УТВЕРЖДЕНО

RU.86432418.00001-01 90 04-1 - ЛУ

Программное обеспечение «Deckhouse Platform Certified Security Edition»

Руководство администратора

RU.86432418.00001-01 90 04-1

202 Листа

2025

Содержание

Список используемых обозначений и сокращений

API	Application Programming Interface, программный интерфейс приложения
CRI, container runtime, контейнерный рантайм	Среда исполнения контейнеров
DVCR	Встроенный сервис, обеспечивающий хранение, версионирование и доступ к образам виртуальных машин
Persistent Volume Claim (PVC)	запрос пользователя или приложения на использование постоянного хранилища (Persistent Volume, PV) в Kubernetes
DVD	Digital Versatile Disc, цифровой многоцелевой диск
TLS	Протокол шифрования, который обеспечивает безопасную передачу данных в интернете, защищая конфиденциальность и целостность информации от перехвата и несанкционированного доступа
SSH	Сетевой протокол, позволяющий производить удалённое управление операционной системой и туннелирование TCP (Transmission Control Protocol)-соединений
USB	Universal Serial Bus, интерфейс для подключения периферийных устройств
YAML	Язык для структурированной записи информации, обладающий простым синтаксисом
ОС	Операционная система
ПО	Программное обеспечение
ПО «Deckhouse Platform»	Программное обеспечение «Deckhouse Platform Certified Security Edition»

1 Действия по приемке поставленного средства

Приемка поставленного ПО «Deckhouse Platform» осуществляется в соответствии с указаниями, содержащимися в документе «Программное обеспечение «Deckhouse Platform». Технические условия. RU.86432418.00001-01 ТУ 04-1».

Перед началом эксплуатации, для обнаружения любого расхождения между оригиналом ПО «Deckhouse Platform» и версией, полученной Заказчиком, проводится процедура приемки.

Приемка поставленной ПО «Deckhouse Platform» включает в себя следующие процедуры:

- проверка упаковки;
- проверка маркировки;
- проверка комплектности;
- проверка целостности.

Проверка упаковки, маркировки и комплектности проводится методом визуального осмотра. При осмотре упаковки проверяется:

- наименование и адрес отправителя (изготовителя);
- отсутствие механических повреждений упаковки.

Если на упаковке имеются значительные повреждения, которые могут свидетельствовать о нарушении целостности упаковки и ее содержимого, а также о неправомерном вскрытии конверта третьими лицами, или отправителем не является Акционерное общество «Флант» (адрес: Акционерное общество «Флант», 115088, г. Москва, ул. Угрешская, д. 12, стр. 4, офис 47А) такой комплект поставки признается бракованным и отсылается обратно отправителю. Если по результатам проверки целостности упаковки нарушений выявлено не было – проводится проверка упаковки и маркировки.

Проверка маркировки подразумевает проверку наличия во вкладыше следующих данных:

- наименование изделия;
- децимальный номер;
- логотип предприятия-производителя;
- год выпуска;
- серийный номер;
- адрес, номер телефона, адрес электронной почты, ссылка на сайт предприятия-производителя;
- краткая информация об изделии и его основных функциональных возможностях.

На этапе приемки должна выполняться проверка заполнения и подписания ответственными разделов «Свидетельство о приемке» и «Свидетельство об упаковке и маркировке» в Формуляре, поставляемом на бумажном носителе.

Если по результатам проверки маркировки обнаружено отсутствие какого-либо элемента маркировки – такой комплект поставки признается бракованным и отсылается обратно отправителю. Если по результатам проверки маркировки нарушений выявлено не было – проводится проверка комплектности поставки.

Проверка комплектности поставки проводится методом сравнения состава полученной поставки требованиям, указанным в п. 4.1 Формуляра. Если по результатам проверки комплектности поставки расхождения с Формуляром отсутствуют – далее производится проверка целостности.

В рамках проверки целостности должны быть проведены:

- проверка целостности USB флэш-накопителя или DVD-диска (носителя);
- проверка целостности дистрибутива ПО «Deckhouse Platform»;
- проверка целостности установленного ПО «Deckhouse Platform».

Проверка целостности носителя выполняется путем визуального определения отсутствия каких-либо механических или иных повреждений. Если по результатам проверки на носителе обнаружены повреждения – такой комплект поставки признается бракованным и отсылается обратно отправителю.

Проверка целостности ПО «Deckhouse Platform» на носителе информации выполняется путем снятия контрольных сумм с помощью утилиты gostsum из состава сертифицированной ОС и сравнения их с контрольными суммами, приведенными в Электронном приложении к Формуляру (каталог «Электронные приложения» – «Deckhouse Platform. Формуляр» – «distr_cs.txt). Если по результатам проверки контрольная сумма дистрибутива, записанного на носителе информации, соответствует значению контрольной суммы, приведенной в Формуляре – требуется выполнить установку ПО «Deckhouse Platform» в соответствии с разделом 4 настоящего документа. Если по результатам проверки контрольная сумма дистрибутива ПО «Deckhouse Platform», записанного на USB флэш-накопителе информации или DVD-диске, не соответствует значению контрольной суммы, приведенной в Формуляре – такой комплект поставки признается бракованным и отсылается обратно отправителю.

Проверка целостности, установленной ПО «Deckhouse Platform» выполняется путем снятия контрольных сумм с исполняемых файлов с помощью программного обеспечения утилиты gostsum из состава сертифицированной ОС и сравнения полученных контрольных сумм с

контрольными суммами, приведенными в Электронном приложении к Формуляру (каталог «Электронные приложения» – «Deckhouse Platform. Формуляр» – «bins_cs.txt). Инструкция по расчету контрольных сумм приведена в Формуляре в разделе 4.3. Если по результатам проверки контрольные суммы исполняемых файлов, установленного ПО «Deckhouse Platform», соответствуют значениям контрольных сумм, приведенных в Формуляре – полученный экземпляр ПО «Deckhouse Platform» считается соответствующим оригиналу ПО «Deckhouse Platform». Если по результатам проверки контрольные суммы исполняемых файлов, установленного ПО «Deckhouse Platform», не соответствуют значениям контрольных сумм, приведенным в Формуляре – такой комплект поставки признается бракованным и отсылается обратно отправителю.

В случае, если по всем указанным процедурам установлено полное соответствие, ПО «Deckhouse Platform» признается годным к эксплуатации.

2 Действия по безопасной установке и настройке ПО «Deckhouse Platform»

При установке и настройке ПО «Deckhouse Platform» для обеспечения безопасности необходимо выполнение следующих условий:

- инсталляция ПО «Deckhouse Platform» должна осуществляться в защищенной инфраструктуре работником соответствующей квалификации, имеющим права администратора с присвоенными ему идентификационными данными (логин, пароль) для работы в среде функционирования ПО «Deckhouse Platform»;
- действия, проводимые при инсталляции ПО «Deckhouse Platform», а также при инициализации ПО «Deckhouse Platform», подлежат логированию (настройка сбора и отправки логов описана в п. 4.14);
- установка и конфигурирование ПО «Deckhouse Platform» должны осуществляться в соответствии с данным документом;
- должно обеспечиваться предотвращение несанкционированного доступа к идентификаторам и паролям пользователей сервиса и привилегированных пользователей (администраторов информационной безопасности) ПО «Deckhouse Platform»;
- должно обеспечиваться предотвращение несанкционированного доступа к идентификаторам и паролям администраторов среды виртуализации, которые необходимы для установки и настройки ПО «Deckhouse Platform».

3 Реализация функциональных возможностей средства согласно исполнениям ПО

В настоящем разделе представлено сопоставление исполнений ПО с функциональными возможностями, описанными в соответствующих разделах документа.

№ п/п	Разделы Руководства администратора	Исполнения
4	**Установка и настройка**	Kubernetes+Virtualization; Virtualization; Kubernetes
4.1	Проверки, выполняемые перед началом установки	Kubernetes+Virtualization; Virtualization; Kubernetes
4.2	Файл конфигурации установки ПО «Deckhouse Platform»	Kubernetes+Virtualization; Virtualization; Kubernetes
4.3	Установка ПО «Deckhouse Platform»	Kubernetes+Virtualization; Virtualization; Kubernetes
4.4	Настройка хранилищ	Kubernetes+Virtualization; Virtualization; Kubernetes
4.5	Откат установки и удаление ПО «Deckhouse Platform»	Kubernetes+Virtualization; Virtualization; Kubernetes
4.6	Создание образов	Kubernetes+Virtualization; Kubernetes
4.7	Настройка ПО «Deckhouse Platform»	Kubernetes+Virtualization; Kubernetes

№ п/п	Разделы Руководства администратора	Исполнения
4.8	Настройка модуля	Kubernetes+Virtualization; Kubernetes
4.9	Включение и отключение модуля	Kubernetes+Virtualization; Kubernetes
4.10	Подключение провайдера аутентификации	Kubernetes+Virtualization; Kubernetes
4.11	Экспорт данных	Kubernetes+Virtualization; Kubernetes
4.12	Обновление	Kubernetes+Virtualization; Virtualization; Kubernetes
4.13	Создание самоподписанного сертификата	Kubernetes+Virtualization; Kubernetes
4.14	Логирование	Kubernetes+Virtualization; Kubernetes
4.15	Виртуализация	Kubernetes+Virtualization; Virtualization
4.16	Миграция данных etcd	Kubernetes+Virtualization; Virtualization; Kubernetes
5	**Описание параметров (настроек) безопасности**	Kubernetes+Virtualization; Kubernetes
5.1	Настройка сканирования на уязвимости	Kubernetes+Virtualization; Kubernetes
5.2	Настройка политик безопасности	Kubernetes+Virtualization;

№ п/п	Разделы Руководства администратора	Исполнения
		Kubernetes
5.3	Настройка уведомлений о событиях безопасности на почту	Kubernetes+Virtualization; Kubernetes
5.4	Настройка доступа к журналам событий безопасности	Kubernetes+Virtualization; Kubernetes
5.5	Просмотр журналов событий безопасности	Kubernetes+Virtualization; Kubernetes
5.6	Хранение журналов событий безопасности	Kubernetes+Virtualization; Kubernetes; Virtualization
5.7	Контроль целостности	Kubernetes+Virtualization; Kubernetes; Virtualization
5.8	Управление информационными потоками	Kubernetes+Virtualization; Kubernetes
6	**Действия по реализации функций безопасности среды функционирования средства**	Kubernetes+Virtualization; Kubernetes; Virtualization

4 Установка и настройка

Перед установкой необходимо убедиться, что выполнены минимальные требования к аппаратным, программным средствам (Таблица 1), а сетевая инфраструктура отвечает требованиям, приведенным в Таблице 2.

Таблица 1 - Минимальные требования к аппаратным и программным средствам (для каждого исполнения ПО «Deckhouse Platform»)

Исполнения ПО «Deckhouse Platform»	**Требования к аппаратной части**	**Требования к программной части**
Kubernetes	− Архитектура процессора x86-64; − Не менее 4 ядер CPU; − Не менее 12 GB RAM; − Не менее 50 ГБ дискового пространства	− ОС РЕД ОС (не ниже 7.3); − ОС Astra Linux Special Edition (не ниже 1.7); − ОС Альт не ниже 8 СП (релиз не ниже 10); − Московская серверная операционная система (не ниже 15.5)
Virtualization	− Архитектура процессора x86-64; − Не менее 4 ядер CPU; − Не менее 12 GB RAM; − Не менее 50 ГБ дискового пространства; − Поддержка инструкций Intel-VT (VMX) или AMD-V (SVM) (для узлов, предназначенных для запуска виртуальных машин); − Быстрый диск (400+ IOPS) и дополнительные диски для программно-определяемого хранилища (для узлов,	− ОС РЕД ОС (не ниже 8); − ОС Astra Linux Special Edition не ниже 1.8.3 (ядро не ниже 6.12); − ОС Альт не ниже 8 СП (релиз не ниже 10); − Московская серверная операционная система (не ниже 15.5)

Исполнения ПО «Deckhouse Platform»	Требования к аппаратной части	Требования к программной части
	предназначенных для запуска виртуальных машин)	
Kubernetes+Virtualization	− Архитектура процессора x86-64; − Не менее 4 ядер CPU; − Не менее 12 GB RAM; − Не менее 50 ГБ дискового пространства; − Поддержка инструкций Intel-VT (VMX) или AMD-V (SVM) (для узлов, предназначенных для запуска виртуальных машин); − Быстрый диск (400+ IOPS) и дополнительные диски для программно-определяемого хранилища (для узлов, предназначенных для запуска виртуальных машин)	− ОС РЕД ОС (не ниже 8); − ОС Astra Linux Special Edition не ниже 1.8.3 (ядро не ниже 6.12); − ОС Альт не ниже 8 СП (релиз не ниже 10); − Московская серверная операционная система (не ниже 15.5)

Таблица 2 - Требования к обеспечению сетевого взаимодействия узлов кластера

Вид трафика	Порты
Трафик между master-узлами	• 2379, 2380/ TCP; • 4200-4201/TCP; • 4223/TCP
Трафик от master-узлов к узлам	• 22/TCP; • 10250 /TCP; • 4221/TCP; • 4227 /TCP

Трафик от узлов к master-узлам	● 4234/ UDP; ● 6443/TCP; ● 4203/TCP; ● 4219/TCP; ● 4222/TCP
Трафик между узлами	● ICMP; ● 4202-4218/TCP; ● 4218/TCP/UDP; ● 4220-4239/TCP; ● 4240-4299/TCP; ● 4287/UDP; ● 4295-4299/UDP; ● 7000-7999/TCP; ● 8469-8472/UDP
Внешний трафик к master-узлам	● 22/TCP; ● 6443/TCP
Внешний трафик к фронтенд-узлам	● 80, 443/TCP; ● 5416/UDP/TCP; ● 10256/TCP; ● 30000-32767/TCP
Внешний трафик для всех узлов	● 53/UDP/TCP; ● 123/UDP; ● 443 /TCP

4.1 Проверки, выполняемые перед началом установки

Перед установкой необходимо проверить следующие условия:

1. Общие требования для узлов кластера:

- ОС находится в списке поддерживаемых и соответствует требованиям (см. Таблицу 1);
- Пакеты ОС обновлены до последних доступных версий;
- Настроен SSH-доступ по ключу;
- Узел имеет доступ к хранилищу образов контейнеров Deckhouse Platform (доступ к приватному registry или проксирующему registry — согласно конфигурации кластера);
- Указанные в конфигурации установки данные аутентификации для хранилища контейнерных образов корректны;
- Значения параметров PublicDomainTemplate и clusterDomain не совпадают;
- Подсетевые диапазоны podSubnetCIDR и serviceSubnetCIDR не пересекаются;
- На узлах отсутствует пользователь ПО «Deckhouse Platform»;
- Узлы кластера, должны иметь уникальный hostname, соответствующий требованиям:
    - Длина не более 63 символов;
    - Состоит только из строчных букв;
    - Не содержит спецсимволов (допускаются символы `-` и `.`, при этом они не могут быть в начале или в конце имени).

2. Для узлов, предназначенных для запуска виртуальных машин (при использовании виртуализации), необходимо дополнительно выполнение следующих требований:
    - доступ к репозиториям пакетов используемой ОС;
    - HTTPS-доступ к хранилищу образов контейнеров ПО «Deckhouse Platform»;
    - Настроен SSH-доступ от машины для запуска установки.
3. Дополнительные требования для статического кластера:
    - В команде для запуска установки указывается только один параметр --ssh-host, обозначающий IP первого master-узла (аргумент команды `dhctl bootstrap`);
    - Выполнимо подключение по SSH с указанными ключами;
    - Выполнима установка SSH-туннеля до первого master-узла;
    - Узел, выбранный под master:
        - соответствует минимальным системным требованиям (Таблица 1);
        - имеет установленный пакетный менеджер (apt, apt-get,dnf,yum, rpm, which - зависит от выбранной ОС);
        - имеет доступ к системным репозиториям;
        - имеет установленный Python (не ниже 3.12.10);
    - Если в конфигурации указаны параметры прокси — хранилище контейнеров доступно через прокси;

- Открыт необходимый порт между хостом запуска установщика и сервером — порт 22/TCP;
- DNS разрешает localhost к IP-адресу 127.0.0.1.
- Пользователь, от имени которого выполняется установка (например, пользователь caps), имеет доступ к sudo;
- На сервере (ВМ) установлено время, синхронизированное с доверенным NTP-сервером.

4. Требования для узлов с Московской серверной операционной системой (Мос.ОС 15 Arbat):
   - добавить systemd.unified_cgroup_hierarchy=1 в GRUB_CMDLINE_LINUX в файле /etc/default/grub, после чего выполнить команду:

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

   - установить расширенные модули zypper install kernel-default-extra и выполнить команду:

```
modprobe  erofs
```

5. Требования к машине для запуска установки ПО «Deckhouse Platform» (ЭВМ, которую не планируется добавлять в кластер):
   - допустимые ОС: РЕД ОС (не ниже 7.3), Astra Linux Special Edition не ниже 1.7., Альт (не ниже 8 СП);
   - установленный Docker для запуска инсталлятора ПО «Deckhouse Platform»;
   - HTTP/HTTPS-доступ к хранилищу образов контейнеров ПО «Deckhouse Platform» (частное registry или проксирующее registry — согласно настройкам);
   - SSH-доступ по ключу к будущему master-узлу;
   - сетевой доступ до хоста master-узла по порту 22/TCP.

Справочник администратора, содержащий дополнительную информацию о ПО «Deckhouse Platform», приведен в электронном приложении к настоящему документу, каталог «Электронные приложения» - «Deckhouse Platform. Руководство администратора» - «Справочник администратора.pdf».

Для использования модуля virtualization необходимо установить ПО «Deckhouse Platform». Работа с модулем описана в п. 4.15. данного Руководства.

4.2 Файл конфигурации установки ПО «Deckhouse Platform»

Для установки ПО «Deckhouse Platform» нужно подготовить YAML-файл конфигурации установки.

Если в кластере будет использоваться внутренний TLS-сертификат, то необходимо также добавить его в конфигурацию установки, поместив в манифест секрета в пространстве имен d8-system.

YAML-файл конфигурации установки содержит манифесты ресурсов и обычно называется config.yml. Файл конфигурации установки может содержать, в том числе следующие манифесты ресурсов:

- InitConfiguration – начальные параметры конфигурации ПО «Deckhouse Platform». С этой конфигурацией ПО «Deckhouse Platform» запустится после установки.
- ClusterConfiguration – общие параметры кластера, такие как версия control plane, сетевые параметры, параметры CRI и т.д.

  Не изменяйте параметры serviceSubnetCIDR, podSubnetNodeCIDRPrefix, podSubnetCIDR в работающем кластере. Если изменение параметров необходимо — разверните новый кластер.
- StaticClusterConfiguration – ресурс для указания списка внутренних сетей узлов кластера, который используется для связи компонентов кластера между собой. Укажите, если узлы кластера имеют более одного сетевого интерфейса. Если на узлах кластера используется только один интерфейс, ресурс StaticClusterConfiguration можно не создавать.
- ModuleConfig – вид ресурсов, содержащих параметры глобальной конфигурации и конфигурации модулей Deckhouse.

4.2.1 Ресурс InitConfiguration

Version: deckhouse.io/v1

Конфигурация ПО «Deckhouse Platform», с которой он запустится после установки.

Пример конфигурации при использовании локального репозитория:

```
apiVersion: deckhouse.io/v1
kind: InitConfiguration
deckhouse:
  imagesRepo: registry.company.my/deckhouse/cse
  registryDockerCfg:
eyJhdXRocyI6IHsicmVnaXN0cnkuY29tcGFueS5teSI6IHsidXNlcm5hbWUiOiJ1c2VyIiwicGFzc3dvcmQiOi
JteS1wQHNzdzByZCIsImF1dGgiOiJkWE5sY2pwdGVTMXdRSE56ZHpCeVpBbz0ifX19Cg==
  registryScheme: HTTPS
```

```
registryCA: |
  -----BEGIN CERTIFICATE-----
  ...
  -----END CERTIFICATE-----
```

- apiVersion *строка*

  Обязательный параметр

  Используемая версия API ПО «Deckhouse Platform».

  Допустимые значения: deckhouse.io/v1, deckhouse.io/v1alpha1

- deckhouse *объект*

  Обязательный параметр

  Параметры, необходимые для установки ПО «Deckhouse Platform».

  - deckhouse.imagesRepo строка

    Адрес container registry с образами ПО «Deckhouse Platform».

    Обязательное поле.

  - deckhouse.registryCA *строка*

    Корневой сертификат, которым можно проверить сертификат container registry при работе по HTTPS (если registry использует самоподписанные SSL-сертификаты).

  - deckhouse.registryDockerCfg *строка*

    Строка с правами доступа к container registry, зашифрованная в Base64.

    Обязательное поле.

  - deckhouse.registryScheme *строка*

    Протокол доступа к container registry (HTTP или HTTPS).

    По умолчанию: "HTTPS"

    Допустимые значения: HTTP, HTTPS

- kind *строка*

  Обязательный параметр

  Допустимые значения: InitConfiguration

4.2.2 Ресурс ClusterConfiguration

Ресурс ClusterConfiguration описывает общие параметры кластера.

Определяет, например, сетевые параметры, параметры CRI, версию control plane и т.д. Некоторые параметры можно изменять после развертывания кластера, во время его работы.

Чтобы изменить содержимое ресурса ClusterConfiguration в работающем кластере, выполните следующую команду (требуется файл конфигурации подключения к кластеру

(kubeconfig) и установленная утилита d8 (поставляется в составе ПО «Deckhouse Platform»), либо выполняйте команды на master-узле):

```
d8 k -n d8-system exec -ti deploy/deckhouse -- deckhouse-controller edit
cluster-configuration
```

Пример:

```
apiVersion: deckhouse.io/v1
kind: ClusterConfiguration
podSubnetNodeCIDRPrefix: '24'
podSubnetCIDR: 10.244.0.0/16
serviceSubnetCIDR: 192.168.0.0/16
kubernetesVersion: "Automatic"
clusterDomain: "cluster.local"
clusterType: Static
#proxy:
#  httpProxy: https://user:password@proxy.company.my:8443
#  httpsProxy: https://user:password@proxy.company.my:8443
#  noProxy:
#  - company.my
```

Здесь:

- apiVersion *строка*

  Обязательный параметр

  Используемая версия API ПО «Deckhouse Platform».

  Допустимые значения: deckhouse.io/v1, deckhouse.io/v1alpha1

- clusterDomain *строка*

  Обязательный параметр

  Домен кластера (используется для маршрутизации внутри кластера).

  По умолчанию: "cluster.local"

- clusterType *строка*

  Обязательный параметр

  Тип инфраструктуры кластера. Всегда - Static

- defaultCRI *строка*

  Тип container runtime, используемый на узлах кластера (в NodeGroup'ax) по умолчанию.

  Если используется значение `NotManaged`, то Deckhouse не будет управлять (устанавливать и настраивать) container runtime.

  В этом случае образы, используемые в NodeGroup'ax, должны содержать уже установленный container runtime.

  Если установлено значение `ContainerdV2`, будет использоваться `CgroupsV2` (обеспечивает улучшенную безопасность и управление ресурсами). Для использования

`ContainerdV2` в качестве container runtime узлы кластера должны соответствовать следующим требованиям:

- поддержка `CgroupsV2`;
- ядро Linux версии `5.8` и новее;
- systemd версии `244` и новее;
- поддержка модуля ядра `erofs`.
- версия ОС из допустимого списка (п. 4.12.1.*)*

!Подробнее о миграции на Containerd V2 – в п. 4.12.1.

По умолчанию: "Containerd"

Допустимые значения: Containerd, ContainerdV2, NotManaged

- kind *строка*

  Обязательный параметр

  Допустимые значения: ClusterConfiguration

- kubernetesVersion *строка*

  Обязательный параметр

  Версия control plane кластера Kubernetes.

  Допустимые значения: 1.29, 1.31, Automatic

- podSubnetCIDR *строка*

  Обязательный параметр

  Адресное пространство подов кластера.

- podSubnetNodeCIDRPrefix *строка*

  Префикс сети подов на узле.

  Внимание! Не меняйте параметр в уже развернутом кластере.

  По умолчанию: "24"

- proxy *объект*

  Глобальная настройка proxy-сервера.

  Внимание! *Для того, чтобы избежать использования прокси в запросах между компонентами кластера, важно заполнить параметр noProxy списком подсетей, которые используются на узлах.*

  - proxy.httpProxy *строка*

    URL proxy-сервера для HTTP-запросов.

    При необходимости укажите имя пользователя, пароль и порт proxy-сервера.

    Шаблон: ^https?://[0-9a-zA-Z\.\-:@]+$

Примеры:

```
httpProxy: http://proxy.company.my
httpProxy: https://user:password@proxy.company.my:8443
```

- proxy.httpsProxy *строка*

  URL proxy-сервера для HTTPS-запросов.

  При необходимости укажите имя пользователя, пароль и порт proxy-сервера.

  Шаблон: ^https?://[0-9a-zA-Z\.\-:@]+$

  Примеры:

  httpsProxy: http://proxy.company.my

  httpsProxy: https://user:password@proxy.company.my:8443

- proxy.noProxy *массив строк*

  Список IP и доменных имен, для которых проксирование не применяется.

  Для настройки wildcard-доменов используйте написание вида “.example.com”.

  Шаблон: ^[a-z0-9\-\./]+$

- serviceSubnetCIDR *строка*

  Обязательный параметр

  Адресное пространство для service’ов кластера.

4.2.3 Ресурс StaticClusterConfiguration

Version: deckhouse.io/v1

Дополнительные параметры кластера.

Пример:

```
apiVersion: deckhouse.io/v1
kind: StaticClusterConfiguration
internalNetworkCIDRs:
- 10.244.0.0/16
- 10.50.0.0/16
```

Здесь:

- apiVersion *строка*

  Обязательный параметр.

  Используемая версия API Deckhouse.

  Допустимые значения: deckhouse.io/v1, deckhouse.io/v1alpha1

- internalNetworkCIDRs *массив строк*

4.2.4 Список внутренних сетей узлов кластера

Внутренние сети используются для связи компонентов Kubernetes (kube-apiserver, kubelet и т.д.) между собой.

Если каждый узел в кластере имеет только один сетевой интерфейс, то параметр можно не указывать и ресурс StaticClusterConfiguration можно не создавать.

