенный режим чтения-записи (RWX); рекомендуется блочное хранилище CephRBD. Если проблемы сохраняются после пересоздания, обратитесь к соответствующим специалистам за помощью.</p>                                                                                                      |

| Сообщение об ошибке                                                                                                                                                                                                                                                                                                                             | Причина                                              | Решение                                                           |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------|-------------------------------------------------------------------|
| <p>ReadWriteMany access mode)</p> <ul style="list-style-type: none"><li>cannot migrate VMI: PVC &lt;pvc name&gt; is not shared, live migration requires that all PVCs must be shared (using ReadWriteMany access mode)</li><li>cannot migrate VMI: Backend storage PVC is not RWX</li><li>cannot migrate VMI with non-shared HostDisk</li></ul> |                                                      |                                                                   |
| Другие сообщения об ошибках                                                                                                                                                                                                                                                                                                                     | Виртуальная машина не поддерживает горячую миграцию. | Пожалуйста, обратитесь к соответствующим специалистам за помощью. |

# Сеть

---

## Введение

### Введение

Преимущества

---

## Руководства

### Настройка сети

Настройка IP

Подключение к виртуальной машине напрямую по IP

Добавление внутренних маршрутов

---

## Как сделать

### Контроль сетевых запросов с Network Policy

Процедура

### Настройка SR-IOV

Терминология

### Настройка биндинга с

Предварительные

---

Проверка результата

Ограничения и лимиты

Процедура

Предварительные требования

Процедуры

Проверка результата

Связанные заметки

# 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 одного и того же IP-адреса у VM после перезапусков, что соответствует устоявшимся паттернам использования VM.
- **Поддержка мультисетевого режима**  
Поддерживает несколько сетевых режимов, таких как контейнерные сети и SR-IOV, удовлетворяя разнообразные пользовательские сценарии.

# Руководства

## Настройка сети

Настройка IP

Подключение к виртуальной машине напрямую по IP

Добавление внутренних маршрутов

# Настройка сети

---

## Содержание

### Настройка IP

Подключение к виртуальной машине напрямую по IP

Добавление внутренних маршрутов

---

## Настройка IP

Ссылки на [Configure IP](#)

## Подключение к виртуальной машине напрямую по IP

Ссылки на [Preparing Kube-OVN Underlay Physical Network](#)

## Добавление внутренних маршрутов

Ссылки на [Add Internal Routes](#)

---

# Практическое руководство

## Контроль сетевых запросов в Network Policy

Процедура

Проверка результата

## Настройка SR-IOV

Терминология

Ограничения и лимиты

Предварительные требования

Процедуры

Проверка результата

Связанные заметки

## Настройка биндинга с

Предварительные

Процедура

# Контроль сетевых запросов виртуальной машины с помощью Network Policy

Решение платформы для виртуальных машин реализовано на базе open-source компонента KubeVirt, который фактически работает внутри Pod. 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                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Association Method</b> | <ul style="list-style-type: none"> <li>• <b>Compute Component:</b> Выберите целевой вычислительный компонент по необходимости; рекомендуется выбрать <b>All</b> в качестве целевого вычислительного компонента.</li> <li>• <b>Label Selector:</b> Соответствие Pod по меткам.</li> </ul>                                                                                                                                                                       |
| <b>Direction</b>          | <ul style="list-style-type: none"> <li>• <b>Ingress:</b> Запросы, поступающие извне в Pod.</li> <li>• <b>Egress:</b> Запросы, отправляемые из Pod во внешнюю сеть; выберите этот вариант, если необходимо запретить виртуальной машине делать запросы к определённому внешнему адресу.</li> </ul>                                                                                                                                                              |
| <b>Protocol</b>           | <p>Выберите TCP или UDP.</p> <p><b>Примечание:</b></p> <ul style="list-style-type: none"> <li>• При использовании доменных имён в виртуальной машине для запросов к внешним сервисам необходимо добавить в белый список протокол UDP, так как DNS использует UDP.</li> <li>• Форма не поддерживает настройку протокола ICMP; при включении правил белого списка протокол ICMP будет отключён, что приведёт к невозможности выполнять операции Ping.</li> </ul> |
| <b>Access Ports</b>       | <p>Укажите порты, трафик через которые разрешён для входящего или исходящего направления. Если поле оставить пустым, по умолчанию разрешён трафик через все порты.</p> <p><b>Примечание:</b> Здесь необходимо разрешить порты 1053 и 53 для протоколов UDP и TCP, чтобы разрешить исходящий DNS-трафик, иначе разрешение доменных имён не будет работать.</p>                                                                                                  |
| <b>Remote Type</b>        | <p>Укажите разрешённый тип удалённого доступа. Варианты: вычислительный компонент, namespace и IP-сегменты.</p>                                                                                                                                                                                                                                                                                                                                                |