- Элемент массива *строка*

  Шаблон:

  ^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\/(3[0-2]|[1-2][0-9]|[0-9]))$

  Пример:

  192.168.42.0/24

- Kind *строка*

  Обязательный параметр

  Допустимые значения: StaticClusterConfiguration

4.3 Установка ПО «Deckhouse Platform»

4.3.1 Пример конфигурации установки

Пример конфигурации установки (измените параметры, специфичные для вашей инфраструктуры). Приведенный пример конфигурации установки настраивает ПО «Deckhouse Platform» для выполнения заявленных функций безопасности:

```
apiVersion: deckhouse.io/v1
kind: InitConfiguration
deckhouse:
 imagesRepo: registry.company.my/deckhouse/cse
 # строку с данными аутентификации в хранилище можно получить командой:
 # echo '{"auths": {"<REGISTRY_HOST>": {"auth": "'$(echo -n
'<REGISTRY_USER>:<REGISTRY_PASSWORD>' | base64 -w0)'"}}}' | base64 -w0
 registryDockerCfg: <ДАННЫЕ_АУТЕНТИФИКАЦИИ_В_ХРАНИЛИЩЕ_ОБРАЗОВ>
 registryScheme: HTTPS
 registryCA: |
   -----BEGIN CERTIFICATE-----

   -----END CERTIFICATE-----
---
apiVersion: deckhouse.io/v1
kind: ClusterConfiguration
# Адресное пространство подов кластера.
podSubnetCIDR: 10.111.0.0/16
# Адресное пространство сервисов кластера.
serviceSubnetCIDR: 10.222.0.0/16
# Версия control plane кластера Kubernetes.
kubernetesVersion: "Automatic"
```

```
# Домен кластера (используется для маршрутизации внутри кластера).
# Не должен совпадать с доменом, указанным
# в глобальном параметре publicDomainTemplate.
clusterDomain: cluster.local
clusterType: Static
defaultCRI: ContainerdV2
# При использовании Containerd v1 режим контроля подписи Enforce
# применяется только после предварительного включения режима Migrate и проведения
# миграций (см. п. 4.16).
---
apiVersion: deckhouse.io/v1
kind: StaticClusterConfiguration
internalNetworkCIDRs:
- <NODES_NETWORK>
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 name: deckhouse
spec:
 version: 1
 enabled: true
 settings:
   releaseChannel: LTS
   logLevel: Info
   bundle: Default
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 name: cert-manager
spec:
 version: 1
 enabled: true
 settings:
   disableLetsencrypt: true
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 name: control-plane-manager
spec:
 version: 2
 enabled: true
 settings:
   apiserver:
     signature: Enforce
# При использовании Containerd v1 режим контроля подписи Enforce
# применяется только после предварительного включения режима Migrate и проведения
# миргаций
# (см. раздел 4.16).
     encryptionEnabled: true
     auditPolicyEnabled: true
---
apiVersion: v1
data:
 tls.crt: <INTERNAL_CA_CERT>
 tls.key: <INTERNAL_CA_KEY>
kind: Secret
metadata:
```

```
 name: internal-ca-key-pair
 namespace: d8-cert-manager
type: Opaque
---
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
 name: internal-ca
spec:
 ca:
   secretName: internal-ca-key-pair
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 name: global
spec:
 version: 2
 enabled: true
 settings:
   modules:
     https:
       mode: CertManager
       certManager:
         # Указывается имя ClusterIssuer,
         # который будет использоваться для выпуска сертификатов.
         clusterIssuerName: internal-ca
     # Укажите шаблон DNS-имен кластера
     publicDomainTemplate: "%s.kube.local"
   defaultClusterStorageClass: local-path
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 name: cni-cilium
spec:
 version: 1
 enabled: true
 settings:
   tunnelMode: VXLAN
   policyAuditMode: false
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 name: cilium-hubble
spec:
 version: 2
 enabled: true
 settings:
   auth:
     allowedUserGroups:
       - admins
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 name: runtime-audit-engine
spec:
 version: 1
```

```
 enabled: true
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 name: gost-integrity-controller
spec:
 version: 1
 enabled: true
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 name: admission-policy-engine
spec:
 version: 1
 enabled: true
 settings:
   podSecurityStandards:
     enforcementAction: Deny
     defaultPolicy: Baseline
   denyVulnerableImages:
     enabled: true
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 name: operator-trivy
spec:
 version: 1
 enabled: true
 settings:
   linkCVEtoBDU: true
   severities:
     - UNKNOWN
     - LOW
     - MEDIUM
     - HIGH
     - CRITICAL
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 name: operator-prometheus
spec:
 version: 1
 enabled: true
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 name: log-shipper
spec:
 version: 1
 enabled: true
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 name: ingress-nginx
```

```
spec:
 version: 1
 enabled: true
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 name: monitoring-kubernetes
spec:
 version: 1
 enabled: true
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 name: node-manager
spec:
 enabled: true
 version: 2
 settings:
   earlyOomEnabled: false
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 name: kube-dns
spec:
 enabled: true
 version: 1
 settings:
   enableLogs: true
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 name: loki
spec:
 enabled: true
 settings:
   storageClass: local-path
   retentionPeriodHours: 24
   storeSystemLogs: true
 version: 1
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 name: prometheus
spec:
 enabled: true
 settings:
   auth:
     allowedUserGroups:
       - admins
   storageClass: local-path
   longtermStorageClass: local-path
 version: 2
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
```

```
metadata:
 name: user-authz
spec:
 enabled: true
 settings:
   enableMultiTenancy: true
 version: 1
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 name: console
spec:
 version: 1
 enabled: true
 settings:
   auth:
     allowedUserGroups:
       - admins
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 annotations:
 name: user-authn
spec:
 enabled: true
 settings:
   controlPlaneConfigurator:
     dexCAMode: FromIngressSecret
   publishAPI:
     enabled: true
 version: 2
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 name: chrony
spec:
 enabled: true
 settings:
   ntpServers:
     - <NTP_SERVER.1>
     - <NTP_SERVER.2>
     - <NTP_SERVER.3>
 version: 1
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
 name: istio
spec:
 version: 3
 enabled: false
 settings:
   auth:
     allowedUserGroups:
       - admins
---
apiVersion: v1
```

```
data:
 audit-policy.yaml:
YXBpVmVyc2lvbjogYXVkaXQuazhzLmlvL3YxICMgVGhpcyBpcyByZXF1aXJlZC4Ka2luZDogUG9saWN5CiMgRG
9uJ3QgZ2VuZXJhdGUgYXVkaXQgZXZlbnRzIGZvciBhbGwgcmVxdWVzdHMgaW4gUmVxdWVzdFJlY2VpdmVkIHN0
YWdlLgpvbWl0U3RhZ2VzOgogIC0gIlJlcXVlc3RSZWNlaXZlZCIKcnVsZXM6CiAgIyBBIGNhdGNoLWFsbCBydW
xlIHRvIGxvZyBhbGwgb3RoZXIgcmVxdWVzdHMgYXQgdGhlIE1ldGFkYXRhIGxldmVsLgogIC0gbGV2ZWw6IE1l
dGFkYXRhCiAgICAjIExvbmctcnVubmluZyByZXF1ZXN0cyBsaWtlIHdhdGNoZXMgdGhhdCBmYWxsIHVuZGVyIH
RoaXMgcnVsZSB3aWxsIG5vdAogICAgIyBnZW5lcmF0ZSBhbiBhdWRpdCBldmVudCBpbiBSZXF1ZXN0UmVjZWl2
ZWQuCiAgICBvbWl0U3RhZ2VzOgogICAgICAtICJSZXF1ZXN0UmVjZWl2ZWQiCgo=
kind: Secret
metadata:
 name: audit-policy
 namespace: kube-system
type: Opaque
---
apiVersion: deckhouse.io/v1alpha1
kind: NodeGroupConfiguration
metadata:
 name: sysctl-tune-fstec
spec:
 weight: 100
 bundles:
 - "*"
 nodeGroups:
 - "*"
 content: |
   sysctl -w kernel.dmesg_restrict=1
   sysctl -w kernel.kptr_restrict=2
   sysctl -w net.core.bpf_jit_harden=2
   sysctl -w kernel.perf_event_paranoid=3
   sysctl -w kernel.kexec_load_disabled=1
   sysctl -w user.max_user_namespaces=0
   sysctl -w kernel.unprivileged_bpf_disabled=1
   sysctl -w vm.unprivileged_userfaultfd=0
   sysctl -w dev.tty.ldisc_autoload=0
   sysctl -w vm.mmap_min_addr=4096
   sysctl -w kernel.randomize_va_space=2
   sysctl -w kernel.yama.ptrace_scope=3
   sysctl -w fs.protected_symlinks=1
   sysctl -w fs.protected_hardlinks=1
   sysctl -w fs.protected_fifos=2
   sysctl -w fs.protected_regular=2
   sysctl -w fs.suid_dumpable=0
---
# Настройки Ingress-контроллера.
# Измените согласно требованиям вашей инфраструктуры.
apiVersion: deckhouse.io/v1
kind: IngressNginxController
metadata:
 name: main
spec:
 ingressClass: nginx
 inlet: HostWithFailover
 nodeSelector:
   node-role.kubernetes.io/master: ""
 tolerations:
 - effect: NoSchedule
   operator: Exists
---
apiVersion: deckhouse.io/v1alpha1
```

```
kind: LocalPathProvisioner
metadata:
 name: local-path
spec:
 # В зависимости от конфигурации ОС путь может быть изменён.
 path: "/opt/local-path-provisioner"
 reclaimPolicy: Delete
---
apiVersion: deckhouse.io/v1
kind: User
metadata:
 name: admin
spec:
 email: admin@deckhouse.ru
 # echo '<ПАРОЛЬ>' | htpasswd -BinC 10 "" | cut -d: -f2 | base64 -w0
 password: '<GENERATED_PASSWORD_HASH>'
---
apiVersion: deckhouse.io/v1alpha1
kind: Group
metadata:
 name: admins
spec:
 name: admins
 members:
   - kind: User
     name: admin
---
apiVersion: deckhouse.io/v1
kind: ClusterAuthorizationRule
metadata:
 name: admin
spec:
 subjects:
   - kind: Group
     name: admins
 accessLevel: SuperAdmin
 portForwarding: true
 namespaceSelector:
   matchAny: true
---
apiVersion: deckhouse.io/v1
kind: NodeGroup
metadata:
 name: master
spec:
 nodeTemplate:
   labels:
     node-role.kubernetes.io/control-plane: ""
     node-role.kubernetes.io/master: ""
   taints:
     - effect: NoSchedule
       key: node-role.kubernetes.io/control-plane
 nodeType: Static
 staticInstances:
   count: 2
   labelSelector:
     matchLabels:
       role: master
---
apiVersion: deckhouse.io/v1
```

```
kind: NodeGroup
metadata:
 name: worker
spec:
 nodeType: Static
 staticInstances:
   count: 3
   labelSelector:
     matchLabels:
       role: worker
---
# SSH данные для доступа к хостам
apiVersion: deckhouse.io/v1alpha1
kind: SSHCredentials
metadata:
 name: ssh-credentials
spec:
 user: caps
 privateSSHKey: '<SSH_PRIVATE_KEY>'
---
apiVersion: deckhouse.io/v1alpha1
kind: StaticInstance
metadata:
 name: cse-master-2
 labels:
   role: master
spec:
 address: '<MASTER_2_NODE_IP>'
 credentialsRef:
   kind: SSHCredentials
   name: ssh-credentials
---
apiVersion: deckhouse.io/v1alpha1
kind: StaticInstance
metadata:
 name: cse-master-3
 labels:
   role: master
spec:
 address: '<MASTER_3_NODE_IP>'
 credentialsRef:
   kind: SSHCredentials
   name: ssh-credentials
---
apiVersion: deckhouse.io/v1alpha1
kind: StaticInstance
metadata:
 name: cse-worker-1
 labels:
   role: worker
spec:
 address: '<WORKER_1_NODE_IP>'
 credentialsRef:
   kind: SSHCredentials
   name: ssh-credentials
---
apiVersion: deckhouse.io/v1alpha1
kind: StaticInstance
metadata:
 name: cse-worker-2
```

```
 labels:
   role: worker
spec:
 address: '<WORKER_2_NODE_IP>'
 credentialsRef:
   kind: SSHCredentials
   name: ssh-credentials
---
apiVersion: deckhouse.io/v1alpha1
kind: StaticInstance
metadata:
 name: cse-worker-3
 labels:
   role: worker
spec:
 address: '<WORKER_3_NODE_IP>'
 credentialsRef:
   kind: SSHCredentials
   name: ssh-credentials
---
apiVersion: deckhouse.io/v1
kind: CustomPrometheusRules
metadata:
 name: falco-critical-alerts
spec:
 groups:
 - name: falco-critical-alerts
   rules:
   - alert: FalcoCriticalAlertsAreFiring
     for: 1m
     annotations:
       description: |
         There is a suspicious activity on a node {{ $labels.node }}.
         Check you events journal for more details.
       summary: Falco detects a critical security incident
     expr: |
       sum by (node)
(rate(falco_events{priority=~"error|critical|warning|notice"}[5m]) > 0)
```

4.3.2 Развёртывание registry

Для установки ПО «Deckhouse Platform» требуется наличие в локальной сети хранилища образов контейнеров (container registry). В случае отсутствия registry, выполните шаги по его установке:

Установите на сервер пакеты ОС docker-registry и apache2-utils.

Пример для ОС с менеджером пакетов apt:

```
apt install docker-registry apache2-utils
```

Пример для ОС с менеджером пакетов dnf:

```
dnf install -y docker-registry httpd-tools
```

Сгенерируйте пользователя для доступа в registry (укажите имя пользователя и пароль, в примере используется пользователь ПО «Deckhouse Platform»):

```
htpasswd -bnB deckhouse deckhouse > /etc/docker/registry/htpasswd
```

Отредактируйте конфигурацию registry в файле /etc/docker/registry/config.yml и измените параметр auth.htpasswd.path с /etc/docker/registry на /etc/docker/registry/htpasswd

Поместите ключ и TLS-сертификат в следующие файлы:

/etc/docker/registry/private.key # Ключ

/etc/docker/registry/public.crt # Сертификат

При отсутствии TLS-сертификатов, можно сгенерировать самоподписанные сертификаты (см п.4.13.)

Обновите конфигурацию локального registry:

```
cat <<EOF > /etc/docker/registry/config.yml
version: 0.1
log:
  fields:
    service: registry
storage:
  cache:
    blobdescriptor: inmemory
  filesystem:
    rootdirectory: /var/lib/docker-registry
  delete:
    enabled: true
http:
  addr: :443
  tls:
    certificate: /etc/docker/registry/public.crt
    key: /etc/docker/registry/private.key
  secret: deckhouse
  headers:
    X-Content-Type-Options: [nosniff]
auth:
  htpasswd:
    realm: basic-realm
    path: /etc/docker/registry/htpasswd
health:
  storagedriver:
    enabled: true
    interval: 10s
    threshold: 3
EOF
```

Добавьте права на чтение приватного ключа сертификата пользователю docker-registry.

```
setfacl -m u:docker-registry:r /etc/docker/registry/private.key
```

Добавьте права на исполняемый файл docker-registry, чтобы он смог использовать системный порт 443:

```
setcap 'cap_net_bind_service=+ep' /usr/bin/docker-registry
```

Перезапустите сервис docker-registry:

```
systemctl restart docker-registry
```

Проверьте работоспособность локального репозитория, например, с помощью следующей команды (укажите пароль пользователя ПО «Deckhouse Platform», который задавали на предыдущих шагах, и адрес registry):

```
curl -u deckhouse:<PASSWORD> -v https://<REGISTRY_HOST>/v2/
```

Используйте адрес registry при загрузке образов перед установкой ПО «Deckhouse Platform» (п. 4.3). Пример:

```
d8 mirror push /root/d8-bundle https://10.128.0.37:5000/deckhouse \
  --registry-login=deckhouse \
  --registry-password=deckhouse
```

Если registry работает по протоколу HTTP, то добавьте параметр --insecure.

В registry ПО «Deckhouse Platform» загружается с поставляемого USB-флеш накопителя или DVD-диска, входящего в комплект поставки, согласно п.3.3 Технических условий RU.86432418.00001-01 ТУ 04-1.

4.3.3 Установка ПО «Deckhouse Platform»

Подготовьте рабочую станцию (хост установки) и master-узел будущего кластера. Минимальные требования к программным и аппаратным средствам представлены в Таблице 1. Порядок подготовки рабочего пространства к установке ПО «Deckhouse Platform» представлен в Разделе 4.

Скопируйте файлы поставки с USB-флеш накопителя или DVD-диска на компьютер, с которого есть доступ до хранилища образа контейнеров.

Также, вы можете скопировать образы их из публичного хранилища образов контейнеров следующей командой:

```
d8 mirror pull \
${PACKAGE_DIR_PATH} \
--source="registry-cse.deckhouse.ru/deckhouse/cse" \
--license="${LICENSE_KEY}" \
--gost-digest \
--deckhouse-tag="v1.73.2" \
--include-module=stronghold
```

Где

${PACKAGE_DIR_PATH} - директория для скаченных образов

${LICENSE_KEY} - лицензионный ключ для доступа в публичное хранилище образов контейнеров ПО «Deckhouse Platform».

Результат выполнения команды - формирование директории со следующим содержимым

```
d8-bundle/
├── module-stronghold.tar
├── module-stronghold.tar.gostsum
├── platform.tar
├── platform.tar.gostsum
├── security.tar
└── security.tar.gostsum
```

С помощью команды `d8 tools gostsum` (утилита из состава сертифицированной ОС по алгоритму ГОСТ Р 34.11-2012 (длина хэш-кода 256 бит), https://wiki.astralinux.ru/pages/viewpage.action?pageId=3277020) получите контрольную сумму файлов d8 и файла архива образов (например, platform.tar) и сравните полученные результаты соответственно с содержимым файлов с суффиксом .gostsum (d8.gostsum для d8 и, например, platform.tar.gostsum для platform.tar из состава поставки). Полученные результаты расчета контрольных сумм и контрольные суммы в указанных файлах поставки должны совпадать.

Загрузите данные в хранилище образов контейнеров, выполнив следующую команду (измените путь к файлу архива или папке образов):

```
d8 mirror push <PATH> <REGISTRY_URL>/<REGISTRY_PATH> \
  --registry-login=<USERNAME> --registry-password=<PASSWORD>
```

Здесь:

- <PATH> - директория поставки, содержащая архивы с образами поставки ПО «Deckhouse Platform»;
- <REGISTRY_URL> - адрес хранилища образов контейнеров в локальной сети;
- <REGISTRY_PATH> - путь в хранилище образов контейнеров, в который будут загружаться образы ПО «Deckhouse Platform». В примерах ниже будет использоваться путь /deckhouse/cse;
- <USERNAME> - имя пользователя для авторизации в хранилище образов контейнеров;
- <PASSWORD> - пароль пользователя для авторизации в хранилище образов контейнеров.

Если ваш container registry допускает анонимный доступ без авторизации, не указывайте параметры --registry-login и --registry-password.

Подробнее загрузка образов в registry рассмотрена в п. 4.3.2. (загрузка образов в изолированный registry).

Создайте SSH-ключ для доступа к узлам кластера:

```
ssh-keygen -t rsa -f "$HOME/.ssh/id_rsa" -C "" -N ""
cat "$HOME/.ssh/id_rsa.pub"
```

Добавьте публичную часть ключа на все узлы кластера ПО «Deckhouse Platform», в данном примере на узлах создается пользователь caps:

```
# Укажите публичную часть SSH-ключа пользователя.
export KEY='<SSH-PUBLIC-KEY>'
useradd -m -s /bin/bash caps
usermod -aG sudo caps
echo 'caps ALL=(ALL) NOPASSWD: ALL' | sudo EDITOR='tee -a' visudo
mkdir /home/caps/.ssh
echo $KEY >> /home/caps/.ssh/authorized_keys
chown -R caps:caps /home/caps
chmod 700 /home/caps/.ssh
chmod 600 /home/caps/.ssh/authorized_keys

# для Astra Linux - максимальный уровень целостности для пользователя caps
pdpl-user -i 63 caps
```

В операционных системах на базе RHEL (Red Hat Enterprise Linux), таких как РЕД ОС, пользователя caps нужно добавлять в группу wheel. Для этого выполните следующую команду, указав публичную часть SSH-ключа:

```
# Укажите публичную часть SSH-ключа пользователя.
export KEY='<SSH-PUBLIC-KEY>'
useradd -m -s /bin/bash caps
usermod -aG wheel caps
echo 'caps ALL=(ALL) NOPASSWD: ALL' | sudo EDITOR='tee -a' visudo
mkdir /home/caps/.ssh
echo $KEY >> /home/caps/.ssh/authorized_keys
chown -R caps:caps /home/caps
chmod 700 /home/caps/.ssh
chmod 600 /home/caps/.ssh/authorized_keys
```

Запустите контейнер установщика:

```
docker run --pull=always -it [<MOUNT_OPTIONS>] <REGISTRY_URL>/<REGISTRY_PATH>/install:v1.73.2 bash
```

Здесь:

- <REGISTRY_URL> – адрес container registry с образами ПО «Deckhouse Platform».
- <REGISTRY_PATH> - путь в хранилище образов контейнеров, в который были загружены образы ПО «Deckhouse Platform». В примерах ниже будет использоваться путь /deckhouse/cse;
- v1.73.2 – версия установщика;
- <MOUNT_OPTIONS> – параметры монтирования файлов в контейнер инсталлятора, таких как:
    - SSH-ключи доступа
    - файл конфигурации
    - файл ресурсов и т.д.

Пример запуска контейнера инсталлятора:

```
docker run -it --pull=always \
  -v "$PWD/config.yaml:/config.yaml" \
  -v "$PWD/dhctl-tmp:/tmp/dhctl" \
  -v "$HOME/.ssh/:/tmp/.ssh/" registry.company.int/deckhouse/cse/install:v1.73.2
bash
```

Установка ПО «Deckhouse Platform» запускается в контейнере инсталлятора с помощью команды `dhctl`.

Для запуска установки ПО «Deckhouse Platform» используйте команду:

```
dhctl bootstrap
```

Для получения справки по параметрам выполните:

```
dhctl bootstrap -h
```

Пример запуска установки ПО «Deckhouse Platform»:

```
dhctl bootstrap --ssh-user=<SSH_USER> --ssh-host=<MASTER_HOST>
--ssh-agent-private-keys=/tmp/.ssh/id_rsa --config=/config.yml --ask-become-pass
```

Здесь:

- /config.yml — файл конфигурации установки;
- <SSH_USER> — пользователь на сервере для подключения по SSH;
- <MASTER_HOST> — адрес сервера для подключения по SSH;
- --ssh-agent-private-keys – файл приватного SSH-ключа, для подключения по SSH.
- --ask-become-pass - параметр указывает, что пользователю на узлах кластера ПО «Deckhouse Platform» необходимо вводить пароль при использовании sudo; если пользователю на узлах кластера sudo доступен без пароля, то данный параметр указывать не нужно.

Дождитесь успешного завершения установки

Пример успешного завершения:

```
...
└ ⛵ ~ Bootstrap: Create Resources (290.81 seconds)

┌ ⛵ ~ Bootstrap: Run post bootstrap actions
│ ┌ Create deckhouse release for version v1.73
│ │ 🎉 Succeeded!
│ └ Create deckhouse release for version v1.73 (0.01 seconds)
└ ⛵ ~ Bootstrap: Run post bootstrap actions (0.01 seconds)

┌ ⛵ ~ Bootstrap: Clear cache
│ ❗ ~ Next run of "dhctl bootstrap" will create a new Kubernetes cluster.
└ ⛵ ~ Bootstrap: Clear cache (0.00 seconds)
```

```
🎉 Deckhouse cluster was created successfully!
```

Проверьте состояние очереди, выполнив команду на master-узле:

```
dhctl system queue list
```

В очереди не должно быть задач на выполнение.

Пример вывода:

```
$ d8 system queue list
Summary:
- 'main' queue: empty.
- 91 other queues (0 active, 91 empty): 0 tasks.
- no tasks to handle.
```

После успешной установки первого master-узла необходимо выполнить дальнейшую настройку кластера в зависимости от выбранного сценария развертывания..

Сценарий 1. Если планируется развернуть кластер для тестирования, состоящий из одного master-узла, то необходимо выполнить команду, которая снимает ограничение на размещение системных компонентов ПО «Deckhouse Platform» на master-узлах.

Для этого на master-узле выполните следующую команду:

```
d8 k patch nodegroup master --type json -p '[{"op": "remove", "path":
"/spec/nodeTemplate/taints"}]'
```

Сценарий 2. Если планируется развернуть полноценный отказоустойчивый кластер, состоящий из трех master-узлов, двух system-узлов, двух frontend-узлов и одного worker-узла, снимать ограничения на master-узлах не требуется.

Для подготовки добавления узлов используйте приведённую ниже команду. Если какие-либо группы узлов не требуются, удалите описания соответствующих NodeGroup и StaticInstance.

Выполните на master-узле следующую команду (измените необходимые параметры):

```
d8 k apply -f - << EOF
# Данные подключения по SSH для доступа к хостам узлов кластера
apiVersion: deckhouse.io/v1alpha1
kind: SSHCredentials
metadata:
```

```
  name: ssh-credentials
spec:
  # измените имя пользователя, если у вас он другой
  user: caps
  privateSSHKey: '<SSH_PRIVATE_KEY>'
---
apiVersion: deckhouse.io/v1
kind: NodeGroup
metadata:
  name: master
spec:
  nodeTemplate:
    labels:
      node-role.kubernetes.io/control-plane: ""
      node-role.kubernetes.io/master: ""
    taints:
      - effect: NoSchedule
        key: node-role.kubernetes.io/control-plane
  nodeType: Static
  staticInstances:
    count: 2
    labelSelector:
      matchLabels:
        role: master
---
apiVersion: deckhouse.io/v1
kind: NodeGroup
metadata:
  name: system
spec:
  nodeTemplate:
    labels:
      node-role.deckhouse.io/system: ""
    taints:
      - effect: NoExecute
        key: dedicated.deckhouse.io
        value: system
  nodeType: Static
  staticInstances:
    count: 2
    labelSelector:
      matchLabels:
        role: system
---
apiVersion: deckhouse.io/v1
kind: NodeGroup
metadata:
  name: frontend
spec:
  nodeTemplate:
    labels:
      node-role.deckhouse.io/frontend: ""
    taints:
      - effect: NoExecute
        key: dedicated.deckhouse.io
        value: frontend
  nodeType: Static
  staticInstances:
    count: 2
    labelSelector:
```

```
      matchLabels:
        role: frontend
---
apiVersion: deckhouse.io/v1
kind: NodeGroup
metadata:
  name: worker
spec:
  nodeType: Static
  staticInstances:
    count: 1
    labelSelector:
      matchLabels:
        role: worker
---
apiVersion: deckhouse.io/v1alpha1
kind: StaticInstance
metadata:
  name: master-2
  labels:
    role: master
spec:
  address: '<MASTER_2_NODE_IP>'
  credentialsRef:
    kind: SSHCredentials
    name: ssh-credentials
---
apiVersion: deckhouse.io/v1alpha1
kind: StaticInstance
metadata:
  name: master-3
  labels:
    role: master
spec:
  address: '<MASTER_3_NODE_IP>'
  credentialsRef:
    kind: SSHCredentials
    name: ssh-credentials
---
apiVersion: deckhouse.io/v1alpha1
kind: StaticInstance
metadata:
  name: system-1
  labels:
    role: system
spec:
  address: '<SYSTEM_1_NODE_IP>'
  credentialsRef:
    kind: SSHCredentials
    name: ssh-credentials
---
apiVersion: deckhouse.io/v1alpha1
kind: StaticInstance
metadata:
  name: system-2
  labels:
    role: system
spec:
  address: '<SYSTEM_2_NODE_IP>'
  credentialsRef:
```

```
    kind: SSHCredentials
    name: ssh-credentials
---
apiVersion: deckhouse.io/v1alpha1
kind: StaticInstance
metadata:
  name: frontend-1
  labels:
    role: frontend
spec:
  address: '<FRONTEND_1_NODE_IP>'
  credentialsRef:
    kind: SSHCredentials
    name: ssh-credentials
---
apiVersion: deckhouse.io/v1alpha1
kind: StaticInstance
metadata:
  name: frontend-2
  labels:
    role: frontend
spec:
  address: '<FRONTEND_2_NODE_IP>'
  credentialsRef:
    kind: SSHCredentials
    name: ssh-credentials
---
apiVersion: deckhouse.io/v1alpha1
kind: StaticInstance
metadata:
  name: worker-1
  labels:
    role: worker
spec:
  address: '<WORKER_NODE_IP>'
  credentialsRef:
    kind: SSHCredentials
    name: ssh-credentials
EOF
```

Здесь:

- <SSH_PRIVATE_KEY> — приватная часть ключа пользователя `caps`, сгенерированная на хосте установки, в формате Base64. Для получения ключа в нужном формате выполните следующую команду:

```
base64 -w0 "$HOME/.ssh/id_rsa"
```

- <MASTER_2_NODE_IP> — IP-адрес узла master-2.
- <MASTER_3_NODE_IP> — IP-адрес узла master-3.
- <SYSTEM_1_NODE_IP> — IP-адрес узла system-1.
- <SYSTEM_2_NODE_IP> — IP-адрес узла system-2.

- <FRONTEND_1_NODE_IP> — IP-адрес узла frontend-1.
- <FRONTEND_2_NODE_IP> — IP-адрес узла frontend-2.
- <WORKER_NODE_IP> — IP-адрес узла worker.

4.3.4 Дополнительные общие настройки кластера

Создайте правило появления предупреждений о наличии необычного поведения в кластере на основе журнала аудита. Данная настройка включает срабатывание на все типы событий, в том числе на предупреждения (warning) и уведомления (notice).

```
d8 k apply -f - << EOF
apiVersion: deckhouse.io/v1
kind: CustomPrometheusRules
metadata:
  name: falco-critical-alerts
spec:
  groups:
  - name: falco-critical-alerts
    rules:
    - alert: FalcoCriticalAlertsAreFiring
      for: 1m
      annotations:
        description: |
          There is a suspicious activity on a node {{ $labels.node }}.
          Check you events journal for more details.
        summary: Falco detects a critical security incident
      expr: |
        sum by (node)
(rate(falco_events{priority=~"error|critical|warning|notice"}[5m]) > 0)
EOF
```

Добавьте локального пользователя и предоставьте ему права администратора системы:

```
d8 k apply -f - << EOF
apiVersion: deckhouse.io/v1
kind: User
metadata:
  name: admin
spec:
  email: admin@deckhouse.ru
  # Сгенерируйте пароль для пользователя и подставьте его в команду echo ниже, чтобы
получить hash и base64 от него, для указания в секции password:
  # echo '<ПАРОЛЬ>' | htpasswd -BinC 10 "" | cut -d: -f2 | base64 -w0
  password: 'base64-от-хеша-вашего-пароля'
---
apiVersion: deckhouse.io/v1alpha1
kind: Group
metadata:
  name: admins
spec:
  name: admins
  members:
    - kind: User
      name: admin
---
```

```
apiVersion: deckhouse.io/v1
kind: ClusterAuthorizationRule
metadata:
  name: admin
spec:
  subjects:
    - kind: Group
      name: admins
  accessLevel: SuperAdmin
  portForwarding: true
  namespaceSelector:
    matchAny: true
EOF
```

Настройте IngressNginxController для входящего трафика. Если у вас кластер только из одного узла, то используйте следующую конфигурацию:

```
d8 k apply -f - << EOF
apiVersion: deckhouse.io/v1
kind: IngressNginxController
metadata:
  name: main
spec:
  ingressClass: nginx
  inlet: HostWithFailover
  nodeSelector:
    node-role.kubernetes.io/master: ""
  tolerations:
  - effect: NoSchedule
    operator: Exists
EOF
```

Если у вас отказоустойчивый кластер с выделенными frontend-узлами для приема входящего трафика, то используйте следующую конфигурацию:

```
d8 k apply -f - << EOF
apiVersion: deckhouse.io/v1
kind: IngressNginxController
metadata:
  name: main
spec:
  ingressClass: nginx
  inlet: HostWithFailover
  nodeSelector:
    node-role.deckhouse.io/frontend: ""
  tolerations:
    - effect: NoExecute
      key: dedicated.deckhouse.io
      value: frontend
EOF
```

4.3.5 Настройка балансировки входящего трафика с помощью виртуальных IP-адресов

Для отказоустойчивой инсталляции, когда необходимо использовать виртуальный IP-адрес или несколько IP-адресов для всего кластера ПО «Deckhouse Platform», чтобы балансировать

входящий трафик между узлами кластера, где располагается IngressNginxController, необходимо включить и настроить модуль metallb.

При наличии двух frontend-узлов и равномерного распределения входящего трафика между ними, необходимо:

- выделить на кластер ПО «Deckhouse Platform» два виртуальных IP-адреса из подсети узлов кластера
- указать выделенные IP-адреса как A-записи на DNS-сервере у соответствующего доменного имени кластера (желательно использовать wildcard-запись, например *.kube.company.my)
- указать выделенные IP-адреса в настройках модуля metallb.

DNS-сервер будет разрешать запросы к доменному имени в IP-адреса поочередно при каждом DNS-запросе. Модуль metallb распределит два выделенных IP-адреса по двум frontend-узлам, по одному на каждый узел. В этом случае трафик равномерно распределится между узлами. В случае отказа одного из frontend-узлов, его IP-адрес «переедет» на другой работающий frontend-узел.

Команды для настройки балансировки при отказоустойчивой установке:

```
# Включите модуль metallb:
d8 k apply -f - << EOF
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: metallb
spec:
  enabled: true
  version: 2
EOF

# Создайте ресурс MetalLoadBalancerClass:
d8 k apply -f - << EOF
apiVersion: network.deckhouse.io/v1alpha1
kind: MetalLoadBalancerClass
metadata:
  name: ingress
spec:
  addressPool:
    - 192.168.2.100-192.168.2.101
  isDefault: false
  nodeSelector:
    node-role.deckhouse.io/frontend: ""
  type: L2
EOF

# Пересоздайте ресурс IngressNginxController:
d8 k delete IngressNginxController main
d8 k apply -f - << EOF
apiVersion: deckhouse.io/v1
```

```
kind: IngressNginxController
metadata:
  name: main
spec:
  ingressClass: nginx
  inlet: LoadBalancer
  loadBalancer:
    loadBalancerClass: ingress
    annotations:
      # Количество адресов, которые будут выделены из пула, описанного в
_MetalLoadBalancerClass_.
      network.deckhouse.io/l2-load-balancer-external-ips-count: "2"
EOF

#Платформа создаст сервис с типом LoadBalancer, которому будет присвоено заданное
количество адресов, например:
d8 k -n d8-ingress-nginx get svc
NAME                    TYPE           CLUSTER-IP      EXTERNAL-IP
PORT(S)                      AGE
main-load-balancer      LoadBalancer   10.222.130.11   192.168.2.100,192.168.2.101
80:30689/TCP,443:30668/TCP   11s
```

Проверьте состояние очереди, выполнив команду:

```
d8 system queue list
```

В очереди не должно быть задач на выполнение. Пример вывода:

```
$ d8 system queue list
Summary:
- 'main' queue: empty.
- 91 other queues (0 active, 91 empty): 0 tasks.
- no tasks to handle
```

Дождитесь запуска подов ПО «Deckhouse Platform». Например, можно использовать следующую команду (вывод должен быть пуст):

```
d8 k get po -A | grep -vE 'Running|Completed'
```

Убедитесь, что у вас есть доступ к веб-интерфейсам ПО «Deckhouse Platform» через браузер. Например, можно проверить доступность следующих адресов:

- https://grafana.<PUBLIC_DOMAIN_NAME>
- https://kubeconfig.<PUBLIC_DOMAIN_NAME>

Проверьте, что на главной странице системы мониторинга (Grafana) корректно отображаются все данные.

4.4 Настройка хранилищ

Настройка хранилищ происходит в несколько шагов, которые зависят от выбранного типа хранилища.

Основные этапы настройки:

- Включение и конфигурирование соответствующих модулей.
- Создание групп томов (Volume Groups).
- Подготовка и создание объектов StorageClass, их последующее назначение и использование.

4.4.1 Локальное хранилище Local Path Provisioner

ПО «Deckhouse Platform» предоставляет возможность настраивать локальные хранилища Local Path Provisioner. Это простое решение без поддержки снимков и ограничений на размер, которое лучше всего подходит для разработки, тестирования и небольших кластеров. Данное хранилище использует локальное дисковое пространство для создания PersistentVolume, не полагаясь на внешние системы хранения данных.

Для каждого ресурса LocalPathProvisioner создается соответствующий объект StorageClass. Список узлов, на которых разрешено использовать StorageClass, определяется на основе поля nodeGroups и используется при размещении подов.

При запросе диска подом происходит следующее:

- создаётся PersistentVolume с типом HostPath;
- на нужном узле создается директория, путь к которой формируется из параметра path, имени PV и PVC.

Пример пути:

```
/opt/local-path-provisioner/pvc-d9bd3878-f710-417b-a4b3-38811aa8aac1_d8-monitoring_p
rometheus-main-db-prometheus-main-0
```

Пример ресурса LocalPathProvisioner (reclaimPolicy устанавливается по умолчанию в Retain):

```
apiVersion: deckhouse.io/v1alpha1
kind: LocalPathProvisioner
```

```
metadata:
  name: localpath-system
spec:
  nodeGroups:
  - system
  path: "/opt/local-path-provisioner"
```

4.4.2 Локальное хранилище на основе Logical Volume Manager

Локальное хранилище снижает сетевые задержки и обеспечивает более высокую производительность по сравнению с удалёнными хранилищами, доступ к которым осуществляется по сети.

Для настройки локального хранилища на основе LVM выполните следующие шаги:

- Настройте LVMVolumeGroup. Перед созданием StorageClass необходимо создать ресурс LVMVolumeGroup модуля sds-node-configurator на узлах кластера.
- Включите модуль sds-node-configurator. Убедитесь, что модуль включен до включения модуля sds-local-volume.
- Создайте соответствующие объекты StorageClass. Создание StorageClass для CSI-драйвера local.csi.storage.deckhouse.io пользователем запрещено.

Включение модуля sds-node-configurator:

1. Создайте ресурс ModuleConfig для включения модуля:

```
d8 k apply -f - <<EOF
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: sds-node-configurator
spec:
  enabled: true
  version: 1
EOF
```

2. Дождитесь состояния модуля Ready. На этом этапе не требуется проверять поды в пространстве имен d8-sds-node-configurator. Состояние модуля проверьте командой:

```
d8 k get modules sds-node-configurator -w
```

Включение модуля snapshot-controller:

1. Создайте ресурс ModuleConfig для включения модуля:

```
d8 k apply -f - <<EOF
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: snapshot-controller
spec:
  enabled: true
EOF
```

2. Дождитесь состояния модуля Ready. Состояние модуля проверьте командой:

```
d8 k get modules snapshot-controller -w
```

Включение модуля sds-local-volume:

1. Активируйте модуль sds-local-volume. Пример ниже запускает модуль с настройками по умолчанию, что приведет к созданию служебных подов компонента sds-local-volume на всех узлах кластера:

```
d8 k apply -f - <<EOF
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: sds-local-volume
spec:
  enabled: true
  version: 1
EOF
```

2. Дождитесь состояния модуля Ready:

```
d8 k get modules sds-local-volume -w
```

3. Убедитесь, что в пространствах имен d8-sds-local-volume и d8-sds-node-configurator все поды находятся в статусе Running или Completed и запущены на всех узлах, где планируется использовать ресурсы LVM.

```
d8 k -n d8-sds-local-volume get pod -owide -w
d8 k -n d8-sds-node-configurator get pod -o wide -w
```

Для корректной работы хранилищ на узлах необходимо, чтобы поды sds-local-volume-csi-node были запущены на выбранных узлах. По умолчанию эти поды запускаются на всех узлах кластера. Проверить их наличие можно с помощью команды:

```
d8 k -n d8-sds-local-volume get pod -owide
```

Размещение подов sds-local-volume-csi-node управляется специальными метками (nodeSelector). Эти метки задаются в параметре spec.settings.dataNodes.nodeSelector модуля.

Для настройки хранилища на узлах необходимо создать группы томов LVM с использованием ресурсов LVMVolumeGroup. В данном примере создается хранилище Thick.

Перед созданием ресурса LVMVolumeGroup убедитесь, что на данном узле запущен под sds-local-volume-csi-node. Это можно сделать командой:

```
d8 k -n d8-sds-local-volume get pod -owide
```

1. Получите все ресурсы BlockDevice, которые доступны в вашем кластере:

```
d8 k get bd
```

Пример вывода:

```
NAME                                             NODE       CONSUMABLE   SIZE
PATH
dev-ef4fb06b63d2c05fb6ee83008b55e486aa1161aa   worker-0   false        976762584Ki
/dev/nvme1n1
dev-0cfc0d07f353598e329d34f3821bed992c1ffbcd   worker-0   false        894006140416
/dev/nvme0n1p6
dev-7e4df1ddf2a1b05a79f9481cdf56d29891a9f9d0   worker-1   false        976762584Ki
/dev/nvme1n1
dev-b103062f879a2349a9c5f054e0366594568de68d   worker-1   false        894006140416
/dev/nvme0n1p6
dev-53d904f18b912187ac82de29af06a34d9ae23199   worker-2   false        976762584Ki
/dev/nvme1n1
dev-6c5abbd549100834c6b1668c8f89fb97872ee2b1   worker-2   false        894006140416
/dev/nvme0n1p6
```

2. Создайте ресурс LVMVolumeGroup для узла worker-0:

```
d8 k apply -f - <<EOF
apiVersion: storage.deckhouse.io/v1alpha1
kind: LVMVolumeGroup
metadata:
  name: "vg-1-on-worker-0" # Подходит любое допустимое имя ресурса в Kubernetes. Это
имя ресурса LVMVolumeGroup будет использоваться для создания LocalStorageClass в
будущем.
spec:
  type: Local
  local:
    nodeName: "worker-0"
  blockDeviceSelector:
    matchExpressions:
      - key: kubernetes.io/metadata.name
        operator: In
        values:
          - dev-ef4fb06b63d2c05fb6ee83008b55e486aa1161aa
          - dev-0cfc0d07f353598e329d34f3821bed992c1ffbcd
```

```
  actualVGNameOnTheNode: "vg-1" # Имя LVM VG, который будет создан из указанных
блочных устройств на узле.
EOF
```

3. Дождитесь, когда созданный ресурс LVMVolumeGroup перейдет в состояние Ready:

```
d8 k get lvg vg-1-on-worker-0 -w
```

Если ресурс перешел в состояние Ready, это значит, что на узле worker-0 из блочных устройств /dev/nvme1n1 и /dev/nvme0n1p6 была создана LVM VG с именем vg-1.

4. Создайте ресурс LVMVolumeGroup для узла worker-1:

```
d8 k apply -f - <<EOF
apiVersion: storage.deckhouse.io/v1alpha1
kind: LVMVolumeGroup
metadata:
  name: "vg-1-on-worker-1"
spec:
  type: Local
  local:
    nodeName: "worker-1"
  blockDeviceSelector:
    matchExpressions:
      - key: kubernetes.io/metadata.name
        operator: In
        values:
          - dev-7e4df1ddf2a1b05a79f9481cdf56d29891a9f9d0
          - dev-b103062f879a2349a9c5f054e0366594568de68d
  actualVGNameOnTheNode: "vg-1"
EOF
```

5. Дождитесь, когда созданный ресурс LVMVolumeGroup перейдет в состояние Ready:

```
d8 k get lvg vg-1-on-worker-1 -w
```

Если ресурс перешел в состояние Ready, это значит, что на узле worker-1 из блочного устройства /dev/nvme1n1 и /dev/nvme0n1p6 была создана LVM VG с именем vg-1.

6. Создайте ресурс LVMVolumeGroup для узла worker-2:

```
d8 k apply -f - <<EOF
apiVersion: storage.deckhouse.io/v1alpha1
kind: LVMVolumeGroup
metadata:
  name: "vg-1-on-worker-2"
spec:
  type: Local
  local:
    nodeName: "worker-2"
  blockDeviceSelector:
    matchExpressions:
```

```
      - key: kubernetes.io/metadata.name
        operator: In
        values:
          - dev-53d904f18b912187ac82de29af06a34d9ae23199
          - dev-6c5abbd549100834c6b1668c8f89fb97872ee2b1
  actualVGNameOnTheNode: "vg-1"
EOF
```

7. Дождитесь, когда созданный ресурс LVMVolumeGroup перейдет в состояние Ready:

```
d8 k get lvg vg-1-on-worker-2 -w
```

Если ресурс перешел в состояние Ready, то это значит, что на узле worker-2 из блочного устройства /dev/nvme1n1 и /dev/nvme0n1p6 была создана LVM VG с именем vg-1.

8. Создайте ресурс LocalStorageClass:

```
d8 k apply -f -<<EOF
apiVersion: storage.deckhouse.io/v1alpha1
kind: LocalStorageClass
metadata:
  name: local-storage-class
spec:
  lvm:
    lvmVolumeGroups:
      - name: vg-1-on-worker-0
      - name: vg-1-on-worker-1
      - name: vg-1-on-worker-2
    type: Thick
  reclaimPolicy: Delete
  volumeBindingMode: WaitForFirstConsumer
EOF
```

9. Дождитесь, когда созданный ресурс LocalStorageClass перейдет в состояние Created:

```
d8 k get lsc local-storage-class -w
```

10. Проверьте, что был создан соответствующий StorageClass:

```
d8 k get sc local-storage-class
```

Если StorageClass с именем local-storage-class появился, значит настройка модуля sds-local-volume завершена. Теперь пользователи могут создавать PVC, указывая StorageClass с именем local-storage-class.

4.4.3 Распределённая система хранения Ceph

Ceph — это масштабируемая распределённая система хранения, обеспечивающая высокую доступность и отказоустойчивость данных. В ПО «Deckhouse Platform» поддерживается

интеграция с Ceph-кластерами. Это даёт возможность динамически управлять хранилищем и использовать StorageClass на основе RADOS Block Device (RBD) или CephFS.

Внимание. Интеграция с Ceph-кластерами возможна только при использовании containerd v1. Работа совместно с containerd v2 не поддерживается.

Включение модуля snapshot-controller.

Создайте ресурс ModuleConfig для включения модуля:

```
d8 k apply -f - <<EOF
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: snapshot-controller
spec:
  enabled: true
EOF
```

Дождитесь состояния модуля Ready. Состояние модуля проверьте командой:

```
d8 k get modules snapshot-controller -w
```

Для подключения Ceph-кластера в ПО «Deckhouse Platform» необходимо включить модуль csi-ceph. Для этого примените ресурс ModuleConfig:

```
d8 k apply -f - <<EOF
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: csi-ceph
spec:
  enabled: true
EOF
```

Чтобы настроить подключение к Ceph-кластеру, примените ресурс CephClusterConnection. Пример команды:

```
d8 k apply -f - <<EOF
apiVersion: storage.deckhouse.io/v1alpha1
kind: CephClusterConnection
metadata:
  name: ceph-cluster-1
spec:
  # FSID/UUID Ceph-кластера.
  # Получить FSID/UUID Ceph-кластера можно с помощью команды `ceph fsid`.
  clusterID: 2bf085fc-5119-404f-bb19-820ca6a1b07e
  # Список IP-адресов ceph-mon'ов в формате `10.0.0.10:6789`.
  monitors:
    - 10.0.0.10:6789
  # Имя пользователя без `client.`.
```

```
  # Получить имя пользователя можно с помощью команды `ceph auth list`.
  userID: admin
  # Ключ авторизации, соответствующий userID.
  # Получить ключ авторизации можно с помощью команды `ceph auth get-key
client.admin`.
  userKey: AQDiVXVmBJVRLxAAg65PhODrtwbwSWrjJwssUg==
EOF
```

Проверьте создание подключения следующей командой (Phase должен быть Created):

```
d8 k get cephclusterconnection ceph-cluster-1
```

Создание объектов StorageClass осуществляется через ресурс CephStorageClass, который определяет конфигурацию для желаемого класса хранения. Ручное создание ресурса StorageClass без CephStorageClass может привести к ошибкам. Пример создания StorageClass на основе RBD:

```
d8 k apply -f - <<EOF
apiVersion: storage.deckhouse.io/v1alpha1
kind: CephStorageClass
metadata:
  name: ceph-rbd-sc
spec:
  clusterConnectionName: ceph-cluster-1
  reclaimPolicy: Delete
  type: RBD
  rbd:
    defaultFSType: ext4
    pool: ceph-rbd-pool
EOF
```

Пример создания StorageClass на основе файловой системы Ceph:

```
d8 k apply -f - <<EOF
apiVersion: storage.deckhouse.io/v1alpha1
kind: CephStorageClass
metadata:
  name: ceph-fs-sc
spec:
  clusterConnectionName: ceph-cluster-1
  reclaimPolicy: Delete
  type: CephFS
  cephFS:
    fsName: cephfs
EOF
```

Проверьте, что созданные ресурсы CephStorageClass перешли в состояние Created, выполнив следующую команду:

```
d8 k get cephstorageclass
```

В результате будет выведена информация о созданных ресурсах CephStorageClass:

```
NAME          PHASE     AGE
ceph-rbd-sc   Created   1h
ceph-fs-sc    Created   1h
```

Проверьте созданный StorageClass с помощью следующей команды:

```
d8 k get sc
```

В результате будет выведена информация о созданном `StorageClass`:

```
NAME          PROVISIONER         RECLAIMPOLICY   VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION   AGE
ceph-rbd-sc   rbd.csi.ceph.com    Delete          WaitForFirstConsumer   true
15s
ceph-fs-sc    rbd.csi.ceph.com    Delete          WaitForFirstConsumer   true
15s
```

Если объекты StorageClass появились, значит настройка модуля csi-ceph завершена. Теперь пользователи могут создавать PersistentVolume, указывая созданные объекты StorageClass.

4.4.4 Сетевое файловое хранилище NFS

ПО «Deckhouse Platform» поддерживает интеграцию с Network File System (NFS), обеспечивая возможность использования сетевых файловых хранилищ в качестве томов. Модуль csi-nfs предоставляет CSI-драйвер для подключения NFS-серверов и создания PersistentVolume на их основе.

Включение модуля snapshot-controller.

Создайте ресурс ModuleConfig для включения модуля:

```
d8 k apply -f - <<EOF
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: snapshot-controller
spec:
  enabled: true
EOF
```

Дождитесь состояния модуля Ready. Состояние модуля проверьте командой:

```
d8 k get modules snapshot-controller -w
```

Для поддержки работы с NFS-хранилищем включите модуль csi-nfs, который позволяет создавать StorageClass с помощью пользовательских ресурсов NFSStorageClass:

```
d8 k apply -f - <<EOF
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: csi-nfs
spec:
  enabled: true
  version: 1
EOF
```

Дождитесь, пока модуль перейдет в состояние Ready:

```
d8 k get module csi-nfs -w
```

Проверьте состояние подов в пространстве имён d8-csi-nfs. Все поды должны быть в состоянии Running или Completed, и запущены на всех узлах:

```
d8 k -n d8-csi-nfs get pod -owide -w
```

Для создания StorageClass необходимо использовать ресурс NFSStorageClass. Пример создания ресурса:

```
d8 k apply -f - <<EOF
apiVersion: storage.deckhouse.io/v1alpha1
kind: NFSStorageClass
metadata:
  name: nfs-storage-class
spec:
  connection:
    host: 10.223.187.3
    share: /
    nfsVersion: "4.1"
  reclaimPolicy: Delete
  volumeBindingMode: WaitForFirstConsumer
EOF
```

Для каждого PV будет создаваться каталог <директория из share>/<имя PV>.

4.4.5 Хранилище данных на основе протокола SCSI

ПО «Deckhouse Platform» поддерживает управление хранилищами, подключенными через iSCSI или Fibre Channel, обеспечивая возможность работы с томами на уровне блоковых устройств. Это позволяет интегрировать системы хранения данных и управлять ими через CSI-драйвер.

Перед включением этого модуля в конфигурации ПО «Deckhouse Platform» необходимо явно разрешить параметр allowExperimentalModules:

```
d8 k patch moduleconfig deckhouse --type='json' -p='[{"op": "add", "path":
"/spec/settings/allowExperimentalModules", "value": true}]'
```

Для настройки таких хранилищ включите модуль csi-scsi-generic:

```
d8 k apply -f - <<EOF
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: csi-scsi-generic
spec:
  enabled: true
  version: 1
EOF
```

Дождитесь, когда модуль перейдет в состояние Ready:

```
d8 k get module csi-scsi-generic -w
```

Для создания SCSITarget необходимо использовать ресурс SCSITarget. Пример команд для создания такого ресурса:

```
d8 k apply -f -<<EOF
apiVersion: storage.deckhouse.io/v1alpha1
kind: SCSITarget
metadata:
  name: hpe-3par-1
spec:
  deviceTemplate:
    metadata:
      labels:
        my-key: some-label-value
  iSCSI:
    auth:
      login: ""
      password: ""
    iqn: iqn.2000-05.com.3pardata:xxxx1
    portals:
    - 192.168.1.1
---
apiVersion: storage.deckhouse.io/v1alpha1
kind: SCSITarget
metadata:
  name: hpe-3par-2
spec:
  deviceTemplate:
    metadata:
      labels:
        my-key: some-label-value
  iSCSI:
    auth:
      login: ""
      password: ""
    iqn: iqn.2000-05.com.3pardata:xxxx2
    portals:
```

```
    - 192.168.1.2
EOF
```

Обратите внимание, что в примере выше используются два SCSITarget. Таким образом можно создать несколько SCSITarget как для одного, так и для разных СХД. Это позволяет использовать multipath для повышения отказоустойчивости и производительности.