| Параметр              | Описание                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Exclude Remote</b> | <p>При выборе типа удалённого доступа <b>IP Segment</b> можно исключить указанный IP из белого списка (то есть запретить доступ). Один IP можно исключить, указав его в формате <code>IP/32</code>.</p> <p><b>Примечание:</b> В это поле можно вводить только IP; если соответствующий IP доменного имени неизвестен, используйте команду <code>curl -vvv &lt;domain&gt;</code>, чтобы сделать запрос к домену и получить IP из возвращаемой информации.</p> |

5. Нажмите **Create**.

## Проверка результата

В данном документе проверяется настройка с использованием виртуальной машины для доступа к [www.example.com](http://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 командном namespace виртуальной машины, добавив правила белого списка для протоколов TCP и UDP со следующими параметрами:
  - Правила белого списка для протокола TCP:

| Параметр                        | Описание                            |
|---------------------------------|-------------------------------------|
| <b>Association Method</b>       | Выберите <b>Compute Component</b> . |
| <b>Target Compute Component</b> | Выберите <b>All</b> .               |

| Параметр    | Описание                                                                                |
|-------------|-----------------------------------------------------------------------------------------|
| Direction   | Выберите <b>Egress</b> .                                                                |
| Protocol    | Выберите <b>TCP</b> .                                                                   |
| Remote Type | Выберите <b>IP Segment</b>                                                              |
| Remote      | Введите <b>0.0.0.0/0</b> , что означает разрешение исходящего трафика для всех адресов. |

- Правила белого списка для протокола UDP:

| Параметр    | Описание                                                                                |
|-------------|-----------------------------------------------------------------------------------------|
| Direction   | Выберите <b>Egress</b> .                                                                |
| Protocol    | Выберите <b>UDP</b> .                                                                   |
| Remote Type | Выберите <b>IP Segment</b>                                                              |
| Remote      | Введите <b>0.0.0.0/0</b> , что означает разрешение исходящего трафика для всех адресов. |

3. После создания Network Policy войдите в виртуальную машину и выполните следующую команду для запроса [www.example.com](http://www.example.com) ↗.

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

4. Запрос выполнен успешно.

## Шаг второй: Обновление Network Policy для исключения [www.example.com](http://www.example.com) ↗ из белого списка

1. Выполните следующую команду для получения IP-адреса [www.example.com](http://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                                                                                                                                                                                                    |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Exclude Remote</b> | В правилах белого списка для протокола TCP заполните параметр <code>exclude remote</code> значением <code>93.184.215.14/32</code> , что означает исключение IP <code>93.184.215.14</code> из белого списка. |

3. После обновления Network Policy войдите в виртуальную машину и выполните следующую команду для запроса [www.example.com](http://www.example.com) ↗.

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

4. Запрос завершается по таймауту, что свидетельствует о том, что функция `exclude remote` работает корректно.

# 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 dual-stack IPv4/IPv6.

## Содержание

### Терминология

Ограничения и лимиты

Предварительные требования

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 идентификатора роли узла для физических узлов

Проверка успешного создания ресурсов

Установка меток функции узла 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 высокую производительность сети.                        |
| VF         | Виртуальное устройство, созданное из физического PCI-устройства; VF можно выделять напрямую виртуальным машинам или контейнерам, они выглядят как независимые физические PCI-устройства, значительно улучшая 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 на этих ОС использовать нельзя.

## Предварительные требования

Получите следующие чарты и образы и загрузите их в репозиторий образов. В данном документе в качестве примера используется адрес репозитория `build-`

`harbor.example.cn`. Для получения конкретных способов получения чартов и образов обратитесь к ответственным лицам.