Проверить создание объекта можно командой (Phase должен быть Created):

```
d8 k get scsitargets.storage.deckhouse.io <имя scsitarget>
```

Для создания StorageClass необходимо использовать ресурс SCSIStorageClass. Пример команды для создания такого ресурса:

```
d8 k apply -f -<<EOF
apiVersion: storage.deckhouse.io/v1alpha1
kind: SCSIStorageClass
metadata:
  name: scsi-all
spec:
  scsiDeviceSelector:
    matchLabels:
      my-key: some-label-value
  reclaimPolicy: Delete
EOF
```

Обратите внимание на scsiDeviceSelector. Этот параметр позволяет выбрать SCSITarget для создания PV по лейблам. В примере выше выбираются все SCSITarget с лейблом my-key: some-label-value. Этот лейбл будет назначен на все девайсы, которые будут обнаружены в указанных SCSITarget.

Проверить создание объекта можно командой (Phase должен быть Created):

```
d8 k get scsistorageclasses.storage.deckhouse.io <имя scsistorageclass>
```

4.4.6 Хранилище данных Yadro Tatlin Unified Storage

ПО «Deckhouse Platform» поддерживает интеграцию с системой хранения данных TATLIN.UNIFIED (Yadro), предоставляя возможность управления томами. Это позволяет использовать централизованное хранилище для контейнеризированных рабочих нагрузок, обеспечивая высокую производительность и отказоустойчивость.

Включение модуля snapshot-controller.

Создайте ресурс ModuleConfig для включения модуля:

```
d8 k apply -f - <<EOF
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: snapshot-controller
spec:
  enabled: true
EOF
```

Дождитесь состояния модуля Ready. Состояние модуля проверьте командой:

```
d8 k get modules snapshot-controller -w
```

Для управления томами на основе системы хранения данных TATLIN.UNIFIED (Yadro) используется модуль csi-yadro-tatlin-unified, позволяющий создавать ресурсы StorageClass через создание пользовательских ресурсов YadroTatlinUnifiedStorageClass. Чтобы включить модуль, выполните команду:

```
d8 k apply -f - <<EOF
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: csi-yadro-tatlin-unified
spec:
  enabled: true
  version: 1
EOF
```

Дождитесь, когда модуль csi-yadro-tatlin-unified перейдет в состояние Ready. Проверить состояние модуля можно, выполнив следующую команду:

```
d8 k get module csi-yadro-tatlin-unified -w
```

В результате будет выведена информация о модуле:

```
NAME                        STAGE    SOURCE     PHASE        ENABLED    READY
csi-yadro-tatlin-unified             Embedded   Available    True       True
```

Чтобы создать подключение к системе хранения данных TATLIN.UNIFIED и иметь возможность настраивать объекты StorageClass, примените следующий ресурс YadroTatlinUnifiedStorageConnection:

```
d8 k apply -f - <<EOF
apiVersion: storage.deckhouse.io/v1alpha1
kind: YadroTatlinUnifiedStorageConnection
metadata:
  name: yad1
spec:
```

```
  controlPlane:
    address: "172.19.28.184"
    username: "admin"
    password: "cGFzc3dvcmQ=" # Должен быть закодирован в Base64.
    ca: "base64encoded"
    skipCertificateValidation: true
  dataPlane:
    protocol: "iscsi"
    iscsi:
      volumeExportPort: "p50,p51,p60,p61"
EOF
```

Для создания StorageClass необходимо использовать ресурс YadroTatlinUnifiedStorageClass. Ручное создание ресурса StorageClass без YadroTatlinUnifiedStorageClass может привести к ошибкам.

Пример команды для создания класса хранения на основе системы хранения данных TATLIN.UNIFIED:

```
d8 k apply -f - <<EOF
apiVersion: storage.deckhouse.io/v1alpha1
kind: YadroTatlinUnifiedStorageClass
metadata:
  name: yad1
spec:
  fsType: "xfs"
  pool: "pool-hdd"
  storageConnectionName: "yad1"
  reclaimPolicy: Delete
EOF
```

4.5 Откат установки и удаление ПО «Deckhouse Platform»

В случае прерывания установки или возникновения ошибок во время установки, могут остаться созданные ресурсы. Операции отката и удаления выполняются из контейнера инсталлятора ПО «Deckhouse Platform» на отдельной машине вне кластера, с использованием того же файла конфигурации, который применялся при установке.

Важно: Версия контейнера инсталлятора должна совпадать с версией Deckhouse Platform, установка или удаление которой выполняется.

Для прерывания установки и удаления созданных на текущем этапе ресурсов используйте команду:

```
dhctl bootstrap-phase abort
```

Обратите внимание, что файл конфигурации (передаваемый через параметр --config) должен быть тот же, с которым производилась установка.

Чтобы удалить кластер развернутого ПО «Deckhouse Platform», используйте команду:

```
dhctl destroy
```

В этом случае dhctl подключится к master-узлу, получит от него terraform state и корректно удалит созданные ресурсы.

4.6 Создание образов

Создание образов происходит с помощью утилиты docker из среды функционирования ПО «Deckhouse Platform».

Описание создания образов приведено ниже:

- Astra Linux Special Edition
  https://wiki.astralinux.ru/pages/viewpage.action?pageId=158601444
- ОС РЕД ОС
  https://redos.red-soft.ru/base/arm/arm-other/docker-install/
- Альт 8 СП
  https://www.altlinux.org/Docker

При включенном и настроенном operator-trivy (п. 5.1), развёртывание контейнеров из уязвимых образов будет запрещено.

4.7 Настройка ПО «Deckhouse Platform»

ПО «Deckhouse Platform» состоит из оператора Deckhouse и модулей. Модуль – это набор из Helm-чарта, хуков, правил сборки компонентов модуля (компонентов) и других файлов.

ПО «Deckhouse Platform» настраивается с помощью:

- *Глобальных настроек.* Глобальные настройки хранятся в кастомном ресурсе ModuleConfig/global. Глобальные настройки можно рассматривать как специальный модуль global, который нельзя отключать.
- *Настроек модулей.* Настройки каждого модуля хранятся в кастомном ресурсе ModuleConfig, имя которого совпадает с именем модуля (в kebab-case).
- *Кастомных ресурсов.* Некоторые модули настраиваются с помощью дополнительных кастомных рсурсов.

Пример набора кастомных рсурсов конфигурации Deckhouse:

```
# Глобальные настройки.
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: global
spec:
  version: 1
  settings:
    modules:
      publicDomainTemplate: "%s.kube.company.my"
---
# Настройки модуля monitoring-ping.
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: monitoring-ping
spec:
  version: 1
  settings:
    externalTargets:
    - host: 8.8.8.8
---
# Отключить модуль dashboard.
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: dashboard
spec:
  enabled: false
```

Посмотреть список кастомных ресурсов ModuleConfig, состояние модуля (включен/выключен) и его статус можно с помощью команды:

```
d8 k get moduleconfigs
```

Список и состояние модулей можно также получить с помощью команды:

```
d8 k get modules
```

Чтобы изменить глобальную конфигурацию Deckhouse или конфигурацию модуля, нужно создать или отредактировать соответствующий ресурс ModuleConfig.

Например, чтобы отредактировать конфигурацию модуля upmeter, выполните следующую команду:

```
d8 k edit moduleconfig/upmeter
```

После завершения редактирования изменения применяются автоматически.

4.8 Настройка модуля

Модуль настраивается с помощью кастомного ресурса ModuleConfig, имя которого совпадает с именем модуля (в kebab-case). Кастомный ресурс ModuleConfig имеет следующие поля:

- metadata.name – название модуля ПО «Deckhouse Platform» в kebab-case (например prometheus, node-manager).
- spec.version – версия схемы настроек модуля (целое число, больше нуля). Обязательное поле, если spec.settings не пустое. Номер актуальной версии можно увидеть в документации модуля в разделе «Настройки».
    - ПО «Deckhouse Platform» поддерживает обратную совместимость версий схемы настроек модуля. Если используется схема настроек устаревшей версии, при редактировании или просмотре кастомного ресурса будет выведено предупреждение о необходимости обновить схему настроек модуля.
- spec.settings – настройки модуля. Необязательное поле, если используется поле spec.enabled.
- spec.enabled – необязательное поле для явного включения или отключения модуля. Если не задано, модуль может быть включен по умолчанию в одном из наборов модулей.

ПО «Deckhouse Platform» не изменяет кастомные ресурсы ModuleConfig. Это позволяет применять подход Infrastructure as Code (IaC) при хранении конфигурации. Другими словами, можно воспользоваться всеми преимуществами системы контроля версий для хранения настроек ПО «Deckhouse Platform», использовать d8, Helm, kubectl и другие привычные инструменты. В данном руководстве большая часть команд подразумевает использование консольной утилиты d8, которая идет в составе поставки ПО «Deckhouse Platform». Утилита d8 позволяет выполнять все необходимые для управления кластером операции.

Пример кастомного ресурса для настройки модуля kube-dns:

```
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: kube-dns
spec:
  version: 1
  settings:
    stubZones:
    - upstreamNameservers:
      - 192.168.121.55
      - 10.2.7.80
      zone: directory.company.my
    upstreamNameservers:
    - 10.2.100.55
    - 10.2.200.55
```

Некоторые модули настраиваются с помощью дополнительных кастомных ресурсов. Справку по ресурсам можно найти в настоящем руководстве или в веб-версии документации в кластере (требуется включенный модуль documentation).

4.9 Включение и отключение модуля

Некоторые модули могут быть включены по умолчанию в зависимости от используемого набора модулей.

Для явного включения или отключения модуля необходимо установить true или false в поле .spec.enabled в соответствующем кастомном ресурсе ModuleConfig. Если для модуля нет такого кастомного ресурса ModuleConfig, его нужно создать.

Пример явного выключения модуля user-authn (модуль будет выключен независимо от используемого набора модулей):

```
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: user-authn
spec:
  enabled: false
```

Проверить состояние модуля можно с помощью команды:

```
d8 k get moduleconfig <ИМЯ_МОДУЛЯ>
```

Пример:

```
$ d8 k get moduleconfig user-authn
NAME         ENABLED   VERSION   AGE   MESSAGE
user-authn   false     1         12h
```

4.10 Механизмы аутентификации и авторизации

4.10.1 Локальная аутентификация

Локальная аутентификация обеспечивает проверку и управление доступом пользователей с возможностью настройки парольной политики, поддержкой двухфакторной аутентификации и управлением группами. Реализация соответствует требованиям безопасности ФСТЭК России и рекомендациям OWASP, она обеспечивает защиту доступа к кластеру и приложениям без необходимости интеграции с внешними системами аутентификации.

Локальная аутентификация подразумевает создание в кластере объектов User и Group для статических пользователей и групп:

- В объекте User хранится информация о пользователе, включая email и хеш пароля (пароль в явном виде не сохраняется).
- В объекте Group задаётся список пользователей, объединённых в группу.

4.10.1.1 Создание статического пользователя

Для создания статического пользователя создайте ресурс User. Пример создания ресурса:

```
apiVersion: deckhouse.io/v1
kind: User
metadata:
  name: admin
spec:
  email: admin@yourcompany.com
  password: $2a$10$etblbZ9yfZaKgbvysf1qguW3WULdMnxwWFrkoKpRH1yeWa5etjjAa
  ttl: 24h
```

Здесь:

- ttl - время жизни учетной записи пользователя. Задаётся в виде строки с указанием часов и минут: 30m, 1h, 2h30m, 24h. Указать ttl можно только 1 раз.

Придумайте пароль и укажите его хеш-сумму в поле password. Пароль хранится в зашифрованном виде (bcrypt). Хеш-сумму можно сгенерировать с помощью команды:

```
echo "$password" | htpasswd -BinC 10 "" | cut -d: -f2 | base64 -w0
```

Если команда `htpasswd` недоступна, установите соответствующий пакет:

- apache2-utils — для ОС Astra Linux Special Edition и Московской серверной операционной системы;
- httpd-tools — для ОС РЕД ОС;
- apache2-htpasswd — для ОС Альт.

4.10.1.2 Добавление пользователя в группу

Чтобы объединять статических пользователей в группы, создайте ресурс Group. Пример создания ресурса:

```
apiVersion: deckhouse.io/v1alpha1
kind: Group
metadata:
  name: admins
spec:
  name: admins
  members:
    - kind: User
      name: admin
```

Здесь:

- members — список пользователей, которые входят в группу.

После создания группы и добавления в неё пользователей, необходимо настроить авторизацию.

Запрещено использовать пользователей и группы с префиксом system. Аутентификация таких пользователей или участников этих групп будет отклонена, а в логах kube-apiserver появится соответствующее предупреждение.

4.10.1.3 Настройка парольной политики

Парольная политика позволяет контролировать сложность пароля, ротацию и блокировку пользователей.

Для настройки парольной политики используйте поле passwordPolicy в конфигурации модуля user-authn:

```
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: user-authn
spec:
  version: 2
  enabled: true
  settings:
    passwordPolicy:
      complexityLevel: Fair
      passwordHistoryLimit: 10
      lockout:
        lockDuration: 15m
        maxAttempts: 3
      rotation:
        interval: "30d"
```

Здесь:

- complexityLevel — уровень сложности пароля;
- passwordHistoryLimit — число предыдущих паролей, которые хранит система, чтобы предотвратить их повторное использование;
- lockout — настройки блокировки при превышении лимита неудачных попыток входа:
- lockout.maxAttempts — лимит неудачных попыток;
- lockout.lockDuration — длительность блокировки пользователя;
- rotation — настройки ротации паролей:
- rotation.interval — период обязательной смены пароля.

4.10.1.4 Настройка двухфакторной аутентификации

Двухфакторная аутентификация позволяет повысить уровень безопасности, требуя ввести код из приложения-аутентификатора при входе.

Для настройки используйте поле staticUsers2FA в конфигурации модуля user-authn:

```
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: user-authn
spec:
  version: 2
  enabled: true
  settings:
    staticUsers2FA:
      enabled: true
      issuerName: "awesome-app"
```

Здесь:

- enabled — включает или отключает 2FA для всех статических пользователей;
- issuerName — имя, которое будет отображаться в приложении-аутентификаторе при добавлении аккаунта.

После включения 2FA каждый пользователь должен пройти процесс регистрации в приложении-аутентификаторе при первом входе.

4.10.2 Подключение провайдера аутентификации

В ПО «Deckhouse Platform» реализован ролевой метод управления доступом с помощью внешнего провайдера аутентификации, реализующего взаимодействие по протоколу LDAP или OIDC. Подключение внешнего провайдера аутентификации выполняется при помощи ресурса DexProvider (см. Справочник администратора, приведенный в электронном приложении к настоящему документу, каталог «Электронные приложения» - «Deckhouse Platform. Руководство администратора» - «Справочник администратора.pdf»).

Перед подключением провайдера аутентификации его необходимо настроить с учетом ролевой модели доступа, используемой в ПО «Deckhouse Platform». В каталоге, доступ к которому обеспечивает провайдер аутентификации, пользователям, которые должны иметь доступ в ПО «Deckhouse Platform», должны быть присвоены группы, в соответствии с необходимым уровнем прав в рамках ролевой модели доступа ПО «Deckhouse Platform» (Технические условия, Приложение 1. Список ролей).

Пример ресурса DexProvider для подключения провайдера аутентификации LDAP:

```
kind: DexProvider
metadata:
  name: ldap
spec:
  displayName: LDAP
  type: LDAP
  ldap:
    bindDN: cn=admin,dc=novalocal
    bindPW: passw0rd
    groupSearch:
      baseDN: ou=groups,dc=novalocal
      filter: (objectClass=groupOfNames)
      nameAttr: cn
      userMatchers:
      - groupAttr: member
        userAttr: DN
    host: 192.168.10.10:389
    insecureNoSSL: true
    insecureSkipVerify: true
    startTLS: false
    userSearch:
      baseDN: ou=users,dc=novalocal
      emailAttr: mail
      filter: (objectClass=person)
      idAttr: DN
      nameAttr: cn
      username: cn
    usernamePrompt: Email Address
```

4.10.2.1 Ресурс DexProvider

- spec.displayName *строка*

  Обязательный параметр.

  Имя провайдера, которое будет отображено на странице выбора провайдера для аутентификации.

  Если настроен всего один провайдер, страница выбора провайдера показываться не будет.

- spec.ldap *объект*

  Параметры провайдера LDAP.

  - spec.ldap.bindDN *строка*

    Путь до сервис-аккаунта приложения в LDAP.

    Пример:
    bindDN: uid=serviceaccount,cn=users,dc=example,dc=com

- spec.ldap.bindPW *строка*

  Пароль для сервис-аккаунта приложения в LDAP.

  Пример:
  bindPW: password

- spec.ldap.groupSearch *объект*

  Настройки фильтра для поиска групп для указанного пользователя.

  - spec.ldap.groupSearch.baseDN *строка*

    Обязательный параметр.
    Откуда будет начат поиск групп

    Пример:
    baseDN: cn=users,dc=example,dc=com

  - spec.ldap.groupSearch.filter *строка*

    Фильтр для директории с группами.

    Пример:
    filter: "(objectClass=person)"

  - spec.ldap.groupSearch.nameAttr *строка*

    Обязательный параметр.
    Имя атрибута, в котором хранится уникальное имя группы.

    Пример:
    nameAttr: name

  - spec.ldap.groupSearch.userMatchers *массив объектов*

    Обязательный параметр.
    Список сопоставлений атрибута имени пользователя с именем группы.

    - spec.ldap.groupSearch.userMatchers.groupAttr *строка*

      Обязательный параметр.

Имя атрибута, в котором хранятся имена пользователей, состоящих в группе.

Пример:
groupAttr: member

- spec.ldap.groupSearch.userMatchers.userAttr *строка*

  Обязательный параметр.
  Имя атрибута, в котором хранится имя пользователя.

  Пример:
  userAttr: uid

- spec.ldap.host *строка*

  Обязательный параметр.
  Адрес и порт (опционально) LDAP-сервера.

  Пример:
  host: ldap.example.com:636

- spec.ldap.insecureNoSSL *булевый*

  Подключаться к каталогу LDAP не по защищенному порту.

  По умолчанию: false

- spec.ldap.insecureSkipVerify *булевый*

  Не производить проверку подлинности провайдера с помощью TLS. Небезопасно, не рекомендуется использовать в production-окружениях.

  По умолчанию: false

- spec.ldap.rootCAData *строка*

  Цепочка CA в формате PEM, используемая для валидации TLS.

  Пример:
  rootCAData: |
  -----BEGIN CERTIFICATE-----
  MIIFaDC...

-----END CERTIFICATE-----

- spec.ldap.startTLS *булевый*

  Использовать STARTTLS для шифрования.

  По умолчанию: false

- spec.ldap.userSearch *объект*

  Обязательный параметр.
  Настройки фильтров пользователей, которые помогают сначала отфильтровать директории, в которых будет производиться поиск пользователей, а затем найти пользователя по полям (его имени, адресу электронной почты или отображаемому имени).

  - spec.ldap.userSearch.baseDN *строка*

    Обязательный параметр.
    Откуда будет начат поиск пользователей.

    Пример:
    baseDN: cn=users,dc=example,dc=com

  - spec.ldap.userSearch.emailAttr *строка*

    Обязательный параметр.
    Имя атрибута, из которого будет получен email пользователя.

    Пример:
    emailAttr: mail

  - spec.ldap.userSearch.filter *строка*

    Позволяет добавить фильтр для директории с пользователями.

    Пример:
    filter: "(objectClass=person)"

  - spec.ldap.userSearch.idAttr *строка*

    Обязательный параметр.
    Имя атрибута, из которого будет получен ID пользователя.

Пример:
idAttr: uid

- spec.ldap.userSearch.nameAttr *строка*

  Атрибут отображаемого имени пользователя.

  Пример:
  nameAttr: name

- spec.ldap.userSearch.username *строка*

  Обязательный параметр.
  Имя атрибута, из которого будет получен username пользователя.

  Пример:
  username: uid

- spec.ldap.usernamePrompt *строка*

  Строка, которая будет отображаться возле поля для имени пользователя в форме ввода логина и пароля.

  По умолчанию: "LDAP username"

  Пример:
  usernamePrompt: SSO Username

- spec.oidc *объект*

  Параметры провайдера OIDC (можно указывать, только если type: OIDC).

  - spec.oidc.basicAuthUnsupported *булевый*

    Использовать POST-запросы для общения с провайдером вместо добавления токена в Basic Authorization header.
    В большинстве случаев Dex сам определяет, какой запрос ему нужно сделать, но иногда включение этого параметра может помочь.

    По умолчанию: false

  - spec.oidc.claimMapping *объект*

Некоторые провайдеры возвращают нестандартные claim’ы (например, mail). Claim mappings помогают Dex преобразовать их в стандартные claim’ы OIDC.
Dex может преобразовать нестандартный claim в стандартный, только если id_token, полученный от OIDC-провайдера, не содержит аналогичный стандартный claim.

- spec.oidc.claimMapping.email *строка*

  Claim, который будет использован для получения email пользователя.

  По умолчанию: "email"

- spec.oidc.claimMapping.groups *строка*

  Claim, который будет использован для получения групп пользователя.

  По умолчанию: "groups"

- spec.oidc.claimMapping.preferred_username *строка*

  Claim, который будет использован для получения предпочтительного имени пользователя.

  По умолчанию: "preferred_username"

o spec.oidc.clientID *строка*

Обязательный параметр.
ID приложения, созданного в OIDC-провайдере.

o spec.oidc.clientSecret *строка*

Обязательный параметр.
Пароль приложения, созданного в OIDC-провайдере.

o spec.oidc.getUserInfo *булевый*

Запрашивать дополнительные данные об успешно подключенном пользователе.

По умолчанию: false

o spec.oidc.insecureSkipEmailVerified *булевый*

Игнорировать информацию о статусе подтверждения email пользователя.
Как именно подтверждается email, решает сам провайдер. В ответе от провайдера приходит лишь информация — подтвержден email или нет.

По умолчанию: false

- spec.oidc.insecureSkipVerify *булевый*

  Не производить проверку подлинности провайдера с помощью TLS. Небезопасно, не рекомендуется использовать в production-окружениях.

  По умолчанию: false

- spec.oidc.issuer *строка*

  Обязательный параметр.
  Адрес OIDC-провайдера.

  Пример:
  issuer: https://accounts.google.com

- spec.oidc.promptType *строка*

  Определяет — должен ли Issuer запрашивать подтверждение и давать подсказки при аутентификации.
  По умолчанию будет запрошено подтверждение при первой аутентификации.
  Допустимые значения могут изменяться в зависимости от Issuer.

  По умолчанию: "consent"

- spec.oidc.rootCAData *строка*

  Цепочка CA в формате PEM, используемая для валидации TLS.

  Пример:
  rootCAData: |
  -----BEGIN CERTIFICATE-----
  MIIFaDC...
  -----END CERTIFICATE-----

- spec.oidc.scopes *массив строк*

Список полей для включения в ответ при запросе токена.

По умолчанию: ["openid","profile","email","groups","offline_access"]

- spec.oidc.userIDKey *строка*

  Claim, который будет использован для получения ID пользователя.

  По умолчанию: "sub"

- spec.oidc.userNameKey *строка*

  Claim, который будет использован для получения имени пользователя.

  По умолчанию: "name"

- spec.type *строка*

  Тип внешнего провайдера.

  Допустимые значения: OIDC, LDAP

4.10.3 Авторизация

Для реализации ролевой модели в кластере должен быть включён модуль user-authz. Модуль создаёт набор кластерных ролей (ClusterRole), подходящий для большинства задач по управлению доступом пользователей и групп.

Особенности ролевой модели:

- Реализует role-based-подсистему сквозной авторизации, расширяя функционал стандартного механизма RBAC.
- Настройка прав доступа происходит с помощью кастомных ресурсов ClusterAuthorizationRule и AuthorizationRule.
- Управление доступом к инструментам масштабирования (параметр allowScale ресурса ClusterAuthorizationRule или AuthorizationRule).
- Управление доступом к форвардингу портов (параметр portForwarding ресурса ClusterAuthorizationRule или AuthorizationRule).
- Управление списком разрешённых пространств имён в формате labelSelector (параметр namespaceSelector ресурса ClusterAuthorizationRule).

Вы можете получить дополнительный список правил доступа для роли модуля из кластера (существующие пользовательские правила и нестандартные правила из других модулей ПО «Deckhouse Platform») с помощью команды:

```
D8_ROLE_NAME=Editor
d8 k get clusterrole -A -o jsonpath="{range
.items[?(@.metadata.annotations.user-authz\.deckhouse\.io/access-level=='$D8_ROLE_NA
ME')]}{.rules}{'\n'}{end}" | jq -s add
```

Пример использования AuthorizationRule для установки правил доступа для пользователей внутри определённого пространства имен:

```
apiVersion: deckhouse.io/v1alpha1
kind: AuthorizationRule
metadata:
  name: beeline
spec:
  accessLevel: Admin
  subjects:
  - kind: Admin
    name: admin@example.com
```

Пример использования ClusterAuthorizationRule для установки правил доступа для пользователей как на уровне всего кластера, так и на уровне определенных пространств имен:

```
apiVersion: deckhouse.io/v1
kind: ClusterAuthorizationRule
metadata:
  name: test-rule
spec:
  subjects:
  - kind: User
    name: some@example.com
  - kind: ServiceAccount
    name: gitlab-runner-deploy
    namespace: d8-service-accounts
  - kind: Group
    name: some-group-name
  accessLevel: PrivilegedUser
  namespaceSelector:
    labelSelector:
      matchExpressions:
      - key: stage
        operator: In
        values:
        - test
        - review
      matchLabels:
        team: frontend
```

4.11 Экспорт данных

ПО «Deckhouse Platform» предоставляет возможность администратору кластера управлять экспортом данных при помощи объектов DataExport.

Экспорт данных возможен только при выполнении следующих условий:

- используется thin-том;
- включён модуль storage-volume-data-manager;
- в кластере установлен snapshot-controller;
- используемый CSI-драйвер поддерживает ресурс VolumeSnapshot.

Для того, чтобы создать объект DataExport:

1. Выведите имя VolumeSnapshotClass (данный ресурс создаётся автоматически при включённом snapshot-controller для CSI-драйверов, поддерживающих VolumeSnapshot и нужен для экспорта данных). Например, sds-local-volume-snapshot-class:

```
d8 k get volumesnapshotclass


Sds-local-volume-snapshot-class
local.csi.storage.deckhouse.io
Delete
22h
```

2. Создайте объект VolumeSnapshot:

```
d8 k apply -f -<<EOF
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: my-snapshot
  namespace: <имя пространства имён, где находится PVC>
spec:
  volumeSnapshotClassName: <имя VolumeSnapshotClass>
  source:
    persistentVolumeClaimName: <имя PVC для экспорта данных>
EOF
```

3. Убедитесь, что VolumeSnapshot создан и готов к использованию (это занимает несколько минут):

```
d8 k -n <имя пространства имён> get volumesnapshot my-snapshot


NAMESPACE
<имя пространства имён>

NAME
```

```
My-snapshot

READYTOUSE
true


SOURCEPVC
test-pvc-for-snapshot

SOURCESNAPSHOTCONTENT

RESTORESIZE
2Gi

SNAPSHOTCLASS
sds-local-volume-snapshot-class

SNAPSHOTCONTENT
snapcontent-faf2ab1f-891d-4e5e-972c-334a490c99d8
```

4. Экспортируйте данные при помощи команды:

```
d8 data create export-name snapshot/my-snapshot
```

5. После выполнения команды объект DataExport будет создан в том же пространстве имён, что и исходный ресурс.

4.12 Обновление

Обновление версии DKP CSE на v1.73.2 возможно с версии v1.67.4 и с версий v1.73.x.

При наличии в кластере узлов с ОС Astra Linux ознакомьтесь с разделом 4.12.3 «Особенности обновления в Astra Linux».

4.12.1 Обновление DKP CSE с версии 1.67.4 на версию 1.73.2

4.12.1.1 Подготовка к обновлению

Перед обновлением:

1. Загрузите в ваш репозиторий образов ПО «Deckhouse Platform» предоставленное вам обновление.

2. Убедитесь в отсутствии в кластере алертов, кроме D8DeckhouseIsNotOnReleaseChannel.

3. Убедитесь, что в очередях ПО «Deckhouse Platform» нет ошибок, с помощью команды:

```
d8 s queue list
```

Пример вывода команды пустых очередей без ошибок:

```
Summary:
- 'main' queue: empty.
- 103 other queues (0 active, 103 empty): 0 tasks.
- no tasks to handle.
```

4. При наличии NodeGroupConfiguration (NGC) sysctl-tune-fstec добавьте аннотации и лейбл. Проверка наличия:

```
d8 k get ngc | grep sysctl-tune-fstec
```

Пример вывода:

```
sysctl-tune-fstec                          100    ["*"]        ["*"]
```

Команды для добавления аннотаций и лейбла:

```
d8 k annotate ngc sysctl-tune-fstec meta.helm.sh/release-namespace=d8-system
d8 k annotate ngc sysctl-tune-fstec meta.helm.sh/release-name=node-manager
d8 k label ngc sysctl-tune-fstec app.kubernetes.io/managed-by=Helm
```

5. Если вы используете local-path-provisioner в качестве драйвера хранилища для ваших stateful или модулей платформы:

5.1. В случае, если включен модуль loki, потребуется совершить ряд действий для корректного обновления платформы.

5.2. Проверьте состояние модуля loki, выполнив команду:

```
  d8 k get module | grep loki
```

5.2.1. В случае выключенного модуля пример вывода будет следующим:

```
loki                                       462       Embedded    Downloaded    False
False
```

Дальнейших действий с модулем loki не требуется.

5.2.2. В случае включенного модуля пример вывода будет следующим:

```
loki                                     462       Embedded    Ready         True
True
```

5.3. Убедитесь, что в хранилище достаточно свободного дискового пространства для расширения хранилища данных модуля loki (не менее 50 Gi).

5.4. Измените размер PVC для loki, выполнив команду:

```
d8 k -n d8-monitoring edit pvc storage-loki-0
```

Выставив значение spec.resources.request.storage: 50Gi, как в примере:

```
...
spec:
   accessModes:
  - ReadWriteOnce
   resources:
      requests:
      storage: 50Gi
...
```

5.5. Измените размер PV, которую использует модуль loki.

5.5.1. Получите имя PV для изменения, выполнив команду:

```
d8 k get pv | grep loki
```

Пример вывода:

```
pvc-09ea47fe-b87a-4bc7-985b-fe2c319c11cd   2Gi          RWO          Delete
      Bound d8-monitoring/storage-loki-0
localpath         <unset>                          3m40s
```

Искомый PV должен находиться в пространстве имен d8-monitoring. В данном случае название PV для изменения будет pvc-09ea47fe-b87a-4bc7-985b-fe2c319c11cd.

5.5.2. Измените размер PV с именем, полученным из предыдущей команды:

```
d8 k edit pv pvc-09ea47fe-b87a-4bc7-985b-fe2c319c11cd
```

Выставив значение spec.capacity.storage: 50Gi, как в примере:

```
...
   spec:
      accessModes:
      - ReadWriteOnce
      capacity:
      storage: 50Gi
      claimRef:
...
```

5.6. Убедитесь, что PV для модуля loki имеет новый размер, выполнив команду:

```
d8 k get pv | grep loki
```

Пример вывода:

```
pvc-09ea47fe-b87a-4bc7-985b-fe2c319c11cd   50Gi         RWO          Delete
      Bound d8-monitoring/storage-loki-0
localpath         <unset>                          11m
```

5.7. Удалите StatefulSet loki, выполнив команду:

```
d8 k --as=system:serviceaccount:d8-system:deckhouse -n d8-monitoring delete sts
loki
```

Для проверки успешного завершения удаления выполните команду:

```
d8 k -n d8-monitoring get po | grep loki
```

Вывод команды должен быть пустым.

5.8. Добавьте в ресурс ModuleConfig loki параметр spec.settings.diskSizeGigabytes: 50, выполнив команду:

```
d8 k edit mc loki
```

Пример конфигурации:

```
...
     spec:
     enabled: true
     settings:
     diskSizeGigabytes: 50
     retentionPeriodHours: 24
     storageClass: localpath
     version: 1
     ...
```

5.9. Дождитесь перехода пода loki в состояние Running. Проверьте переход с помощью команды:

```
d8 k -n d8-monitoring get po | grep loki
```

Пример успешного вывода:

```
loki-0                          2/2    Running    0              5m28s
```

6. Если вы используете csi-nfs в качестве драйвера хранилища для ваших statefullset или модулей платформы:

6.1. Для корректной работы модуля csi-nfs после обновления требуется включение модуля snapshot-controller. Проверьте состояние модуля snapshot-controller, выполнив команду:

```
d8 k get modules | grep snapshot-controller
```

6.1.1. В случае выключенного модуля пример вывода будет следующим:

```
snapshot-controller                    37       Embedded   Downloaded   False
False
```

Для включения модуля snapshot-controller выполните команду:

```
d8 p module enable snapshot-controller
```

Пример вывода команды:

```
Module snapshot-controller enabled
```

Дождитесь разбора очередей deckhouse. Проверка очередей deckhouse:

```
d8 s queue list
```

Пример вывода пустых очередей без ошибок:

```
Summary:
    - 'main' queue: empty.
    - 103 other queues (0 active, 103 empty): 0 tasks.
    - no tasks to handle.
```

Проверьте состояние модуля snapshot-controller, выполнив команду:

```
d8 k get modules | grep snapshot-controller
```

Пример вывода при включенном модуле:

```
snapshot-controller                    37       Embedded   Ready        True
True
```

6.1.2. В случае включенного модуля пример вывода будет следующим:

```
snapshot-controller                    37       Embedded   Ready        True
True
```

Дальнейших действий с модулем `snapshot-controller` не требуется.

6.2. В случае, если включен модуль loki потребуется совершить ряд действий для корректного обновления платформы.

6.3. Проверьте состояние модуля loki, выполнив команду:

```
d8 k get module | grep loki
```

6.3.1. В случае выключенного модуля пример вывода будет следующим:

```
loki                                   462      Embedded   Downloaded   False
False
```

Дальнейших действий с модулем loki не требуется.

6.3.2. В случае включенного модуля пример вывода будет следующим:

```
loki                                   462      Embedded   Ready        True
True
```

6.4. Убедитесь, что в хранилище достаточно свободного дискового пространства для расширения хранилища данных модуля loki (не менее 50 Gi)

6.5. Измените размер PVC для loki, выполнив команду:

```
d8 k -n d8-monitoring edit pvc storage-loki-0
```

Выставив значение spec.resources.request.storage: 50Gi, как в примере:

```
...
spec:
   accessModes:
  - ReadWriteOnce
   resources:
      requests:
      storage: 50Gi
...
```

6.6. Убедитесь, что PV для модуля loki имеет новый размер, выполнив команду:

```
d8 k get pv | grep loki
```

Пример вывода:

```
pvc-09ea47fe-b87a-4bc7-985b-fe2c319c11cd   50Gi        RWO          Delete
      Bound d8-monitoring/storage-loki-0
localpath         <unset>                          11m
```

6.7. Удалите StatefulSet loki, выполнив команду:

```
d8 k --as=system:serviceaccount:d8-system:deckhouse -n d8-monitoring delete sts
loki
```

Для проверки успешного завершения удаления выполните команду:

```
d8 k -n d8-monitoring get po | grep loki
```

Вывод команды должен быть пустым.

6.8. Добавьте в ресурс ModuleConfig loki параметр spec.settings.diskSizeGigabytes: 50, выполнив команду:

```
d8 k edit mc loki
```

Пример конфигурации:

```
...
  spec:
    enabled: true
    settings:
      diskSizeGigabytes: 50
      retentionPeriodHours: 24
      storageClass: nfs-storage-class
    version: 1
  ...
```

6.9. Дождитесь перехода пода loki в состояние Running. Для проверки выполните команду:

```
d8 k -n d8-monitoring get po | grep loki
```

Пример успешного вывода:

```
loki-0                        2/2    Running    0              5m28s
```

7. Если вы используете sds-local-volume в качестве драйвера хранилища для ваших модулей:

7.1. Для корректной работы модуля sds-local-volume после обновления требуется включение модуля snapshot-controller.

Проверьте состояние модуля snapshot-controller, выполнив команду:

```
d8 k get modules | grep snapshot-controller
```

7.1.1. В случае выключенного модуля пример вывода будет следующим:

```
snapshot-controller                    37       Embedded   Downloaded   False
False
```

Для включения модуля snapshot-controller выполните команду:

```
d8 p module enable snapshot-controller
```

Пример вывода команды:

```
Module snapshot-controller enabled
```

Дождитесь разбора очередей deckhouse. Проверка очередей deckhouse:

```
d8 s queue list
```

Пример вывода пустых очередей без ошибок:

```
Summary:
    - 'main' queue: empty.
    - 103 other queues (0 active, 103 empty): 0 tasks.
    - no tasks to handle.
```

Проверьте состояние модуля snapshot-controller, выполнив команду:

```
d8 k get modules | grep snapshot-controller
```

Пример вывода при включенном модуле:

```
snapshot-controller                    37       Embedded   Ready        True
True
```

7.1.2. В случае включенного модуля пример вывода будет следующим:

```
snapshot-controller                    37       Embedded   Ready        True
True
```

Дальнейших действий с модулем snapshot-controller не требуется.

7.2. В случае, если включен модуль loki потребуется совершить ряд действий для корректного обновления платформы.

7.3. Проверьте состояние модуля loki, выполнив команду:

```
d8 k get module | grep loki
```

7.3.1. В случае выключенного модуля пример вывода будет следующим:

```
loki                                   462      Embedded   Downloaded   False
False
```

Дальнейших действий с модулем loki не требуется.

7.3.2. В случае включенного модуля пример вывода будет следующим:

```
loki                                   462      Embedded   Ready        True
True
```

7.4. Убедитесь, что в хранилище достаточно свободного дискового пространства для расширения хранилища данных модуля loki (не менее 50 Gi)

7.5. Измените размер PVC для loki, выполнив команду:

```
d8 k -n d8-monitoring edit pvc storage-loki-0
```

Выставив значение spec.resources.request.storage: 50Gi, как в примере:

```
...
spec:
   accessModes:
  - ReadWriteOnce
   resources:
      requests:
      storage: 50Gi
...
```

7.6. Убедитесь, что PV для модуля loki имеет новый размер, выполнив команду:

```
d8 k get pv | grep loki
```

Пример вывода:

```
pvc-09ea47fe-b87a-4bc7-985b-fe2c319c11cd   50Gi        RWO          Delete
      Bound d8-monitoring/storage-loki-0
localpath         <unset>                          11m
```

7.7. Удалите StatefulSet loki, выполнив команду:

```
d8 k --as=system:serviceaccount:d8-system:deckhouse -n d8-monitoring delete sts
loki
```

Для проверки успешного завершения удаления выполните команду:

```
d8 k -n d8-monitoring get po | grep loki
```

Вывод команды должен быть пустым.

7.8. Добавьте в ресурс ModuleConfig loki параметр spec.settings.diskSizeGigabytes: 50, выполнив команду:

```
d8 k edit mc loki
```

Пример конфигурации:

```
 ...
   spec:
     enabled: true
     settings:
       diskSizeGigabytes: 50
       retentionPeriodHours: 24
       storageClass: local-storage-class
     version: 1
   ...
```

7.9. Дождитесь перехода пода loki в состояние Running. Для проверки выполните команду:

```
 d8 k -n d8-monitoring get po | grep loki
```

Пример успешного вывода:

```
loki-0                       2/2    Running    0              5m28s
```

8. Если вы используете csi-ceph в качестве драйвера хранилища для ваших модулей:

8.1. Для корректной работы модуля csi-ceph после обновления требуется включение модуля snapshot-controller.

Проверьте состояние модуля snapshot-controller, выполнив команду:

```
d8 k get modules | grep snapshot-controller
```

8.1.1. В случае выключенного модуля пример вывода будет следующим:

```
snapshot-controller                    37       Embedded   Downloaded   False
False
```

Для включения модуля snapshot-controller выполните команду:

```
d8 p module enable snapshot-controller
```

Пример вывода команды:

```
Module snapshot-controller enabled
```

Дождитесь разбора очередей deckhouse. Проверка очередей deckhouse:

```
d8 s queue list
```

Пример вывода пустых очередей без ошибок:

```
Summary:
    - 'main' queue: empty.
    - 103 other queues (0 active, 103 empty): 0 tasks.
    - no tasks to handle.
```

Проверьте состояние модуля snapshot-controller, выполнив команду:

```
d8 k get modules | grep snapshot-controller
```

Пример вывода при включенном модуле:

```
snapshot-controller                    37       Embedded   Ready        True
True
```

8.1.2. В случае включенного модуля пример вывода будет следующим:

```
snapshot-controller                    37       Embedded   Ready        True
True
```

Дальнейших действий с модулем snapshot-controller не требуется.

8.2. В случае, если включен модуль loki потребуется совершить ряд действий для корректного обновления платформы.

8.3. Проверьте состояние модуля loki, выполнив команду:

```
d8 k get module | grep loki
```

8.3.1. В случае выключенного модуля пример вывода будет следующим:

```
loki                                   462      Embedded   Downloaded   False
False
```

Дальнейших действий с модулем loki не требуется.

8.3.2. В случае включенного модуля пример вывода будет следующим:

```
 loki                                   462       Embedded   Ready        True
 True
```

8.4. Убедитесь, что в хранилище достаточно свободного дискового пространства для расширения хранилища данных модуля loki (не менее 50 Gi)

8.5. Измените размер PVC для loki, выполнив команду:

```
d8 k -n d8-monitoring edit pvc storage-loki-0
```

Выставив значение spec.resources.request.storage: 50Gi, как в примере:

```
...
spec:
   accessModes:
  - ReadWriteOnce
   resources:
      requests:
      storage: 50Gi
...
```

8.6. Убедитесь, что PV для модуля loki имеет новый размер, выполнив команду:

```
d8 k get pv | grep loki
```

Пример вывода:

```
pvc-09ea47fe-b87a-4bc7-985b-fe2c319c11cd   50Gi        RWO          Delete
      Bound d8-monitoring/storage-loki-0
localpath         <unset>                          11m
```

8.7. Удалите StatefulSet loki, выполнив команду:

```
d8 k --as=system:serviceaccount:d8-system:deckhouse -n d8-monitoring delete sts
loki
```

Для проверки успешного завершения удаления выполните команду:

```
d8 k -n d8-monitoring get po | grep loki
```

Вывод команды должен быть пустым.

8.8. Добавьте в ресурс ModuleConfig loki параметр spec.settings.diskSizeGigabytes: 50, выполнив команду:

```
d8 k edit mc loki
```

Пример конфигурации:

```
...
  spec:
    enabled: true
    settings:
      diskSizeGigabytes: 50
      retentionPeriodHours: 24
      storageClass: ceph-rbd-sc
    version: 1
  ...
```

8.9. Дождитесь перехода пода loki в состояние Running. Для проверки выполните команду:

```
d8 k -n d8-monitoring get po | grep loki
```

Пример успешного вывода:

```
loki-0                        2/2    Running    0              5m28s
```

9. Переведите обновления версии Kubernetes в кластере в ручной режим.

9.1. Проверьте версию Kubernetes внутри ресурса ClusterConfiguration, выполнив команду:

```
d8 p edit cluster-configuration
```

- В случае, если значение параметра kubernetesVersion равно "1.29", действий не требуется.

- В случае, если значение параметра kubernetesVersion равно "1.27", установите значения параметра kubernetesVersion: равным "1.29" и дождитесь обновления версии Kubernetes на всех узлах и разбора очередей ПО «Deckhouse Platform».

- В случае, если значение параметра kubernetesVersion равно Automatic, установите значения параметра kubernetesVersion: равным "1.29".

9.2. Проверка очередей ПО «Deckhouse Platform»:

```
d8 s queue list
```

Пример вывода команды пустых очередей без ошибок:

```
Summary:
  - 'main' queue: empty.
  - 103 other queues (0 active, 103 empty): 0 tasks.
  - no tasks to handle.
```

9.3. Проверка статуса обновления всех узлов в кластере.

```
d8 k get ng
```

Пример вывода команды с успешно обновленными узлами:

```
NAME       TYPE   READY    NODES    UPTODATE    INSTANCES    DESIRED    MIN    MAX
STANDBY    STATUS    AGE    SYNCED
  master    Static    3          3      3
                  34h    True
  system    Static    2          2      2
                  34h    True
  worker    Static    3          3      3
                  34h    True
```

Количество узлов UPTODATE должно быть равно количеству NODES и READY.

10. Убедитесь в отсутствии релизного канала в ресурсе ModuleConfig deckhouse.

10.1. Проверьте актуальность ресурса ModuleConfig с помощью команды:

```
d8 k get mc deckhouse -o yaml
```

Пример вывода:

```
apiVersion: deckhouse.io/v1alpha1
  kind: ModuleConfig
  metadata:
    creationTimestamp: "<some-timestamp>"
    finalizers:
    - modules.deckhouse.io/module-config
    generation: 1
    name: deckhouse
    resourceVersion: "<some-version>"
    uid: <some-uid>
  spec:
    enabled: true
    settings:
      bundle: Default
      logLevel: Info
    version: 1
  status:
    message: ""
    version: "1"
```

При наличии поля spec.settings.releaseChannel удалите его, выполнив команду:

```
kubectl patch mc deckhouse --type='merge'
-p='{"spec":{"settings":{"releaseChannel": null}}}'
```

Результат выполнения команды:

```
moduleconfig.deckhouse.io/deckhouse patched
```

4.12.1.2 Обновление версии платформы

1. Примените NodeGroupConfiguration с именем bashible-migrations-002.sh следующего вида:

```
apiVersion: deckhouse.io/v1alpha1
kind: NodeGroupConfiguration
metadata:
```

```
  name: bashible-migrations-002.sh
spec:
  bundles: ['*']
  content: |
    if ! declare -F bb-d8-node-name >/dev/null; then
      echo "Function 'bb-d8-node-name' NOT found, restarting 'bashible' service..."
      sleep 30
      systemctl restart bashible.service
      echo "Service 'bashible' restarted."
    fi
  nodeGroups:
  - '*'
  weight: 002
```

2. Поменяйте образ в Deployment deckhouse:

```
d8 k --as=system:serviceaccount:d8-system:deckhouse set image deployment -n d8-system deckhouse deckhouse=<your-cse-registry>/deckhouse/cse:v1.73.2
```

3. Дождитесь разбора очередей ПО «Deckhouse Platform» и полного обновления всех узлов в кластере:

3.1. Проверка очередей ПО «Deckhouse Platform».

```
d8 s queue list
```

Пример вывода команды пустых очередей без ошибок:

```
Summary:
  - 'main' queue: empty.
  - 103 other queues (0 active, 103 empty): 0 tasks.
  - no tasks to handle.
```

3.2. Проверка статуса обновления всех узлов в кластере.

```
d8 k get ng
```

Пример вывода команды с успешно обновленными узлами.

```
     NAME      TYPE      READY    NODES    UPTODATE    INSTANCES    DESIRED    MIN    MAX
STANDBY   STATUS   AGE    SYNCED
          master         Static         3                    3                     3
34h   True
          system         Static         2                    2                     2
34h   True
          worker         Static         3                    3                     3
34h   True
```

Количество узлов UPTODATE должно быть равно количеству NODES и READY.

4. Обращайте внимание на появление алертов вида NodeRequiresDisruptionApprovalForUpdate. В случае возникновения следуйте инструкции в алерте.

5. Обновите версию control-plane-manager, изменив поле spec.version в mc control-plane-manager с 1 на 2.

```
d8 k edit mc control-plane-manager
```

Пример ресурса ModuleConfig с валидной версией:

```
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
spec:
  settings:
  ...
  version: 2
```

6. Обновите версию Kubernetes до 1.31. Выполните команду:

```
d8 p edit cluster-configuration
```

В ресурсе ClusterConfiguration измените версию Kubernetes на Automatic, дождитесь обновления версии Kubernetes на всех узлах:

- Узлы с политикой применения деструктивных изменений Auto будут обновлены в автоматическом режиме.

- Узлы с политикой применения деструктивных изменений Manual потребуют ручного применения изменений, о чем будет свидетельствовать алерт вида NodeRequiresDisruptionApprovalForUpdate. Для применения изменений следуйте инструкции в алерте.

Команда для получения версии Kubernetes на узлах:

```
d8 k get no -o custom-columns=NAME:.metadata.name,VERSION:.status.nodeInfo.kubeletVersion
```

Пример вывода:

```
NAME                                        VERSION
example-cse-master-0.ru-central1.internal   v1.31.13
example-cse-master-1.ru-central1.internal   v1.31.13
example-cse-master-2.ru-central1.internal   v1.31.13
example-cse-system-0.ru-central1.internal   v1.31.13
example-cse-system-1.ru-central1.internal   v1.31.13
example-cse-worker-0.ru-central1.internal   v1.31.13
example-cse-worker-1.ru-central1.internal   v1.31.13
example-cse-worker-2.ru-central1.internal   v1.31.13
```

Проверка статуса обновления узлов в каждой нодгруппе:

```
d8 k get ng
```

Пример вывода команды с успешно обновленными узлами:

```
NAME       TYPE       READY    NODES    UPTODATE    INSTANCES    DESIRED    MIN    MAX
STANDBY   STATUS   AGE   SYNCED
master           Static           3                        3                          3
34h   True
system           Static           2                        2                          2
34h   True
worker           Static           3                        3                          3
34h   True
```

Количество узлов `UPTODATE` должно быть равно количеству `NODES` и `READY`

7. Удалите NodeGroupConfiguration с именем bashible-migrations-002.sh.

```
d8 k delete ngc bashible-migrations-002.sh
```

8. Установите канал обновлений в ресурс Moduleconfig deckhouse. Выполните команду

```
d8 k edit mc deckhouse
```

В открывшемся окне установите параметр spec.settings.releaseChannel: LTS. Пример ресурса ModuleConfig с заданным параметром.

```
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: deckhouse
spec:
  enabled: true
  settings:
    bundle: Default
    logLevel: Info
    releaseChannel: LTS
  version: 1
```

4.12.1.3 Включение контроля целостности

Для включения контроля целостности требуется предварительная миграция данных etcd.

1. Переведите режим работы контроля целостности на режим Migrate. Пример ресурса ModuleConfig control-plane-manager с режимом Migrate:

```
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: control-plane-manager
spec:
  settings:
```

```
    apiserver:
      signature: Migrate
...
```

2. Дождитесь очистки очередей ПО «Deckhouse Platform».

```
d8 system queue list
```

Пример вывода команды пустых очередей без ошибок:

```
Summary:
- 'main' queue: empty.
- 103 other queues (0 active, 103 empty): 0 tasks.
- no tasks to handle.
```

3. Осуществите миграцию данных. Для этого запустите утилиту от пользователя root с master-узла кластера со следующими аргументами:

```
d8 tools sig-migrate
```

Рекомендуется запуск в эмуляторе терминала по типу screen или tmux. В случае отсутствия данных утилит на master узле необходима их установка.

По окончании выполнения, если на какие-либо объекты не удалось установить аннотацию, команда автоматически выведет предупреждающее сообщение со следующей информацией:

- Количество объектов, для которых не удалось произвести миграцию.

- Пути к файлам с логами ошибок.

- Инструкции по расследованию и запуску повторной попытки.

Пример вывода при успешном завершении миграции:

```
Retrying failed annotations from previous runs...
Loaded 3 objects for retry from /tmp/failed_annotations.txt.
Progress: [100%] Annotating: Kind=nodegroups, Namespace=clusterwide, Name=system
```

Пример вывода при наличии ошибок:

```
⚠️ Migration completed with 5 failed object(s).

Some objects could not be annotated. Please check the error details:

- Error log file: `/tmp/failed_errors.txt`
- Failed objects list: `/tmp/failed_annotations.txt`

To investigate the issues:
1. Review the error log file to understand why objects failed
2. Check permissions and resource availability
3. Retry migration for failed objects only using:
```

```
   d8 tools sig-migrate --retry
```

При получении ошибки, запустите миграцию повторно (с флагом --retry)

```
d8 tools sig-migrate --retry
```

Повторять процедуру миграции следует до тех пор, пока утилита не сообщит, что больше нет объектов, для которых не удалось установить аннотацию. А также до прекращения алерта D8SignatureErrorsDetected.

Если повторный запуск миграции не проходит с третьего раза, выполните следующую команду:

```
sed -i '/nodegroups/s/v1alpha[^|]*/v1/g' /tmp/failed_annotations.txt
```

И повторите миграцию:

```
d8 tools sig-migrate --retry
```

В случае многократного (больше 10 раз) получения ошибки обратитесь в техническую поддержку.

4. Переведите режим работы контроля целостности на режим Enforce. Пример ресурса ModuleConfig control-plane-manager с режимом Enforce:

```
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: control-plane-manager
spec:
  settings:
    apiserver:
      signature: Enforce
...
```

5. Дождитесь очистки очередей ПО «Deckhouse Platform».

```
d8 system queue list
```

Пример вывода команды пустых очередей без ошибок:

```
Summary:
- 'main' queue: empty.
- 103 other queues (0 active, 103 empty): 0 tasks.
- no tasks to handle.
```

4.12.1.4 Обновление версии containerd

Внимание. Не обновляйте версию containerd до v2, если в кластере используется модуль csi-ceph. Пропустите этот этап.

Начиная с версии 1.73 в ПО «Deckhouse Platform» доступна возможность использовать container runtime обновленной версии — containerd v2. Containerd v2 можно использовать как основной container runtime на уровне всего кластера или для отдельных групп узлов. Использование containerd v2 обеспечивает более гибкое управление ресурсами, лучшую безопасность, за счет cgroups v2 и контроля целостности системных компонентов.