## 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 нижеуказанные команды выполняются в терминале.

### 1 Включение SR-IOV в BIOS физической машины

Перед настройкой выполните 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
```

Процесс включения SR-IOV в BIOS отличается у разных производителей серверов. Обратитесь к документации соответствующего производителя. Обычно шаги следующие:

1. Перезагрузите сервер.
2. При появлении логотипа бренда 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 настройки и перезагрузите сервер.

## 2

## Включение IOMMU

Процесс включения IOMMU может отличаться в разных ОС. В этом документе приведён пример для CentOS.

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 qu
iet 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 появится `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 команду. Если информация выводится корректно, 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
```

## Создание устройств VF

1. Выполните команду для просмотра поддерживаемых 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:

- **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 NIC 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>
    ether a0:36:9f:29:6c:00 txqueuelen 1000 (Ethernet)
    RX packets 13889 bytes 1075801 (1.0 MB)
    RX errors 0 dropped 1603 overruns 0 frame 0
    TX packets 5057 bytes 440807 (440.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp5s0f1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet6 fe80::a236:9fff:fe29:6c02 prefixlen 64 scopeid 0x2
    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 NIC, 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 NIC 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 NIC 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 данном документе в качестве примера используется настройка NIC enp5s0f1. Если необходимо виртуализировать несколько NIC, настройте их все.

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

echo 8 > /sys/devices/pci0000:00/0000:00:03.0/0000:05:00.1/sriov_n
umvfs    ## 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 8 настроенных адресов VF, например `05:10.1`. Эти адреса VF необходимо дополнить идентификатором домена, итоговый 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 Cont
roller Virtual Function (rev 01)
05:10.3 Ethernet controller: Intel Corporation 82599 Ethernet Cont
roller Virtual Function (rev 01)
05:10.5 Ethernet controller: Intel Corporation 82599 Ethernet Cont
roller Virtual Function (rev 01)
05:10.7 Ethernet controller: Intel Corporation 82599 Ethernet Cont
roller Virtual Function (rev 01)
05:11.1 Ethernet controller: Intel Corporation 82599 Ethernet Cont
roller Virtual Function (rev 01)
05:11.3 Ethernet controller: Intel Corporation 82599 Ethernet Cont
roller Virtual Function (rev 01)
05:11.5 Ethernet controller: Intel Corporation 82599 Ethernet Cont
roller Virtual Function (rev 01)
05:11.7 Ethernet controller: Intel Corporation 82599 Ethernet Cont
roller Virtual Function (rev 01)
```

5

## 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 NIC `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 в разделе **Network devices using DPDK-compatible driver** найдите уже привязанные VF. 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=en  
o1 drv=tg3 unused=vfio-pci
```

```
0000:01:00.1 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=en  
o2 drv=tg3 unused=vfio-pci
```

```
0000:02:00.0 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=en  
o3 drv=tg3 unused=vfio-pci
```

```
0000:02:00.1 'NetXtreme BCM5720 Gigabit Ethernet PCIe 165f' if=en  
o4 drv=tg3 unused=vfio-pci
```

```
0000:05:00.0 'Ethernet 10G 2P X520 Adapter 154d' if=enp5s0f0 drv=i  
xgbe unused=vfio-pci *Active*
```

```
0000:05:00.1 'Ethernet 10G 2P X520 Adapter 154d' if=enp5s0f1 drv=i  
xgbe unused=vfio-pci
```

```
No 'Baseband' devices detected
```

```
=====
```

```
No 'Crypto' devices detected
```

```
=====
```

```
No 'DMA' devices detected
```

```
=====  
  
No 'Eventdev' devices detected  
=====  
  
No 'Mempool' devices detected  
=====  
  
No 'Compress' devices detected  
=====  
  
No 'Misc (rawdev)' devices detected  
=====  
  
No 'Regex' devices detected  
=====
```

6

## Развёртывание плагина Multus CNI

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

7

## Развёртывание sriov-network-operator

Выполните 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"], разделяя несколько запятыми
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

```

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

1. Перейдите в раздел **Administrator**.
2. В левой навигационной панели выберите **Cluster Management > Clusters**.
3. Нажмите на имя кластера и переключитесь 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` для проверки успешного создания ресурса `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 функции узла SR-IOV для физических узлов

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

1. Перейдите в раздел **Administrator**.
2. В левой навигационной панели выберите **Cluster Management > Clusters**.
3. Нажмите на имя кластера и переключитесь 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 текущий vendor ID и device ID 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 X52
0 Adapter (rev 01)
```

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

**Примечание:** Если текущего NIC нет в списке поддержки, необходимо добавить NIC в конфигурационный файл согласно [Шагу 4](#). Если NIC уже есть 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 `<NIC Name>: <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          ## Добавьте ин
формацию о новом 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, можно найти в [результатах привязки](#).