Требования к узлам для миграции на containerd v2:

- Поддержка CgroupsV2;
- Ядро Linux версии 5.8 и новее;
- Systemd версии 244 и новее;
- Поддержка модуля ядра erofs;
- Отсутствие кастомных конфигураций в /etc/containerd/conf.d.

Также ПО «Deckhouse Platform» выполняет дополнительную проверку используемой версии ядра Linux на наличие известных уязвимостей. При обнаружении уязвимой версии ядра использование containerd v2 на узле запрещается, даже если все остальные требования выполнены.

Например, в РЕД ОС могут использоваться версии ядра Linux из приведённых ниже диапазонов, которые приводят к автоматическому ограничению использования containerd v2:

- Ядра ветки 6.12.x с версиями 6.12.0–6.12.28 включительно. Исправление доступно, начиная с версии 6.12.29;
- Ядра ветки 6.14.x с версиями 6.14.0–6.14.6 включительно. Исправление доступно, начиная с версии 6.14.7.

При несоответствии любому из требований для миграции, ПО «Deckhouse Platform» добавляет на узел лейбл node.deckhouse.io/containerd-v2-unsupported. Если на узле есть кастомные конфигурации в /etc/containerd/conf.d, на него добавляется лейбл node.deckhouse.io/containerd-config=custom.

При наличии одного из этих лейблов смена параметра spec.cri.type для группы узлов будет недоступна. Узлы, которые не подходят под условия миграции можно посмотреть с помощью следующих команд:

```
d8 k get node -l node.deckhouse.io/containerd-v2-unsupported
```

```
d8 k get node -l node.deckhouse.io/containerd-config=custom
```

Также администратор может проверить конкретный узел на соответствие требованиям с помощью команд:

```
uname -r | cut -d- -f1
stat -f -c %T /sys/fs/cgroup
systemctl --version | awk 'NR==1{print $2}'
modprobe -qn erofs && echo "TRUE" || echo "FALSE"
ls -l /etc/containerd/conf.d
```

Включение containerd v2 возможно двумя способами:

1. Для всего кластера.

2. Для конкретной группы узлов.

Для включения containerd v2 для всего кластера, укажите значение ContainerdV2 в параметре defaultCRI ресурса ClusterConfiguration. Это значение будет применяться ко всем NodeGroup, в которых явно не указан spec.cri.type.

```
d8 p edit cluster-configuration
```

Пример конфигурации:

```
apiVersion: deckhouse.io/v1
kind: ClusterConfiguration
...
defaultCRI: ContainerdV2
...
```

Для включения containerd v2 для конкретной группы узлов, укажите ContainerdV2 в параметре spec.cri.type в объекте NodeGroup.

Рассмотрим на примере группы узлов worker.

```
d8 k edit ng worker
```

Пример конфигурации:

```
apiVersion: deckhouse.io/v1
kind: NodeGroup
metadata:
  name: worker
spec:
  cri:
    type: ContainerdV2
...
```

При переходе на containerd v2 ПО «Deckhouse Platform» начнет поочерёдное обновление узлов. Обновление узла приводит к прерыванию работы размещенной на нем нагрузки

(disruptive-обновление). На процесс обновления узла влияют параметры применения disruptive-обновлений группы узлов (spec.disruptions.approvalMode NodeGroup):

- Узлы с политикой применения disruptive-обновлений Auto будут обновлены в автоматическом режиме

- Узлы с политикой применения disruptive-обновлений Manual потребуют ручного применения изменений, о чем будет свидетельствовать алерт вида NodeRequiresDisruptionApprovalForUpdate. Для применения изменений следуйте инструкции в алерте.

4.12.1.5 Проверка успешности обновления

4.12.1.5.1 В очередях ПО «Deckhouse Platform» нет заданий. Для проверки выполните команду:

```
d8 system queue list
```

Пример вывода команды пустых очередей без ошибок:

```
Summary:
- 'main' queue: empty.
- 103 other queues (0 active, 103 empty): 0 tasks.
- no tasks to handle.
```

4.12.1.5.2 Все узлы в кластере обновлены. Для проверки выполните команду:

```
d8 k get ng
```

Пример вывода команды с успешно обновленными узлами:

```
NAME       TYPE       READY    NODES    UPTODATE    INSTANCES    DESIRED    MIN    MAX
STANDBY   STATUS   AGE   SYNCED
     master          Static          3                        3                         3
34h   True
     system          Static          2                        2                         2
34h   True
     worker          Static          3                        3                         3
34h   True
```

Количество узлов UPTODATE должно быть равно количеству NODES и READY.

4.12.1.5.3 Образ контейнера в Deployment deckhouse имеет актуальный тег. Для проверки выполните команду:

```
d8 k -n d8-system get deploy/deckhouse -oyaml | grep image
```

Пример вывода:

```
image: <your-cse-registry>/deckhouse/cse:v1.73.2
```

4.12.1.5.4 В веб-интерфейсе кластера на главной странице отображается редакция платформы, новая версия платформы и релизный канал.

4.12.2 Обновление DKP CSE с версии 1.73.x (любая патч-версия)на версию 1.73.2

4.12.2.1 Подготовка к обновлению

Перед обновлением:

1. Убедитесь в отсутствии в кластере алертов, кроме SecurityEventsDetected.
2. Убедитесь, что в очередях DKP CSE нет ошибок.

   Выполните следующую команду:

```
d8 system queue list
```

   Пример вывода команды пустых очередей без ошибок:

```
Summary: - 'main' queue: empty. - 103 other queues (0 active, 103 empty): 0 tasks.
- no tasks to handle.
```

3. Установите желаемый режим обновлений в ModuleConfig deckhouse — заполните секцию параметров spec.settings.update.

   Выполните следующую команду для редактирования ModuleConfig deckhouse:

```
d8 k edit mc deckhouse
```

   Пример вывода команды:

```
        # ...
        spec:
          settings:
            update:
              windows:
              - from: '8:00'
                to: '15:00'
                days:
                - Tue
                - Sat
        # ...
```

Также вы можете заполнить параметр spec.settings.update.mode для выбора режима обновлений версии платформы.

Режим обновления Deckhouse на выбранном канале обновлений:

- AutoPatch — автоматический режим обновления для патч-версий.
- Для обновления минорной версии (например, с v1.69.* на v1.70.*) необходимо подтверждение.

Обновление патч-версии (например, с v1.70.1 на v1.70.2) применяется с учетом окон обновлений, если они заданы.

- Auto — автоматический режим обновления для всех версий.

Обновления минорной версии (например, с v1.69.* на v1.70.*) и патч-версии (например, с v1.70.1 на v1.70.2) применяются с учетом заданных окон обновлений, либо, если окна обновлений не заданы, по мере появления обновлений на соответствующем канале обновлений.

- Manual — ручной режим обновления для всех версий.

Для обновления и минорной, и патч-версии необходимо подтверждение.

Для подтверждения обновления версии установите поле approved в true в соответствующем объекте DeckhouseRelease.

Выбранный режим обновления применяется только при установленном канале обновлений.

По умолчанию: AutoPatch

Допустимые значения: AutoPatch, Auto, Manual

4.12.2.2 Обновление версии платформы

1. Загрузите предоставленное вам обновление в ваше хранилище образов контейнеров DKP CSE.

2. Обновление произойдет в соответствии с выставленными окнами и режимом обновлений.
3. Проверьте, что обновление прошло успешно.

4.12.2.3 Проверка успешности обновления

Проверка выполняется аналогично шагам, описанным в п. 4.12.1.5 Проверка успешности обновления.

4.12.3 Особенности обновления в Astra Linux

Если в Astra Linux включен режим замкнутой программной среды (ЗПС), то перед обновлением его нужно отключить **на всех узлах кластера.**

Для проверки, включён ли режим ЗПС, выполните на узле следующие команды:

```
astra-digsig-control is-enabled
cat /sys/digsig/elf_mode
```

Для ручного отключения режима ЗПС, выполните на узле следующие команды:

```
sed -i '/^DIGSIG_ELF_MODE=/d' /etc/digsig/digsig_initramfs.conf && echo
'DIGSIG_ELF_MODE=0' | tee -a /etc/digsig/digsig_initramfs.conf
 update-initramfs -u -k all
 systemctl reboot
```

В качестве альтернативного варианта, для отключения режима ЗПС можно использовать утилиту astra-digsig-control:

```
astra-digsig-control disable
```

После завершения обновления, создайте следующий NodeGroupConfiguration, для автоматического включения режима ЗПС на узлах кластера с Astra Linux:

```
apiVersion: deckhouse.io/v1alpha1
kind: NodeGroupConfiguration
metadata:
```

```
  name: astra-zps.sh
spec:
  weight: 98
  nodeGroups: [ "*" ] # ВЫ МОЖЕТЕ УКАЗАТЬ КОНРЕТНЫЕ NODEGROUP
  bundles: [ "astra" ]
  content: |
    # Copyright 2025 Flant JSC
    #
    # Licensed under the Apache License, Version 2.0 (the "License");
    # you may not use this file except in compliance with the License.
    # You may obtain a copy of the License at
    #
    #     http://www.apache.org/licenses/LICENSE-2.0
    #
    # Unless required by applicable law or agreed to in writing, software
    # distributed under the License is distributed on an "AS IS" BASIS,
    # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
    # See the License for the specific language governing permissions and
    # limitations under the License.

    modeswitch="$(astra-modeswitch get 2>/dev/null)"
    case "$modeswitch" in
      0|1|2) ;;
      *) modeswitch=0 ;;
    esac

    if (( modeswitch < 1 )); then
          bb-log-info "Astra modeswitch level $modeswitch detected, skip digital
signature provisioning"
      exit 1
    fi

    if ! grep -qw digsig_verif /proc/modules; then
      bb-log-warning "Module digsig_verif is not loaded"
      exit 1
    fi

    keys_dir="/etc/digsig/keys"
    digsig_conf="/etc/digsig/digsig_initramfs.conf"
```

```
digsig_mode="1"

mkdir -p "$keys_dir"
mkdir -p "$(dirname "$digsig_conf")"

_digsig_rebuild_initramfs() {
  bb-log-info "Rebuilding initramfs to refresh digital signature keys"
  update-initramfs -uk all
  bb-flag-set reboot
}

bb-event-on 'digsig-initramfs-update' '_digsig_rebuild_initramfs'

is_enabled_zps() {
  astra-digsig-control is-enabled >/dev/null 2>&1
}

sync_key() {
  local filename="$1"
  local payload="$2"
  local target="$keys_dir/$filename"
  local tmp_file

  tmp_file="$(bb-tmp-file)"
  printf '%s' "$payload" | base64 -d > "$tmp_file"
  bb-sync-file "$target" "$tmp_file" digsig-initramfs-update
  chmod 600 "$target"
  rm -f "$tmp_file"
}

sync_digsig_conf() {
  if ! is_enabled_zps; then
    if ! astra-digsig-control enable; then
      bb-log-error "Failed to enable Astra digital signature control"
      exit 1
    fi
    bb-flag-set reboot
  fi
}
```

```
    mkdir -p "$keys_dir/flant"

                                                sync_key           "flant/flant-2025.gpg"
mN4EaPngniMBAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQxAAMHAP9fv/SYqpOM5zm44CL7r+9A
Vj9uajRy/CpRTAzp2uI7fgACAgD8COKooOZRR9S9YxYDDhbRnIXJfwqcomcSK5arvOp+j8gAAQEBAIAAAAAA
AAAAAAAAAAAAAAFQ/ooYkpdhVMWc/Bk6zPWzAP9Ict4rRYNZqoxJTYx1HHRNP3DYIpj7Z06CiD3ligMevgD+
JEvcELVEzaAJpaESTiWhZRQWLSHwCGvkGXqFsL636G8AAQG0QdCQ0J4gItCk0LvQsNC90YIiICjQl9Cf0KEs
IERIIHByb2R1Y3Rpb24ga2V5KSA8c2VjdXJpdHlAZmxhbnQucnU+iJAEEyMMADgWIQSP/y49Jif7FuB0hfMV
qO+CTcPnQQUCaPngngIbAwULCQgHAgYVCgkICwIEFgIDAQIeAQIXgAAKCRAVqO+CTcPnQdnrAP0TrjvMKSqo
NCYkJumjd744RYUT7g/NZ5dnXukQet0wbAD/a7/aCKtLkL75878crt8E+CMnsEgGrD39RkeZNZyT7UqIdQQQ
IwwAHRYhBPq2qyAJ6YwXAMD+S7IL0fENyY9ZBQJo+eDBAAoJELIL0fENyY9ZsA8A/jlimDJzpBDvs2s1saEl
UBdQ77c2o0G6Y7TA9MSD2uGEAP9SSgNP2OBGXBg7ztG28iO6BBmuBv2Ut3nQ648AawoLfg==
                                                sync_key           "flant-2025-root.gpg"
mN4EaD7ryyMBAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAQxAAMHAP9fv/SYqpOM5zm44CL7r+9A
Vj9uajRy/CpRTAzp2uI7fgACAgD8COKooOZRR9S9YxYDDhbRnIXJfwqcomcSK5arvOp+j8gAAQEBAIAAAAAA
AAAAAAAAAAAAAAFQ/ooYkpdhVMWc/Bk6zPWzAP9mAxC2f4hBFjBuYwaoFNziI75LPCXLMgUpNmObeXFHJQD9
EBCPqnvFso5zQiB+4BfKjFWZoSH6V4PBa/JINtpYPxcAAQG0NNCQ0J4gwqvQpNC70LDQvdGCwrsgKGtleSBm
b3Igc2lnbmluZykgPDc3N0BmbGFudC5ydT6IkAQTIwwAOBYhBPq2qyAJ6YwXAMD+S7IL0fENyY9ZBQJoPuvL
AhsDBQsJCAcCBhUICQoLAgQWAgMBAh4BAheAAAoJELIL0fENyY9Z9BEA/0tB8rXfO2aLlpMLmyfPLDGvkURq
Ho+zJ6XRj9/aH+RpAP9e0QWrin5NI4pHPSOAuEx3pBT97nLNG3KOuDRdLLQrz4h1BBAjDAAdFiEEoS17W6/O
QNh/1B2vyC1J/DZ1tvoFAmg+68wACgkQyC1J/DZ1tvqEAwD7B3xlPHa+pONWB1kxVj9+SZY+08phl3KTKW20
Coq6EBAA/RCgTmuFPZDkQ+yRtK59TtefIzdTQq94ifhoxiSKsww7

    sync_digsig_conf
```

После создания NodeGroupConfiguration astra-zps.sh на узлах кластера будут выполнены следующие действия:

- Проверка загрузки модуля digsig_verif в ОС узла.
- Добавление публичных ключей, используемых для валидации бинарных файлов компонентов DKP CSE.
- Включение проверки подписи бинарных файлов.
- Обновление образов initramfs для всех установленных ядер в системе.
- Перезагрузка узла.

4.12.4 Возможные проблемы и пути их решения

4.12.4.1 Обновление cilium

В рамках обновления версии платформы с v1.67.4 на v1.73.2 происходит обновление Cilium с 1.14 на 1.17.

В случае, если обновление не продвигается (поды safe-agent-updater в пространстве имен d8-cni-cilium находятся в состоянии Init:4/5 на одном узле дольше 20 минут):

1. Выясните для каких узлов ПО «Deckhouse Platform» запрашивает согласие на деструктивные изменения:

```
d8 k get nodes -o json | jq '.items[] | select(.metadata.annotations."update.node.deckhouse.io/approved"=="") | .metadata.name' -r
```

Пример вывода:

```
example-cse-master-0.ru-central1.internal
example-cse-system-0.ru-central1.internal
```

2. Удалите под safe-agent-updater на этих нодах.

3. Далее поведение процесса обновления зависит от значения параметра spec.disruptions.approvalMode в нодгруппе:

При значении Auto обновление выполняется автоматически — версия Cilium обновляется поочерёдно на каждом узле с выполнением операций Cordon и Drain.

При значении Manual обновление узлов требует ручного подтверждения. Для разрешения обновления конкретного узла необходимо добавить аннотацию kubectl annotate node <NODE> update.node.deckhouse.io/disruption-approved=.

4.12.4.2 Потеря одного участника (member) etcd во время миграции версии etcd с 3.5.17 на 3.6.1

В рамках обновления ПО «Deckhouse Platform» происходит обновление etcd с версии 3.5.17 на 3.6.1, что в некоторых случаях приводит к переходу одного из подов d8-control-plane-manager в состояние CrashLoopBackOff. Такая ситуация возникает, если после обновления один из участников кластера etcd остается в статусе Unstarted. В результате данный узел исключается из кворума etcd.

Если в процессе обновления один из подов etcd-<nodename> в пространстве имен kube-system переходит в состояние CrashLoopBackOff и в кворуме etcd уже нет участника с именем этого control-plane узла, то нужно повторно добавить узел etcd.

Просмотр участников кворума etcd:

```
d8 k -n kube-system exec -ti $(d8 k -n kube-system get pod -l
component=etcd,tier=control-plane -o name | head -n1) -- etcdctl --cacert
/etc/kubernetes/pki/etcd/ca.crt --cert /etc/kubernetes/pki/etcd/ca.crt --key
/etc/kubernetes/pki/etcd/ca.key --endpoints https://127.0.0.1:2379/ member list -w
table
```

Пример вывода с пропавшим участником кворума в кластере с 3 узлами control-plane:

```
     +------------------+---------+-----------------------------------------+-----
----------------------+--------------------------+-------------+
     |         ID         | STATUS  |                       NAME                  |
PEER ADDRS          |       CLIENT ADDRS        | IS LEARNER |
     +------------------+---------+-----------------------------------------+-----
----------------------+--------------------------+-------------+
     |  898edf6255f45679  |  started  |  example-cse-master-1.ru-central1.internal  |
https://10.241.32.23:2380 | https://10.241.32.23:2379 |      false |
     |  988286c43353ca92  |  started  |  example-cse-master-0.ru-central1.internal  |
https://10.241.32.6:2380  | https://10.241.32.6:2379  |      false |
     +------------------+---------+-----------------------------------------+-----
----------------------+--------------------------+-------------+
```

Для решения необходимо выполнить ряд действий на проблемном узле control-plane:

1. Перенесите папку с данными etcd: /var/lib/etcd.

```
mv /var/lib/etcd /tmp
```

2. Перенесите манифест статического пода etcd: /etc/kubernetes/manifests/etcd.yaml.

```
mv /etc/kubernetes/manifests/etcd.yaml /tmp
```

3. Перезапустите под d8-control-plane-manager на проблемном узле control-plane.

3.1. Получите имя пода с помощью команды.

```
d8 k -n kube-system get po -owide | grep control-plane | grep <node-name>
```

3.2. Удалите под, имя которого было получено в предыдущем шаге.

```
d8 k -n kube-system delete po <pod-name>
```

4. Убедитесь в восстановлении кворума etcd.

```
d8 k -n kube-system exec -ti $(d8 k -n kube-system get pod -l
component=etcd,tier=control-plane -o name | head -n1) -- etcdctl --cacert
```

```
/etc/kubernetes/pki/etcd/ca.crt    --cert    /etc/kubernetes/pki/etcd/ca.crt    --key
/etc/kubernetes/pki/etcd/ca.key  --endpoints  https://127.0.0.1:2379/  member  list  -w
table
```

Целевое состояние - количество участников etcd равно количеству узлов control-plane. Пример вывода для трёх узлов:

```
     +------------------+---------+-----------------------------------------+-----
----------------------+--------------------------+-------------+
     |         ID         | STATUS   |                       NAME                    |
PEER ADDRS          |       CLIENT ADDRS        | IS LEARNER |
     +------------------+---------+-----------------------------------------+-----
----------------------+--------------------------+-------------+
     |  898edf6255f45679  |  started  |  example-cse-master-1.ru-central1.internal  |
https://10.241.32.23:2380 | https://10.241.32.23:2379 |      false |
     |  988286c43353ca92  |  started  |  example-cse-master-0.ru-central1.internal  |
https://10.241.32.6:2380  | https://10.241.32.6:2379  |      false |
     |  bb3fcfe1f7464434  |  started  |  example-cse-master-2.ru-central1.internal  |
https://10.241.32.21:2380 | https://10.241.32.21:2379 |      false |
     +------------------+---------+-----------------------------------------+-----
----------------------+--------------------------+-------------+
```

4.13 Создание самоподписанного сертификата

TLS-сертификат необходим для организации работы по протоколу HTTPS. Если нет сертификата, выпущенного доверенным центром сертификации, то можно сгенерировать самоподписанный сертификат, ориентируясь на шаги, приведенные в данном разделе. Для некоторых версий операционных систем шаги могут отличаться.

Сгенерируйте самоподписанный сертификат, указав свой домен от кластера и репозитория в переменных CLUSTER_DNS_NAME и REGISTRY_DNS_NAME.

Укажите актуальные данные в переменных окружения (они потребуются для выполнения команд далее):

```
CLUSTER_DNS_NAME="cluster-domain.test"
REGISTRY_DNS_NAME="registry.cluster-domain.test"
```

Создайте директорию для сертификатов:

```
mkdir ~/ca && cd ~/ca
```

Сгенерируйте сертификат корневого ЦС (Root CA)

```
openssl req -x509 -newkey rsa:2048 -nodes -days 3650 -keyout rootCA.key -out
rootCA.crt -subj "/CN=Root CA" -extensions v3_ca -config <(echo -e
"[req]\ndistinguished_name=req_distinguished_name\n[ v3_ca
]\nbasicConstraints=critical,CA:TRUE\nkeyUsage=critical,keyCertSign,cRLSign\n[req_di
stinguished_name]")
```

Сгенерируйте сертификат промежуточного ЦС (Intermediate CA):

```
openssl req -newkey rsa:2048 -nodes -keyout intermediateCA.key -out
intermediateCA.csr -subj "/CN=Intermediate CA"
openssl x509 -req -in intermediateCA.csr -CA rootCA.crt -CAkey rootCA.key
-CAcreateserial -out intermediateCA.crt -days 3648 -extensions v3_ca -extfile <(echo
-e
"[v3_ca]\nbasicConstraints=critical,CA:TRUE,pathlen:0\nkeyUsage=critical,keyCertSign
,cRLSign\n")
```

Сгенерируйте wildcard-сертификат:

```
openssl req -newkey rsa:2048 -nodes -keyout wildcard.key -out wildcard.csr -subj
"/CN=*.$CLUSTER_DNS_NAME"
openssl x509 -req -in wildcard.csr -CA intermediateCA.crt -CAkey intermediateCA.key
-CAcreateserial -out wildcard.crt -days 3646 -extensions v3_req -extfile <(echo -e
"[ v3_req
]\nkeyUsage=critical,digitalSignature,keyEncipherment\nextendedKeyUsage=serverAuth,c
lientAuth\nsubjectAltName=DNS:*.$CLUSTER_DNS_NAME,DNS:$CLUSTER_DNS_NAME,DNS:$REGISTR
Y_DNS_NAME")
```

Проверьте корректность выпущенного сертификата:

```
openssl x509 -noout -text -in ~/ca/wildcard.crt
```

Добавьте сертификат в хранилище доверенных центров сертификации ОС (пример для ОС с менеджером пакетов apt):

```
apt install -y ca-certificates
cp ~/ca/rootCA.crt /usr/local/share/ca-certificates/self_signed_rootCA.crt
update-ca-certificates
```

4.14 Логирование

ПО «Deckhouse Platform» предусмотрен сбор и доставка логов из узлов и подов кластера во внутреннюю или внешние системы хранения. Предоставляются возможности:

- собирать логи из всех или отдельных подов и пространств имён;
- фильтровать логи по лейблам, содержимому сообщений и другим признакам;
- направлять логи одновременно в несколько хранилищ (например, Loki и Elasticsearch);
- обогащать логи метаданными Kubernetes;
- использовать буферизацию логов для повышения производительности.

Администраторам ПО «Deckhouse Platform» доступна настройка сбора и отправки логов с помощью трёх кастомных ресурсов:

- ClusterLoggingConfig — описывает источник логов на уровне кластера, включая правила сбора, фильтрации и парсинга;

- PodLoggingConfig — описывает источник логов в рамках заданного пространства имён, включая правила сбора, фильтрации и парсинга;

- ClusterLogDestination — задаёт параметры хранилища логов.

На основе этих ресурсов формируется процесс (pipeline), который используется в ПО «Deckhouse Platform» для чтения логов и дальнейшей работы с ними с помощью модуля log-shipper.

4.14.1 Настройка сбора и доставки логов

Ниже приведён вариант базовой конфигурации ПО «Deckhouse Platform», при котором логи со всех подов кластера отправляются в хранилище на базе Elasticsearch.

Для настройки выполните следующие шаги:

1. Включите модуль log-shipper с помощью следующей команды:

```
d8 platform module enable log-shipper
```

2. Создайте ресурс ClusterLoggingConfig, который задаёт правила сбора логов. Данный ресурс позволяет вам настроить сбор логов с подов в определенном пространстве имён и с определенным лейблом, настраивать парсинг многострочных логов и задавать другие правила.

В этом примере указывается, что нужно собирать логи со всех подов и отправлять их в Elasticsearch:

```
apiVersion: deckhouse.io/v1alpha1
kind: ClusterLoggingConfig
metadata:
  name: all-logs
spec:
  type: KubernetesPods
  destinationRefs:
  - es-storage
```

3. Создайте ресурс ClusterLogDestination, который описывает параметры отправки логов в хранилище. Данный ресурс позволяет вам указать одно или несколько хранилищ и описать параметры подключения, буферизации и дополнительные лейблы, которые будут применяться к логам перед отправкой.

В этом примере в качестве принимающего хранилища указан Elasticsearch:

```
apiVersion: deckhouse.io/v1alpha1
kind: ClusterLogDestination
metadata:
  name: es-storage
spec:
  type: Elasticsearch
  elasticsearch:
    endpoint: http://192.168.1.1:9200
    index: logs-%F
    auth:
      strategy: Basic
      user: elastic
      password: c2VjcmV0IC1uCg==
```

4.14.2 Преобразование логов

Есть возможность настроить один или несколько видов трансформаций, которые будут применяться к логам перед отправкой в хранилище.

Трансформация ParseMessage позволяет преобразовать строку в поле message в структурированный JSON-объект на основе одного или нескольких заданных форматов (String, Klog, SysLog и другие).

При использовании нескольких трансформаций ParseMessage преобразование строки (sourceFormat: String) должно выполняться в последнюю очередь.

Пример настройки преобразования записей смешанных форматов:

```
apiVersion: deckhouse.io/v1alpha2
kind: ClusterLogDestination
metadata:
  name: parse-json
spec:
  ...
  transformations:
  - action: ParseMessage
    parseMessage:
   sourceFormat: JSON
  - action: ParseMessage
```

```
    parseMessage:
      sourceFormat: Klog
  - action: ParseMessage
    parseMessage:
      sourceFormat: String
      string:
        targetField: "text"
```

Пример изначальной записи в логе:

```
/docker-entrypoint.sh: Configuration complete; ready for start up
{"level" : { "severity": "info" },"msg" : "fetching.module.release"}
I0505 17:59:40.692994   28133 klog.go:70] hello from klog
```

Результат преобразования:

```
{... "message": {
  "text": "/docker-entrypoint.sh: Configuration complete; ready for start up"
  }
}
{... "message": {
  "level" : "{ "severity": "info" }",
  "msg" : "fetching.module.release"
  }
}
{... "message": {
  "file":"klog.go",
  "id":28133,
  "level":"info",
  "line":70,
  "message":"hello from klog",
  "timestamp":"2025-05-05T17:59:40.692994Z"
  }
}
```

4.14.3 Замена лейблов

Трансформация ReplaceKeys позволяет рекурсивно заменить все совпадения шаблона source на значение target в указанных ключах лейблов.

Перед применением трансформации ReplaceKeys к полю message или его вложенным полям преобразуйте запись лога в структурированный объект с помощью трансформации ParseMessage.

Пример настройки замены точек на нижние подчеркивания в лейблах:

```
apiVersion: deckhouse.io/v1alpha2
kind: ClusterLogDestination
```

```
metadata:
  name: replace-dot
spec:
  ...
  transformations:
    - action: ReplaceKeys
      replaceKeys:
        source: "."
        target: "_"
        labels:
          - .pod_labels
```

Пример изначальной записи в логе:

```
{"msg" : "fetching.module.release"} # Лейбл пода pod.app=test
```

Результат преобразования:

```
{... "message": {
  "msg" : "fetching.module.release"
  },
  "pod_labels": {
    "pod_app": "test"
  }
}
```

4.14.4 Удаление лейблов

Трансформация DropLabels позволяет удалить указанные лейблы из структурированного JSON-сообщения.

Перед применением трансформации DropLabels к полю message или его вложенным полям преобразуйте запись лога в структурированный объект с помощью трансформации ParseMessage.

Пример конфигурации с удалением лейбла и предварительной трансформацией ParseMessage:

```
apiVersion: deckhouse.io/v1alpha2
kind: ClusterLogDestination
metadata:
  name: drop-label
spec:
  ...
  transformations:
    - action: ParseMessage
      parseMessage:
```

```
      sourceFormat: JSON
  - action: DropLabels
    dropLabels:
      labels:
        - .message.example
```

Пример изначальной записи в логе:

```
{"msg" : "fetching.module.release", "example": "test"}
```

Результат преобразования:

```
{... "message": {
  "msg" : "fetching.module.release"
  }
}
```

4.14.5 Отладка и расширенные возможности

4.14.5.1 Включение debug-логов агента log-shipper

Чтобы включить debug-логи агента log-shipper на узлах с информацией об HTTP-запросах, переиспользовании подключения, трассировке и прочими данными, включите параметр debug в конфигурации модуля log-shipper.

Пример конфигурации модуля:

```
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: log-shipper
spec:
  version: 1
  enabled: true
  settings:
    debug: true
```

4.14.5.2 Дополнительная информация о каналах передачи логов

Используя команды для Vector, можно получить дополнительную информацию о каналах передачи данных.

Для начала подключитесь к одному из подов log-shipper:

```
d8 k -n d8-log-shipper get pods -o wide | grep $node
```

```
d8 k -n d8-log-shipper exec $pod -it -c vector -- bash
```

Последующие команды выполняйте из командной оболочки пода.

Чтобы получить схему топологии вашей конфигурации в формате DOT, выполните команду:

```
vector graph
```

Используйте WebGraphviz или аналогичный сервис для отрисовки схемы на основе содержимого DOT-файла.

Пример схемы для одного канала передачи логов в формате ASCII:

```
+------------------------------------------------+
|  d8_cluster_source_flant-integration-d8-logs   |
+------------------------------------------------+
  |
  |
  v
+------------------------------------------------+
|       d8_tf_flant-integration-d8-logs_0        |
+------------------------------------------------+
  |
  |
  v
+------------------------------------------------+
|       d8_tf_flant-integration-d8-logs_1        |
+------------------------------------------------+
  |
  |
  v
+------------------------------------------------+
| d8_cluster_sink_flant-integration-loki-storage |
+------------------------------------------------+
```

Чтобы посмотреть объем трафика на каждом этапе обработки логов, используйте команду:

```
vector top
```

Пример вывода команды:

```
Vector TOP output
```

Для просмотра входных данных на разных стадиях обработки логов используйте команду:

```
vector tap
```

Указав в ней ID конкретного этапа обработки, вы сможете увидеть логи которые поступают на этом этапе. Также поддерживаются выборки в формате glob, например, cluster_logging_config/*.

Просмотр логов до применения правил трансформаций (cluster_logging_config/* является первой стадией обработки согласно выводу команды vector graph):

```
vector tap 'cluster_logging_config/*'
```

Изменённые логи, поступающие на вход следующих в цепочке компонентов каналов:

```
vector tap 'transform/*'
```

Для отладки правил на языке Vector Remap Language (VRL) используйте команду:

```
vector vrl
```

Пример VRL-программы:

```
. = {"test1": "lynx", "test2": "fox"}
del(.test2)
```

4.14.5.3 Добавление поддержки нового source или sink

Модуль log-shipper в собирается на основе Vector с ограниченным набором cargo-функций, чтобы минимизировать размер запускаемого файла и ускорить сборку.

Чтобы посмотреть весь список поддерживаемых функций, выполните команду:

```
vector list
```

Если нужный source или sink отсутствует, добавьте соответствующую cargo-функцию в Dockerfile.

4.14.6 Особые случаи

Если в кластере пространства имён размечены с помощью лейблов (например, environment=production), вы можете использовать опцию labelSelector для сбора логов из продуктивных пространств имён.

Пример конфигурации:

```
apiVersion: deckhouse.io/v1alpha1
kind: ClusterLoggingConfig
metadata:
  name: production-logs
spec:
  type: KubernetesPods
  kubernetesPods:
    namespaceSelector:
      labelSelector:
        matchLabels:
          environment: production
  destinationRefs:
  - loki-storage
```

В ПО «Deckhouse Platform» предусмотрен лейбл log-shipper.deckhouse.io/exclude=true для исключения определенных подов и пространств имён. Он помогает остановить сбор логов с подов и пространств имён без изменения глобальной конфигурации.

```
---
apiVersion: v1
kind: Namespace
metadata:
  name: test-namespace
  labels:
    log-shipper.deckhouse.io/exclude: "true"
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test-deployment
spec:
  ...
  template:
    metadata:
      labels:
        log-shipper.deckhouse.io/exclude: "true"
```

4.15 Виртуализация

ПО «Deckhouse Platform» позволяет декларативно создавать, запускать и управлять виртуальными машинами и их ресурсами. Виртуализация осуществляется с помощью модуля virtualization, который обеспечивает запуск и управление виртуальными машинами и их ресурсами. Для управления ресурсами кластера используется утилита командной строки d8. Виртуальная машина запускается внутри пода, для того, чтобы управлять виртуальными машинами как обычными ресурсами Kubernetes и использовать все возможности, включая балансировщики нагрузки, сетевые политики, средства автоматизации и т. д.

4.15.1 Включение модуля virtualization

Работа с модулем предполагает наличие предварительно развёрнутого кластера.

Для включения модуля примените файл конфигурации или воспользуйтесь консолью:

Пример файла конфигурации:

```
d8 k apply -f -<<EOF
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: virtualization
spec:
  enabled: true
  version: 1
  settings:
    dvcr:
      storage:
        persistentVolumeClaim:
          size: 50G
          storageClassName: sds-replicated-thin-r1 #Ваш существующий storageClass
        type: PersistentVolumeClaim
    virtualMachineCIDRs:
      - 10.66.10.0/24 #Актуальная сеть для вашего кластера
EOF
```

Отследить готовность модуля можно с использованием следующей команды:

```
d8 k get modules virtualization
```

Пример вывода:

```
NAME             WEIGHT   SOURCE      PHASE   ENABLED   READY
virtualization   900      deckhouse   Ready   True      True
```

Фаза модуля должна быть Ready.

4.15.2 Настройка хранилища

Для работы модуля virtualization необходимо настроить хранилище, которое используется несколькими компонентами платформы. Оно применяется для работы сервиса DVCR, в котором хранятся образы виртуальных машин, а также для создания и хранения дисков виртуальных машин.

Модулем virtualization поддерживаются следующие хранилища:

- Локальное хранилище на основе LVM;
- Распределённая система хранения Ceph;
- Сетевое файловое хранилище NFS;
- Хранилище данных на основе протокола SCSI;
- Унифицированное хранилище TATLIN.UNIFIED (Yadro).

Настройка этих хранилищ подробно описана в п. 4.4.

4.15.3 Параметры модуля

Конфигурация модуля virtualization задаётся через ресурс ModuleConfig в формате YAML. Пример базовой настройки:

```
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: virtualization
spec:
  enabled: true
  version: 1
  settings:
    dvcr:
      storage:
        persistentVolumeClaim:
          size: 50G
          storageClassName: sds-replicated-thin-r1
        type: PersistentVolumeClaim
    virtualMachineCIDRs:
      - 10.66.10.0/24
```

Управление состоянием модуля возможно осуществлять через поле .spec.enabled. Укажите:

- *true* — чтобы включить модуль;
- *false* — чтобы выключить модуль.

Выключение модуля потребует аннотацию:

```
kubectl annotate moduleconfig virtualization
modules.deckhouse.io/allow-disabling="true" --overwrite
```

Блок .spec.settings.dvcr.storage задаёт параметры постоянного тома для хранения образов (DVCR).

- .spec.settings.dvcr.storage.persistentVolumeClaim.size — размер тома, например 50G. Увеличение значения приводит к расширению хранилища.
- .spec.settings.dvcr.storage.persistentVolumeClaim.storageClassName — класс хранения, например sds-replicated-thin-r1.

Хранилище, соответствующее указанному классу хранения, должно быть доступно на узлах, где запускается DVCR. Используются system-узлы или worker-узлы при отсутствии system-узлов.

Блок .spec.settings.virtualMachineCIDRs содержит перечень подсетей в формате CIDR. Из указанных диапазонов виртуальным машинам выделяются IP-адреса автоматически либо по запросу.

Ограничения:

- первый и последний адреса каждой подсети зарезервированы;
- подсети блока .spec.settings.virtualMachineCIDRs не должны пересекаться с подсетями узлов кластера, подсетью сервисов и подсетью podCIDR;
- удаление подсети, если из неё выданы адреса виртуальным машинам, запрещено.

Параметр .spec.settings.virtualImages определяет допустимые классы хранения для объектов VirtualImage. Пример:

```
spec:
  settings:
    virtualImages:
      allowedStorageClassNames:
        - sc-1
        - sc-2
```

```
    defaultStorageClassName: sc-1
```

Здесь:

- allowedStorageClassNames (опционально) — список допустимых StorageClass;
- defaultStorageClassName (опционально) — StorageClass, используемый по умолчанию при создании VirtualImage, если в спецификации не указан .spec.persistentVolumeClaim.storageClassName.

Параметр .spec.settings.virtualDisks определяет допустимые классы хранения для объектов VirtualDisk. Пример:

```
spec:
  settings:
    virtualDisks:
      allowedStorageClassNames:
        - sc-1
        - sc-2
      defaultStorageClassName: sc-1
```

Здесь:

- allowedStorageClassNames (опционально) — список допустимых StorageClass;
- defaultStorageClassName (опционально) — StorageClass, используемый по умолчанию при создании VirtualDisk, если в спецификации не указан .spec.persistentVolumeClaim.storageClassName.

4.15.4 Образы

Ресурс ClusterVirtualImage служит для загрузки образов виртуальных машин во внутрикластерное хранилище, после чего с его помощью можно создавать диски виртуальных машин. Он доступен во всех пространствах имен и проектах кластера.

Процесс создания образа включает следующие шаги:

- Пользователь создаёт ресурс ClusterVirtualImage.
- После создания образ автоматически загружается из указанного в спецификации источника в хранилище (DVCR).
- После завершения загрузки ресурс становится доступным для создания дисков.

Существуют различные типы образов:

ISO-образ — установочный образ, используемый для начальной установки операционной системы (ОС). Такие образы выпускаются производителями ОС и используются для установки на физические и виртуальные серверы.

Образ диска с предустановленной системой — содержит уже установленную и настроенную операционную систему, готовую к использованию после создания виртуальной машины. Готовые образы можно получить на ресурсах разработчиков дистрибутива, либо создать самостоятельно.

Поддерживаются следующие форматы образов с предустановленной системой:

- qcow2;
- raw;
- vmdk;
- vdi.

Образы могут быть сжаты одним из следующих алгоритмов сжатия: gz, xz.

После создания ресурса ClusterVirtualImage тип и размер образа определяются автоматически, и эта информация отражается в статусе ресурса.

Образы могут быть загружены из различных источников, таких как HTTP-серверы, где расположены файлы образов, или контейнерные реестры. Также доступна возможность загрузки образов напрямую из командной строки с использованием утилиты curl. Образы могут быть созданы из других образов и дисков виртуальных машин.

Рассмотрим вариант создания кластерного образа с HTTP-сервера.

Чтобы создать ресурс ClusterVirtualImage, выполните следующую команду (укажите URL образа):

```
d8 k apply -f - <<EOF
apiVersion: virtualization.deckhouse.io/v1alpha2
kind: ClusterVirtualImage
metadata:
  name: myimage-pvc
spec:
  # Источник для создания образа.
  dataSource:
    type: HTTP
    http:
      url: <IMAGE_URL>
EOF
```

Проверьте результат создания ресурса ClusterVirtualImage, выполнив следующую команду:

```
d8 k get clustervirtualimage myimage-pvc
```

Короткий вариант команды:

```
d8 k get cvi myimage-pvc
```

В результате будет выведена информация о ресурсе:

```
NAME           PHASE   CDROM   PROGRESS   AGE
myimage-pvc    Ready   false   100%       23h
```

После создания ресурс ClusterVirtualImage может находиться в одном из следующих состояний (фаз):

- *Pending* — ожидание готовности всех зависимых ресурсов, требующихся для создания образа;
- *WaitForUserUpload* — ожидание загрузки образа пользователем (фаза присутствует только для type=Upload);
- *Provisioning* — идёт процесс создания образа;
- *Ready* — образ создан и готов для использования;
- *Failed* — произошла ошибка в процессе создания образа;
- *Terminating* — идёт процесс удаления образа. Образ может «зависнуть» в данном состоянии, если ещё подключен к виртуальной машине.

До тех пор, пока образ не перешёл в фазу Ready, содержимое всего блока .spec допускается изменять. При изменении процесс создании диска запустится заново. После перехода в фазу Ready содержимое блока .spec менять нельзя.

Диагностика проблем с ресурсом осуществляется путем анализа информации в блоке .status.conditions.

Чтобы отследить процесс создания образа, добавьте ключ -w к команде проверки результата создания ресурса:

```
d8 k get cvi myimage-pvc -w
```

Пример вывода:

```
NAME           PHASE          CDROM   PROGRESS   AGE
myimage-pvc    Provisioning   false              4s
myimage-pvc    Provisioning   false   0.0%       4s
myimage-pvc    Provisioning   false   28.2%      6s
myimage-pvc    Provisioning   false   66.5%      8s
myimage-pvc    Provisioning   false   100.0%     10s
myimage-pvc    Provisioning   false   100.0%     16s
myimage-pvc    Ready          false   100%       18s
```

В описании ресурса ClusterVirtualImage можно получить дополнительную информацию о скачанном образе. Для этого выполните следующую команду:

```
d8 k describe cvi myimage-pvc
```

Образ, хранящийся в реестре контейнеров, имеет определённый формат. Рассмотрим создание такого образа:

Для начала подготовьте образ виртуальной машины.

В закрытом контуре доступ к внешним ресурсам, как правило, отсутствует, поэтому образ необходимо предварительно загрузить и разместить в доступном локальном хранилище.

При наличии доступа к сети образ можно скачать по URL:

```
curl -L <IMAGE_URL> -o myimage-pvc.img
```

Далее на **отдельной** виртуальной машине, не входящей в кластер Deckhouse Platform, создайте Dockerfile со следующим содержимым:

```
FROM scratch
COPY myimage-pvc.img /disk/myimage-pvc.img
```

Соберите контейнерный образ и загрузите его в **заранее подготовленный реестр контейнеров**, доступный из кластера ПО «Deckhouse Platform». В примере ниже используется публичный реестр docker.io. Для выполнения команд требуется учётная запись в выбранном реестре и настроенное окружение сборки.

```
docker build -t docker.io/<username>/myimage-pvc:latest
```

где <username> — имя пользователя, указанное при регистрации в docker.io.

Загрузите созданный образ в реестр контейнеров:

```
docker push docker.io/<username>/myimage-pvc:latest
```

Чтобы использовать этот образ, создайте в качестве примера ресурс:

```
d8 k apply -f - <<EOF
apiVersion: virtualization.deckhouse.io/v1alpha2
kind: ClusterVirtualImage
metadata:
  name: myimage-pvc
spec:
  dataSource:
    type: ContainerImage
    containerImage:
      image: docker.io/<username>/myimage-pvc:latest
EOF
```

Чтобы загрузить образ из командной строки, предварительно создайте следующий ресурс, как представлено ниже на примере ClusterVirtualImage:

```
d8 k apply -f - <<EOF
apiVersion: virtualization.deckhouse.io/v1alpha2
kind: ClusterVirtualImage
metadata:
  name: some-image
spec:
  # Настройки источника образа.
  dataSource:
    type: Upload
EOF
```

После создания ресурс перейдёт в фазу WaitForUserUpload, что говорит о готовности к загрузке образа.

Доступно два варианта загрузки — с узла кластера и с произвольного узла за пределами кластера:

```
d8 k get cvi some-image -o jsonpath="{.status.imageUploadURLs}"  | jq
```

Пример вывода:

```
{

"external":"https://virtualization.example.com/upload/g2OuLgRhdAWqlJsCMyNvcdt4o5ERIwmm",
  "inCluster":"http://10.222.165.239/upload"
```

```
}
```

Здесь:

- *inCluster* — URL-адрес, который используется, если необходимо выполнить загрузку образа с одного из узлов кластера;
- *external* — URL-адрес, который используется во всех остальных случаях.

В качестве примера загрузите образ Cirros:

```
curl -L http://download.cirros-cloud.net/0.5.1/cirros-0.5.1-x86_64-disk.img -o
cirros.img
```

Выполните загрузку образа с использованием следующей команды:

```
curl https://virtualization.example.com/upload/g2OuLgRhdAWqlJsCMyNvcdt4o5ERIwmm
--progress-bar -T cirros.img | cat
```

После завершения загрузки образ должен быть создан и переведён в фазу Ready. Чтобы проверить это, выполните следующую команду:

```
d8 k get cvi some-image
```

Пример вывода:

```
NAME          PHASE   CDROM   PROGRESS   AGE
some-image    Ready   false   100%       1m
```

4.15.5 Классы виртуальных машин

Ресурс VirtualMachineClass предназначен для централизованной конфигурации предпочтительных параметров виртуальных машин. Он позволяет определять инструкции CPU, политики конфигурации ресурсов CPU и памяти для виртуальных машин, а также определять соотношения этих ресурсов. Помимо этого, VirtualMachineClass обеспечивает управление размещением виртуальных машин по узлам платформы. Это позволяет администраторам эффективно управлять ресурсами платформы виртуализации и оптимально размещать виртуальные машины на узлах платформы.

По умолчанию автоматически создается один ресурс VirtualMachineClass с типом generic, который представляет универсальную модель CPU, использующую достаточно старую, но

поддерживаемую большинством современных процессоров модель Nehalem. Это позволяет запускать ВМ на любых узлах кластера с возможностью «живой» миграции.

Рекомендуется создать как минимум один ресурс VirtualMachineClass в кластере с типом Discovery сразу после того, как все узлы будут настроены и добавлены в кластер. Это позволит использовать в виртуальных машинах универсальный процессор с максимально возможными характеристиками с учетом CPU на узлах кластера, что позволит виртуальным машинам использовать максимум возможностей CPU и при необходимости беспрепятственно осуществлять миграцию между узлами кластера.

Чтобы вывести список ресурсов VirtualMachineClass, выполните следующую команду:

```
d8 k get virtualmachineclass
```

Пример вывода:

```
NAME                PHASE   AGE
generic             Ready   6d1h
```

Обязательно указывайте ресурс VirtualMachineClass в конфигурации виртуальной машины. Пример указания класса в спецификации ВМ:

```
apiVersion: virtualization.deckhouse.io/v1alpha2
kind: VirtualMachine
metadata:
  name: linux-vm
spec:
  virtualMachineClassName: generic # Название ресурса VirtualMachineClass.
  ...
```

Структура ресурса VirtualMachineClass выглядит следующим образом:

```
apiVersion: virtualization.deckhouse.io/v1alpha2
kind: VirtualMachineClass
metadata:
  name: <vmclass-name>
spec:
  # Блок описывает параметры виртуального процессора для виртуальных машин.
  # Изменять данный блок нельзя после создания ресурса.
  cpu: ...

  # (опциональный блок) Описывает правила размещения виртуальных машины по узлам.
  # При изменении автоматически применяется ко всем виртуальных машинам,
использующим данный VirtualMachineClass.
  nodeSelector: ...

  # (опциональный блок) Описывает политику настройки ресурсов виртуальных машин.
```

```
  # При изменении автоматически применяется ко всем виртуальных машинам,
использующим данный VirtualMachineClass.
  sizingPolicies: ...
```

Далее рассмотрим настройки блоков более детально.

Блок .spec.cpu позволяет задать или настроить vCPU для ВМ.

Настройки блока .spec.cpu после создания ресурса VirtualMachineClass изменять нельзя.

Примеры настройки блока .spec.cpu:

Класс с vCPU с требуемым набором процессорных инструкций. Для этого используйте type: Features, чтобы задать необходимый набор поддерживаемых инструкций для процессора:

```
spec:
  cpu:
    features:
      - vmx
    type: Features
```

Класс с универсальным vCPU для заданного набора узлов. Для этого используйте type: Discovery:

```
spec:
  cpu:
    discovery:
      nodeSelector:
        matchExpressions:
          - key: node-role.kubernetes.io/control-plane
            operator: DoesNotExist
    type: Discovery
```

Класс с type: Host использует виртуальный vCPU, максимально соответствующий набору инструкций vCPU узла платформы, что обеспечивает высокую производительность и функциональность. Он также гарантирует совместимость с живой миграцией для узлов с похожими типами процессоров. Например, миграция виртуальной машины между узлами с процессорами Intel и AMD невозможна. Это также относится к процессорам разных поколений, так как их наборы инструкций могут отличаться.

```
spec:
  cpu:
    type: Host
```

Класс с type: HostPassthrough использует физический CPU узла платформы без изменений. Виртуальная машина, использующая этот класс, может быть мигрирована только на узел, у которого CPU точно совпадает с CPU исходного узла.

```
spec:
  cpu:
    type: HostPassthrough
```

Чтобы создать vCPU конкретного процессора с предварительно определённым набором инструкций, используйте тип type: Model. Предварительно, чтобы получить перечень названий поддерживаемых CPU для узла кластера, выполните команду:

```
d8 k get nodes <node-name> -o json | jq '.metadata.labels | to_entries[] |
select(.key | test("cpu-model.node.virtualization.deckhouse.io")) | .key |
split("/")[1]' -r
```

Пример вывода:

```
Broadwell-noTSX
Broadwell-noTSX-IBRS
Haswell-noTSX
Haswell-noTSX-IBRS
IvyBridge
IvyBridge-IBRS
Nehalem
Nehalem-IBRS
Penryn
SandyBridge
SandyBridge-IBRS
Skylake-Client-noTSX-IBRS
Westmere
Westmere-IBRS
```

Далее укажите в спецификации ресурса VirtualMachineClass следующее:

```
spec:
  cpu:
    model: IvyBridge
    type: Model
```

Блок .spec.nodeSelector опционален. Он позволяет задать узлы, на которых будут размещаться ВМ, использующие данный vmclass:

```
spec:
    nodeSelector:
      matchExpressions:
```

```
      - key: node.deckhouse.io/group
        operator: In
        values:
        - green
```

Блок .spec.sizingPolicy позволяет задать политики сайзинга ресурсов виртуальных машин, которые используют vmclass.

Изменения в блоке .spec.sizingPolicy также могут повлиять на виртуальные машины. Для виртуальных машин, чья политика сайзинга не будет соответствовать новым требованиям политики, условие SizingPolicyMatched в блоке .status.conditions будет ложным (status: False).

При настройке sizingPolicy будьте внимательны и учитывайте топологию CPU для виртуальных машин.

Блок cores обязательный и задает диапазоны ядер, на которые распространяется правило, описанное в этом же блоке.

Диапазоны [min; max] для параметра cores должны быть строго последовательными и непересекающимися.

Правильная структура (диапазоны идут друг за другом без пересечений):

```
- cores:
    min: 1
    max: 4
    ...
- cores:
    min: 5   # Начало следующего диапазона = (предыдущий max + 1)
    max: 8
```

Недопустимый вариант (пересечение значений):

```
- cores:
    min: 1
    max: 4
    ...
- cores:
    min: 4   # Ошибка: Значение 4 уже входит в предыдущий диапазон
    max: 8
```

Правило: Каждый новый диапазон должен начинаться со значения, непосредственно следующего за max предыдущего диапазона.

Для каждого диапазона ядер cores можно задать дополнительные требования:

- Память (memory) — указывается:
- Либо минимум и максимум памяти для всех ядер в диапазоне,
- Либо минимум и максимум памяти на одно ядро (memoryPerCore).
- Допустимые доли ядер (coreFractions) — список разрешенных значений (например, [25, 50, 100] для 25%, 50% или 100% использования ядра).

Важно. Для каждого диапазона cores обязательно укажите: либо memory (или memoryPerCore), либо coreFractions, либо оба параметра одновременно.