12

## Настройка IP-адреса

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

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

## Проверка результата

1. Перейдите в раздел **Container Platform**.
2. В левой навигационной панели выберите **Virtualization > Virtual Machines**.
3. Нажмите **Create Virtual Machine**, 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 означает успешную настройку.

```

root@sriov-demo ~]#
root@sriov-demo ~]# dhclient eth1
root@sriov-demo ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1400 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:00:00:0c:8f:c0 brd ff:ff:ff:ff:ff:ff
    inet 10.33.0.44/16 brd 10.33.255.255 scope global dynamic eth0
        valid_lft 86313367sec preferred_lft 86313367sec
    inet6 fe80::200:ff:fe0c:8fc0/64 scope link
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 06:1e:b5:e1:5f:f7 brd ff:ff:ff:ff:ff:ff
    inet 192.168.39.7/24 brd 192.168.39.255 scope global dynamic eth1
        valid_lft 86398sec preferred_lft 86398sec
    inet6 2002::41e:b5ff:fee1:5ff7/64 scope global mngtmpaddr dynamic
        valid_lft 2591997sec preferred_lft 604797sec
    inet6 fe80::41e:b5ff:fee1:5ff7/64 scope link
        valid_lft forever preferred_lft forever
root@sriov-demo ~]#

```

## Связанные заметки

### Конфигурация параметров ядра для виртуальных машин CentOS

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

1. Откройте терминал и выполните команду для изменения параметров ядра для соответствующего NIC. 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 будут иметь возможности 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

### [Предварительные 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

---

# Предварительные 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>
```

# Хранение

---

## 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**  
Возможность настройки автоматического удаления дисков VM при завершении работы соответствующей виртуальной машины.

# Руководства

## Управление виртуальными дисками

[Создание виртуального диска](#)

[Монтирование виртуального диска](#)

[Расширение виртуального диска](#)

[Отмонтирование виртуального диска](#)

[Удаление виртуального диска](#)

# Управление виртуальными дисками

Дисковые устройства данных могут использоваться для удовлетворения требований бизнеса к сохранности 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 **дисковое устройство 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 **running**.

## Процедуры

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

| Параметр              | Описание                                                                                                                                                                                                                                                                                                                                                                                                               |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Volume Mode</b>    | <ul style="list-style-type: none"> <li>- <b>File System:</b> Монтирование диска с использованием файловой системы.</li> <li>- <b>Block Device:</b> Монтирование диска как блочного устройства.</li> </ul>                                                                                                                                                                                                              |
| <b>Storage Class</b>  | <p>Платформа поддерживает диски виртуальных машин, автоматически создавая и управляя persistent volume claims. Необходимо указать класс хранилища, который будет использоваться для динамического создания persistent volume claims.</p> <p>Разные классы хранилища поддерживают разные режимы томов. Если для выбранного режима тома нет доступных классов хранилища, обратитесь к администратору для добавления.</p> |
| <b>Delete with VM</b> | Если включено, данные диска будут удалены при удалении виртуальной машины.                                                                                                                                                                                                                                                                                                                                             |
| <b>Mount</b>          | <ul style="list-style-type: none"> <li>- <b>Do Not Mount:</b> Создать только виртуальный диск; монтирование можно выполнить позже при необходимости.</li> <li>- <b>Mount to VM:</b> Выбрать целевую виртуальную машину, к которой необходимо примонтировать виртуальный диск.</li> </ul>                                                                                                                               |

5. Нажмите **Create**.

# Монтирование виртуального диска

Примонтируйте **дисковое устройство 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 **running**.

## Процедуры

1. Зайдите в **Container Platform**.
2. В левой навигационной панели выберите **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 **дисковое устройство данных**, уже примонтированные к виртуальной машине.

## Процедуры

1. Зайдите в **Container Platform**.
2. В левой навигационной панели выберите **Virtualization > Virtual Machine**.
3. Нажмите на имя виртуальной машины, чтобы перейти на страницу **Details**.
4. В разделе **Virtual Disk** найдите диск для расширения и нажмите **:** > **Expand**.
5. Введите новый размер и нажмите **Expand**.

# Отмонтирование виртуального диска

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

## Процедуры

1. Зайдите в **Container Platform**.
2. В левой навигационной панели выберите **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 состоянии отмонтированного.

**Примечание:** Системные диски удалять нельзя.

## Процедуры

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

# Резервное копирование и восстановление

## Введение

### Введение

Сценарии применения

Ограничения использования

## Руководства

### Использование снимков

Предварительные требования

Примечания

Создание снимка

Откат к снимку

Удаление снимка

### Использование Velero

Предварительные требования

Шаги выполнения

# Введение

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

# Руководства

---

## Использование снимков

Предварительные требования

Примечания

Создание снимка

Откат к снимку

Удаление снимка

## Использование Velero

Предварительные требования

Шаги выполнения

# Использование снимков

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

## Содержание

### [Предварительные требования](#)

Примечания

Создание снимка

Процедура

Откат к снимку

Примечания

Процедура

Удаление снимка

Примечания

Процедура

## Предварительные требования

- Администратором на платформе был развернут **Volume Snapshot**.
- Снимки виртуальной машины основаны на volume snapshots. Убедитесь, что хотя бы один диск привязан к классу хранения, поддерживающему volume snapshots,

например, встроенному хранилищу CephFS.


## 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 включаются: 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 состояния сети и нагрузки, пожалуйста, подождите.
6. Проверьте статус снимка.
  - Когда статус снимка изменится на **Ready**, это означает успешное создание.
  - Если снимок долго находится в статусе **Not Ready**, нажмите  **>** Просмотрите причины и устраните неполадки, затем создайте снимок заново.

## Откат к снимку

Выполняется откат настроек виртуальной машины и дисков, поддерживающих volume snapshots, к состоянию на момент создания снимка. Например, диски, добавленные после создания снимка, будут удалены; изменённые данные на дисках будут восстановлены.

## Примечания

Если диски привязаны к классу хранения, поддерживающему механизм LVM (например, ToroLVM), пожалуйста, уточните у администратора, что политика рекламации для этого класса хранения 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 на экране и нажмите **Rollback** после подтверждения правильности данных.  
**Примечание:** Операцию отката нельзя прервать или отменить, будьте внимательны.
6. Нажмите на имя снимка, чтобы проверить в разделе «Snapshot Rollback Records», завершён ли откат. Время отката зависит от состояния сети и нагрузки, пожалуйста, подождите.

## Описание

- Если откат не удался, состояние виртуальной машины останется без изменений. Вы можете запустить виртуальную машину в обычном режиме или повторить попытку отката снимка.
- 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 работе, убедитесь, что последний откат завершён, прежде чем выполнять другие операции 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 диск виртуальной машины должен копировать данные на основе снимка (например, ToraLVM), необходимо дождаться запуска виртуальной машины на основе версии после отката, прежде чем удалять снимок, 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 и нажмите **Delete** после подтверждения.

# Использование Velero

Velero — это инструмент с открытым исходным кодом для резервного копирования и миграции кластеров Kubernetes от VMware. KubeVirt предоставляет плагин Velero для поддержки резервного копирования и восстановления виртуальных машин.

## Содержание

### [Предварительные требования](#)

#### Шаги выполнения

Подготовка

Резервное копирование

Восстановление

Восстановление между кластерами

Восстановление в другой namespace

Восстановление с другим StorageClass

## Предварительные требования

- **Версия Kubernetes:** 1.20 или выше (требование Velero)
- **S3-хранилище:** для BackupStorageLocation Velero используйте объектное хранилище S3 или MinIO, предоставляемое платформой
- **Блочное хранилище:** S3 RBD, предоставляемое платформой

# Шаги выполнения

## Подготовка

1. Разверните Velero на платформе ACP через **Platform Management** → **Cluster Management** → **Backup and Recovery** → **Backup Repository**. Используйте данные объектного хранилища для создания репозитория резервных копий.
2. Включите функцию CSI и добавьте плагин KubeVirt:

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

Добавьте следующее в деплоймент Velero:

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

Добавьте в секцию initContainers:

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

3. Проверьте установку плагина KubeVirt:

```

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

```

Пример вывода:

```

kubevirt-velero-plugin/backup-datavolume-action      BackupItemAction
kubevirt-velero-plugin/backup-datavolume-action      BackupItemAction
kubevirt-velero-plugin/backup-datavolume-action      BackupItemAction
kubevirt-velero-plugin/backup-virtualmachine-action  BackupItemAction
kubevirt-velero-plugin/backup-virtualmachine-action  BackupItemAction
kubevirt-velero-plugin/backup-virtualmachine-action  BackupItemAction
kubevirt-velero-plugin/backup-virtualmachineinstance-action BackupItemAction
kubevirt-velero-plugin/backup-virtualmachineinstance-action BackupItemAction
kubevirt-velero-plugin/backup-virtualmachineinstance-action BackupItemAction
kubevirt-velero-plugin/restore-pod-action             RestoreItemAction
kubevirt-velero-plugin/restore-pod-action             RestoreItemAction
kubevirt-velero-plugin/restore-pod-action             RestoreItemAction
kubevirt-velero-plugin/restore-pvc-action             RestoreItemAction
kubevirt-velero-plugin/restore-pvc-action             RestoreItemAction
kubevirt-velero-plugin/restore-pvc-action             RestoreItemAction
kubevirt-velero-plugin/restore-pvc-action             RestoreItemAction
kubevirt-velero-plugin/restore-vm-action              RestoreItemAction
kubevirt-velero-plugin/restore-vm-action              RestoreItemAction
kubevirt-velero-plugin/restore-vm-action              RestoreItemAction
kubevirt-velero-plugin/restore-vm-action              RestoreItemAction
kubevirt-velero-plugin/restore-vmi-action             RestoreItemAction
kubevirt-velero-plugin/restore-vmi-action             RestoreItemAction
kubevirt-velero-plugin/restore-vmi-action             RestoreItemAction

```

4. Настройте node-agent для монтирования директории kubelet хоста и включите привилегированный режим (требуется для передачи данных через

`/var/lib/kubelet`):

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

Обновите следующим образом:

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

## Резервное копирование

1. Создайте виртуальную машину с необходимым количеством дисков.
2. Создайте ресурс резервного копирования:

```

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

```

3. Дождитесь завершения резервного копирования и проверьте его:

```

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

```

Пример вывода:

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

4. Проверьте статус dataupload для передачи данных:

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

Пример вывода:

| NAME                 | STATUS    | STARTED        | BYTES DONE  | TOTAL BYTES |
|----------------------|-----------|----------------|-------------|-------------|
| example-backup-fhc6b | Completed | 11h            | 21474836480 | 21474836480 |
| default              | 11h       | 192.168.254.66 |             |             |
| example-backup-qwzwh | Completed | 11h            | 21474836480 | 21474836480 |
| default              | 11h       | 192.168.254.15 |             |             |

5. Подтвердите завершение резервного копирования:

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

Пример вывода:

| NAME           | STATUS    | ERRORS | WARNINGS | CREATED                       |
|----------------|-----------|--------|----------|-------------------------------|
| example-backup | Completed | 0      | 0        | 2025-01-25 14:14:08 +0000 UTC |
| 29d            | default   |        | <none>   |                               |

6. Проверьте репозиторий резервных копий (например, MinIO) на наличие директорий:

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

Пример вывода:

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

## Восстановление

1. Удалите исходную виртуальную машину.
2. Создайте ресурс восстановления через **Platform Management** → **Cluster Management** → **Backup and Recovery** → **Restore Management** или вручную:

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

3. Проверьте восстановление:

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

Проверьте статус datadownload:

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

Пример вывода:

```

NAME          STATUS      STARTED   BYTES DONE   TOTAL BYTES   STORAGE LO
CATION    AGE      NODE
example-restore-7lfc8  Completed   11m        21474836480  21474836480
default                               15m        192.168.254.66
example-restore-cjcd8  Completed   15m        21474836480  21474836480
default                               15m        192.168.254.66

```

Статус восстановления:

```

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

```

4. Подтвердите, что виртуальная машина восстановлена и работает.

## Восстановление между кластерами

1. Разверните Velero в новом кластере и выполните шаги подготовки.
2. Настройте тот же репозиторий резервных копий (тот же бакет и директорию), что и в исходном кластере. Ресурс резервного копирования появится автоматически.
3. Следуйте шагам восстановления, описанным выше.

## Восстановление в другой namespace

Добавьте поле `namespaceMapping` в спецификацию восстановления:

```

спес:
  namespaceMapping:
    example-namespace: test-namespace

```

## Восстановление с другим StorageClass

1. Создайте ConfigMap в namespace `cpaas-system` перед началом восстановления:

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

2. Убедитесь, что возможности хранилища (например, доступность между узлами, RWX) совпадают.
3. Примечание: изменение режима RWX в настоящее время не поддерживается.