Пример политики с подобными настройками:

```
spec:
  sizingPolicies:
    # Для диапазона от 1 до 4 ядер возможно использовать от 1 до 8 ГБ оперативной
памяти с шагом 512Mi,
    # т.е 1 ГБ, 1,5 ГБ, 2 ГБ, 2,5 ГБ и т. д.
    # Запрещено использовать выделенные ядра.
    # Доступны все варианты параметра `corefraction`.
    - cores:
        min: 1
        max: 4
      memory:
        min: 1Gi
        max: 8Gi
        step: 512Mi
      coreFractions: [5, 10, 20, 50, 100]
    # Для диапазона от 5 до 8 ядер возможно использовать от 5 до 16 ГБ оперативной
памяти с шагом 1 ГБ,
    # т.е. 5 ГБ, 6 ГБ, 7 ГБ и т. д.
    # Запрещено использовать выделенные ядра.
    # Доступны некоторые варианты параметра `corefraction`.
    - cores:
        min: 5
        max: 8
      memory:
        min: 5Gi
        max: 16Gi
        step: 1Gi
      coreFractions: [20, 50, 100]
    # Для диапазона от 9 до 16 ядер возможно использовать от 9 до 32 ГБ оперативной
памяти с шагом 1 ГБ.
    # При необходимости можно использовать выделенные ядра.
    # Доступны некоторые варианты параметра `corefraction`.
    - cores:
        min: 9
        max: 16
      memory:
        min: 9Gi
        max: 32Gi
        step: 1Gi
      coreFractions: [50, 100]
    # Для диапазона от 17 до 248 ядер возможно использовать от 1 до 2 ГБ оперативной
памяти из расчёта на одно ядро.
```

```
    # Доступны для использования только выделенные ядра.
    # Единственный доступный параметр `corefraction` = 100%.
    - cores:
        min: 17
        max: 248
      memory:
        perCore:
          min: 1Gi
          max: 2Gi
      coreFractions: [100]
```

Пример конфигурации VirtualMachineClass.

Представим, что у нас есть кластер из четырех узлов. Два из этих узлов с лейблом group=blue оснащены процессором «CPU X» с тремя наборами инструкций, а остальные два узла с лейблом group=green имеют более новый процессор «CPU Y» с четырьмя наборами инструкций.

Для оптимального использования ресурсов данного кластера рекомендуется создать три дополнительных класса виртуальных машин (VirtualMachineClass):

- *universal* — этот класс позволит виртуальным машинам запускаться на всех узлах платформы и мигрировать между ними. При этом будет использоваться набор инструкций для самой младшей модели CPU, что обеспечит наибольшую совместимость;
- *cpuX* — этот класс будет предназначен для виртуальных машин, которые должны запускаться только на узлах с процессором «CPU X». ВМ смогут мигрировать между этими узлами, используя доступные наборы инструкций «CPU X»;
- *cpuY* — этот класс предназначен для виртуальных машин, которые должны запускаться только на узлах с процессором «CPU Y». ВМ смогут мигрировать между этими узлами, используя доступные наборы инструкций «CPU Y».

Набор инструкций для процессора — это набор всех команд, которые процессор может выполнять, таких как сложение, вычитание или работа с памятью. Они определяют, какие операции возможны, влияют на совместимость программ и производительность, а также могут меняться от одного поколения процессоров к другому.

Примерные конфигурации ресурсов для данного кластера:

```
---
apiVersion: virtualization.deckhouse.io/v1alpha2
kind: VirtualMachineClass
metadata:
```

```
  name: universal
spec:
  cpu:
    discovery: {}
    type: Discovery
  sizingPolicies: { ... }
---
apiVersion: virtualization.deckhouse.io/v1alpha2
kind: VirtualMachineClass
metadata:
  name: cpuX
spec:
  cpu:
    discovery: {}
    type: Discovery
  nodeSelector:
    matchExpressions:
      - key: group
        operator: In
        values: ["blue"]
  sizingPolicies: { ... }
---
apiVersion: virtualization.deckhouse.io/v1alpha2
kind: VirtualMachineClass
metadata:
  name: cpuY
spec:
  cpu:
    discovery:
   nodeSelector:
        matchExpressions:
          - key: group
            operator: In
            values: ["green"]
    type: Discovery
  sizingPolicies: { ... }
```

4.15.6 Механизмы обеспечения надежности

Для обеспечения надежности в ПО «Deckhouse Platform» предусмотрены механизмы:

- перебалансировка ВМ;
- миграция и режим обслуживания;
- ColdStandby.

Платформа предоставляет возможность автоматизировать управление размещением уже запущенных виртуальных машин в кластере. Для активации этой функции необходимо включить модуль descheduler.

После включения модуля система самостоятельно следит за оптимальной работой виртуальных машин в кластере. Основные возможности модуля:

Балансировка нагрузки — система анализирует резервирование процессора на узлах кластера. Если на узле зарезервировано более 80% процессора, система автоматически переносит часть ВМ на менее загруженные узлы. Это предотвращает перегрузку и обеспечивает стабильную работу ВМ.

Подходящее размещение — система проверяет, соответствует ли текущий узел требованиям каждой ВМ, соблюдены ли правила размещения по отношению к узлу или другим ВМ кластера. Например, если ВМ не должна находиться на одном узле с другой ВМ, модуль переносит её на более подходящий узел.

Миграция виртуальных машин является важной функцией в управлении виртуализированной инфраструктурой. Она позволяет перемещать работающие виртуальные машины с одного физического узла на другой без их отключения. Миграция виртуальных машин необходима для ряда задач и сценариев:

Балансировка нагрузки — перемещение виртуальных машин между узлами позволяет равномерно распределять нагрузку на серверы, обеспечивая использование ресурсов наилучшим образом.

Перевод узла в режим обслуживания — виртуальные машины могут быть перемещены с узлов, которые нужно вывести из эксплуатации для выполнения планового обслуживания или обновления программного обеспечения.

Обновление «прошивки» виртуальных машин — миграция позволяет обновить «прошивку» виртуальных машин, не прерывая их работу.

Далее будет рассмотрен пример миграции выбранной виртуальной машины.

Перед запуском миграции проверьте текущий статус виртуальной машины:

```
d8 k get vm -w
```

Пример вывода:

```
NAME                          PHASE      NODE           IPADDRESS      AGE
linux-vm                      Running    virtlab-pt-1   10.66.10.14    79m
```

В выводе отображено, что на данный момент ВМ запущена на узле virtlab-pt-1.

Для осуществления миграции виртуальной машины с одного узла на другой, с учетом требований к размещению виртуальной машины используется ресурс VirtualMachineOperations (vmop) с типом Evict. Создайте данный ресурс, следуя примеру:

```
d8 k create -f - <<EOF
apiVersion: virtualization.deckhouse.io/v1alpha2
kind: VirtualMachineOperation
metadata:
  generateName: evict-linux-vm-
spec:
  # Имя виртуальной машины.
  virtualMachineName: linux-vm
  # Операция для миграции.
  type: Evict
EOF
```

Сразу после создания ресурса vmop выполните следующую команду:

```
d8 k get vm -w
```

Пример вывода:

```
NAME                          PHASE       NODE            IPADDRESS     AGE
linux-vm                      Running     virtlab-pt-1    10.66.10.14   79m
linux-vm                      Migrating   virtlab-pt-1    10.66.10.14   79m
linux-vm                      Migrating   virtlab-pt-1    10.66.10.14   79m
linux-vm                      Running     virtlab-pt-2    10.66.10.14   79m
```

Если необходимо прервать миграцию, удалите соответствующий ресурс vmop, пока он находится в фазе Pending или InProgress.

При выполнении работ на узлах с запущенными виртуальными машинами существует риск нарушения их работоспособности. Чтобы этого избежать, узел можно перевести в режим обслуживания и мигрировать виртуальные машины на другие свободные узлы.

Для этого выполните следующую команду:

```
d8 k drain <nodename> --ignore-daemonsets --delete-emptydir-data
```

где <nodename> — узел, на котором предполагается выполнить работы и который должен быть освобождён от всех ресурсов (в том числе от системных).

Если необходимо вытеснить с узла только виртуальные машины, выполните следующую команду:

```
d8 k drain <nodename> --pod-selector
vm.kubevirt.internal.virtualization.deckhouse.io/name --delete-emptydir-data
```

После выполнения команды d8 k drain узел перейдёт в режим обслуживания, и виртуальные машины на нём запускаться не смогут.

Чтобы вывести его из режима обслуживания, остановите выполнение команды drain (Ctrl+C), затем выполните:

```
d8 k uncordon <nodename>
```

ColdStandby обеспечивает механизм восстановления работы виртуальной машины после сбоя на узле, на котором она была запущена.

Для работы данного механизма необходимо выполнить следующие требования:

- для политики запуска виртуальной машины (.spec.runPolicy) должно быть установлено одно из следующих значений: AlwaysOnUnlessStoppedManually, AlwaysOn;
- на узлах, где запущены виртуальные машины, должен быть включён механизм Fencing.

Рассмотрим, как это работает на примере:

- Кластер состоит из трех узлов: master, workerA и workerB. На worker-узлах включён механизм Fencing. Виртуальная машина linux-vm запущена на узле workerA.
- На узле workerA возникает проблема (выключилось питание, пропала сеть, и т. д.).
- Контроллер проверяет доступность узлов и обнаруживает, что workerA недоступен.
- Контроллер удаляет узел workerA из кластера.
- Виртуальная машина linux-vm запускается на другом подходящем узле (workerB).

4.15.7 Контроль целостности в модуле virtualization

Контроль целостности в модуле virtualization осуществляется согласно общим правилам, применяемым в ПО «Deckhouse Platform» и описанным в п. 5.7.

4.16 Миграция данных etcd

Начиная с версии 1.73 ПО «Deckhouse Platform» добавлен функционал контроля целостности объектов, хранимых в базе данных etcd . Для этого изменен формат данных (подробная информация указана в разделе 5.7.3).

После обновления платформы с более ранней версии, а также в случае развертывания кластера с режимом контроля подписи *Rollback* (стандартное значение), требуется осуществить процедуру миграции данных.

Для осуществления миграции требуется:

1. Переключить режим работы контроля целостности на *Migrate*.

Режим работы контроля целостности указывается параметром apiserver.signature в параметрах модуля control-plane-manager (объект ModuleConfig control-plane-manager).

Пример манифеста ModuleConfig control-plane-manager с указанием режима работы:

```
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: control-plane-manager
spec:
  settings:
    apiserver:
      signature: Migrate
```

2. Дождаться очистки очереди ПО «Deckhouse Platform».

   В процессе переключения режима работы контроля целостности будет осуществлен перезапуск подов apiserver, в связи с чем возможна временная недоступность api.

   Проверить состояние очереди можно, выполнив команду на master-узле:

```
d8 system queue list
```

3. Осуществить миграцию данных.

Преобразование формата хранимых данных происходит при их изменении. Таким образом процедура миграции сводится к принудительной модификации всех объектов kubernetes, которые хранятся в etcd.

Миграция производится с помощью утилиты `d8`. Для этого в утилиту добавлена команда `sig-migrate`, расположенная в разделе `tools`.

Команда `sig-migrate` формирует список всех ресурсов Kubernetes, а также поочередно добавляет к ним аннотацию `d8-migration=<timestamp>`, и затем удаляет аннотации с префиксом `d8-migration-`.

Запустите утилиту от пользователя root с master-узла кластера со следующими аргументами:

```
d8 tools sig-migrate
```

Внимание! *Процедура миграции выполняется продолжительное время. В случае сбоя связности или отключения ssh-сессии процедура будет прервана. Рекомендуется запуск в эмуляторе терминала по типу screen или tmux.*

Возможен запуск процедуры миграции с другого узла, у которого есть доступ к api кластера, путем изменения флагов запуска запуска утилиты.

Команда `sig-migrate` имеет следующие флаги запуска:

Флаг	Описание	Значение по умолчанию
`--retry`	Выполнить установку аннотаций только для объектов, которые не удалось обработать в предыдущем запуске	false
`--as`	Указать Kubernetes service account для выполнения операций kubectl (impersonation)	system:serviceaccount:d8-system:deckhouse
`--log-level`	Уровень логирования (INFO, DEBUG, TRACE)	DEBUG
`--kubeconfig`	Путь к файлу kubeconfig для CLI запросов	$HOME/.kube/config или $KUBECONFIG
`--context`	Имя контекста kubeconfig для использования	kubernetes-admin@kubernetes

По окончании выполнения, если на какие-либо объекты не удалось установить аннотацию, команда автоматически выведет предупреждающее сообщение со следующей информацией:

- Количество объектов, для которых не удалось произвести миграцию
- Пути к файлам с логами ошибок
- Инструкции по расследованию и запуску повторной попытки

Пример вывода при наличии ошибок:

```
⚠  Migration completed with 5 failed object(s).

Some objects could not be annotated. Please check the error details:
  Error log file: /tmp/failed_errors.txt
  Failed objects list: /tmp/failed_annotations.txt

To investigate the issues:
  1. Review the error log file to understand why objects failed
  2. Check permissions and resource availability
  3. Retry migration for failed objects only using:
     d8 tools sig-migrate --retry
```

Для повторной установки аннотаций только на объекты, для которых не удалось произвести миграцию, используйте флаг `--retry`

```
d8 tools sig-migrate --retry
```

Повторять процедуру миграции следует до тех пор, пока утилита не сообщит, что больше нет объектов, для которых не удалось установить аннотацию, а также до прекращения алерта D8SignatureErrorsDetected.

5 Описание параметров (настроек) безопасности

5.1 Настройка сканирования на уязвимости

Сканирование на уязвимости осуществляется с помощью модуля operator-trivy. Модуль позволяет получать информацию о проблемах в настройке кластера или отдельных объектов кластера, а также информацию об уязвимостях, найденных в используемых образах контейнеров.

Чтобы получать информацию о наличии уязвимостей, необходимо:

- включить модуль сканирования;
- обновить базы уязвимостей (см. п. 5.1.1)
- установить на пространствах имен, содержащих подлежащие сканированию поды, аннотацию security-scanning.deckhouse.io/enabled.

Если модуль operator-trivy ранее был отключён, то для его включения выполните следующую команду (требуется файл конфигурации подключения к кластеру (kubeconfig) и установленная утилита d8 (поставляется в составе ПО «Deckhouse Platform»)):

```
d8 system module enable operator-trivy
```

Для установки аннотации security-scanning.deckhouse.io/enabled на пространство имен, содержащее поды, подлежащие сканированию на уязвимости, выполните, например, следующую команду (требуется файл конфигурации подключения к кластеру (kubeconfig) и установленная утилита d8 (поставляется в составе ПО «Deckhouse Platform»)):

```
d8 k label namespace <ПРОСТРАНСТВО_ИМЕН> security-scanning.deckhouse.io/enabled=""
```

Информация о найденных уязвимостях обновляется после включения модуля operator-trivy или установки аннотации на пространство имен и далее каждые 6 часов.

Просмотр отчетов о сканировании осуществляется в веб-интерфейсе Grafana в дашбордах CIS Kubernetes Benchmark и Trivy Image Vulnerability Overview, сгруппированных в папке Security (см.п.5.5):

5.1.1 Обновление базы уязвимостей

В составе поставки ПО «Deckhouse Platform» входит база уязвимостей, которая не является актуальной. Для получения актуальной базы уязвимостей необходимо выполнять ее периодическое обновление. Рекомендуемая частота обновления – один раз в сутки.

При сканировании автоматически определяется дистрибутив ОС и применяются соответствующие данные из баз уязвимостей, предоставляемых DKP CSE. В DKP CSE используются следующие источники, для наполнения базы уязвимостей:

- Банк данных угроз безопасности информации ФСТЭК России (БДУ ФСТЭК);
- AlmaLinux Errata;
- Alpine SecDB;
- ALTRepo Errata OVAL;
- Amazon Linux Security Advisories;
- Arch Linux Security Tracker;
- Debian Security Tracker;
- GitHub Security Advisory Database;
- National Vulnerability Database (NVD);
- Oracle OVAL;
- Photon Security Advisory;
- Red Hat OVAL;
- RED SOFT OVAL;
- Rocky Linux UpdateInfo;
- SUSE Security CVRF;
- Ubuntu CVE Tracker;
- Wolfi SecDB.

Первичным источником базы уязвимостей является хранилище образов контейнеров по адресу registry-cse.deckhouse.ru. Обновление баз уязвимостей подразумевает копирование образов контейнеров с базой уязвимостей из первичного источника в промежуточный файл, а затем из промежуточного файла в хранилище образов контейнеров, используемое для работы ПО «Deckhouse Platform».

Для обновления базы уязвимостей, необходим лицензионный ключ (входит в поставку ПО «Deckhouse Platform»).

Для скачивания обновлений базы уязвимостей в файл, выполните следующую команду на узле, с которого производили установку или с выделенного хоста для обновления базы (укажите имя файла и лицензионный ключ):

```
d8 mirror pull \
  ${PACKAGE_DIR_PATH} \
  --source="registry-cse.deckhouse.ru/deckhouse/cse" \
  --license="${LICENSE_KEY}" \
  --gost-digest \
  --deckhouse-tag="v1.73" \
  --no-modules \
  --no-platform
```

Для загрузки обновлений базы уязвимостей из файла, выполните следующую команду на мастер-узле (укажите имя файла с базой обновлений и данные хранилища образа контейнеров):

```
d8 mirror push <PACKAGE_DIR_PATH> <LOCAL_REGISTRY_URL> \
  --registry-login <LOCAL_REGISTRY_LOGIN> \
  --registry-password <LOCAL_REGISTRY_PASSWORD>
```

После загрузки базы уязвимостей в хранилище образов контейнеров модуль operator-trivy обновит внутреннюю базу данных об уязвимостях (раз в 4 часа), и будет использовать обновленные данные при сканировании.

5.2 Настройка политик безопасности

В ПО «Deckhouse Platform» предусмотрено использование следующих политик безопасности:

- *Privileged* — не ограничивающая политика с максимально широким уровнем разрешений;
- *Baseline* — минимально ограничивающая политика, которая предотвращает наиболее известные и популярные способы повышения привилегий. Позволяет использовать стандартную (минимально заданную) конфигурацию пода;
- *Restricted* — политика со значительными ограничениями. Предъявляет самые жёсткие требования к подам.

Политика может быть определена для использования по умолчанию, а также для конкретных пространств имен.

При установке ПО «Deckhouse Platform» политика по умолчанию установлена как Baseline.

Политика по умолчанию определяется в параметре podSecurityStandards.defaultPolicy в ModuleConfig admission-policy-engine (см. п. 5.2.1).

При необходимости изменить политику на конкретном пространстве имен, на него нужно установить лейбл security.deckhouse.io/pod-policy=<НАЗВАНИЕ_ПОЛИТИКИ_В_НИЖНЕМ_РЕГИСТРЕ>.

Пример команды установки политики Restricted на пространство имен my-namespace (должна быть настроена конфигурация подключения к кластеру (kubeconfig), установленная утилита d8 (поставляется в составе ПО «Deckhouse Platform»)):

```
d8 k label ns my-namespace security.deckhouse.io/pod-policy=restricted
```

Действие (режим работы политик), которое будет выполнено по результатам проверки ограничений политики, может стать одним из следующих:

- *Deny* — запрет действия;
- *Dryrun* — отсутствие действия. Применяется при отладке. Информацию о событии можно посмотреть в Grafana или консоли с помощью d8 k;
- *Warn* — аналогично Dryrun, но дополнительно к информации о событии будет выведена информация о том, из-за какого ограничения (constraint) был бы запрет действия, если бы вместо Warn использовался Deny.

По умолчанию, политики применяются в режиме Deny. В этом режиме поды приложений, не удовлетворяющие политикам, не могут быть запущены. Режим работы политик может быть задан как глобально для кластера, так и для каждого пространства имен отдельно.

Глобальный режим работы политик определяется в параметре podSecurityStandards. enforcementAction в ModuleConfig admission-policy-engine (см. п. 5.2.1). В случае если необходимо переопределить глобальный режим политик для конкретного пространства имен, на него нужно установить лейбл security.deckhouse.io/pod-policy-action =<РЕЖИМ_ПОЛИТИКИ_В_НИЖНЕМ_РЕГИСТРЕ> на соответствующем namespace. Список допустимых режимом политик состоит из: Dryrun, Warn, Deny.

Пример команды установки режима *Warn* на пространство имен my-namespace (требуется файл конфигурации подключения к кластеру (kubeconfig) и установленная утилита d8 (поставляется в составе ПО «Deckhouse Platform»)):

```
d8 k label ns my-namespace security.deckhouse.io/pod-policy-action=warn
```

5.2.1 Ресурс ModuleConfig

Пример ресурса ModuleConfig/admission-policy-engine для настройки модуля:

```
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: admission-policy-engine
spec:
  version: 1
  enabled: true
  settings
```

Параметры:

- settings.denyVulnerableImages *объект*

Настройки trivy-провайдера.

Trivy-провайдер запрещает создание Pod/Deployment/StatefulSet/DaemonSet с образами, которые имеют уязвимости в пространствах имен с лейблом security.deckhouse.io/trivy-provider: "".

- settings.denyVulnerableImages.enabled *булевый*

Включить trivy-провайдер.

По умолчанию: false

- settings.denyVulnerableImages.registrySecrets *массив объектов*

Список дополнительных секретов приватных регистри.
По умолчанию для загрузки образов для сканирования используется секрет deckhouse-registry.

По умолчанию: []

- settings.denyVulnerableImages.registrySecrets.name *строка*

ОБЯЗАТЕЛЬНЫЙ ПАРАМЕТР

- denyVulnerableImages.registrySecrets.namespace *строка*

ОБЯЗАТЕЛЬНЫЙ ПАРАМЕТР

- podSecurityStandards *объект*

Настройки политик Pod Security Standards (PSS).

- podSecurityStandards.defaultPolicy *строка*

Определяет политику Pod Security Standards по умолчанию для всех несистемных пространств имен:

- *Privileged* — политика без ограничений. Данная политика допускает эскалацию привилегий;
- *Baseline* — политика с минимальными ограничениями, ограничивающая использование эскалаций привилегий;
- *Restricted* — политика с максимальными ограничениями, соответствующая актуальным рекомендациям по безопасному запуску приложений в кластере.

- podSecurityStandards.enforcementAction *строка*

Действие, которое будет выполнено по результатам проверки ограничений:

- *Deny* — запрет;
- *Dryrun* — отсутствие действия. Применяется при отладке. Информацию о событии можно посмотреть в Grafana или консоли с помощью d8 k;
- *Warn* — аналогично Dryrun, но дополнительно к информации о событии будет выведена информация о том, из-за какого ограничения (constraint) был бы запрет действия, если бы вместо Warn использовался Deny.

По умолчанию: "Deny"

- podSecurityStandards.policies *объект*

Определяет дополнительные параметры политик

- podSecurityStandards.policies.hostPorts *объект*

Настройки ограничения HostPort.

- podSecurityStandards.policies.hostPorts.knownRanges *массив объектов*

Список диапазонов портов, которые будут разрешены в привязке hostPort.

- podSecurityStandards.policies.hostPorts.knownRanges.max
- podSecurityStandards.policies.hostPorts.knownRanges.min

5.3 Настройка уведомлений о событиях безопасности на почту

Чтобы настроить отправку уведомлений о событиях безопасности на почту, необходимо выполнить следующую команду:

```
d8 k apply -f email.yaml
```

Пример файла email.yaml:

```
apiVersion: deckhouse.io/v1alpha1
kind: CustomAlertmanager
spec:
  internal:
    receivers:
    - emailConfigs:
      - authPassword:
          key: password
          name: alertmanager-email
        authUsername: stand@mg.flant.dev
```

```
      from: stand@mg.flant.dev
      sendResolved: true
      smarthost: smtp.mailgun.org:25
      to: example@flant.com
    name: email
  route:
    groupBy:
    - job
    groupInterval: 5m
    groupWait: 30s
    receiver: email
    repeatInterval: 12h
type: Internal
```

5.4 Настройка доступа к журналам событий безопасности

Чтобы настроить доступ пользователям к Grafana и Prometheus, необходимо выполнить следующую команду:

```
d8 k apply -f prometheus.yaml
```

Пример файла prometheus.yaml:

```
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: prometheus
spec:
  enabled: true
  settings:
    auth:
      allowedUserGroups:
      - administrator
      - security
  version: 2
```

Здесь:

- auth – опции, связанные с аутентификацией или с авторизацией в приложении.
- auth.allowedUserGroups – массив групп, пользователям которых позволен доступ в Grafana и Prometheus.

5.5 Просмотр журналов событий безопасности

Просмотр журналов событий безопасности осуществляется в веб-интерфейсе Grafana. Необходимые дашборды сгруппированы в папке Security:

- *Admission policy engine.* Содержит информацию, связанную с работой политик безопасности. В том числе: количество событий запрета выполнения действий из-за

нарушения политики безопасности; разбивку запретов выполнения действий по типу запрета; журнал событий.

Журнал событий безопасности, связанных с политиками безопасности, находится в окне OPA Violations.

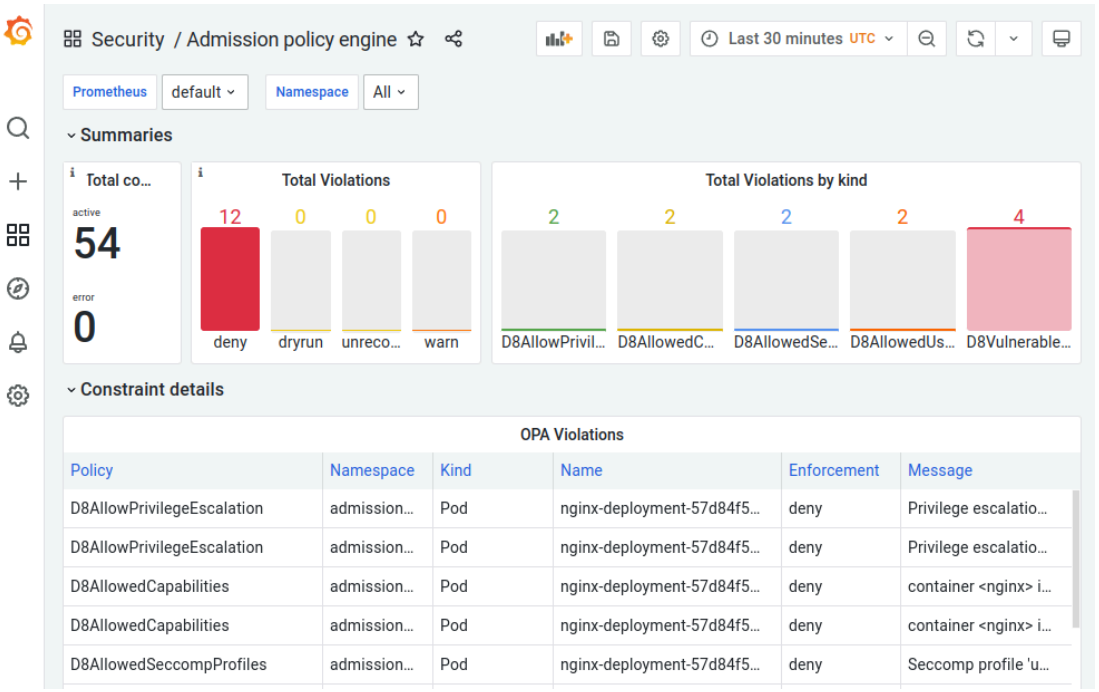

Рисунок 1. Пример дашборда Admission policy engine.

- *CIS Kubernetes Benchmark.* Дашборд с информацией о результатах работы сканера проверок конфигурации кластера на соответствие принятым подходам (лучшим практикам). Содержит сводную информацию о результатах проверки, без возможности детализации. Дашборд доступен при включенном модуле operator-trivy (см. п.5.1).

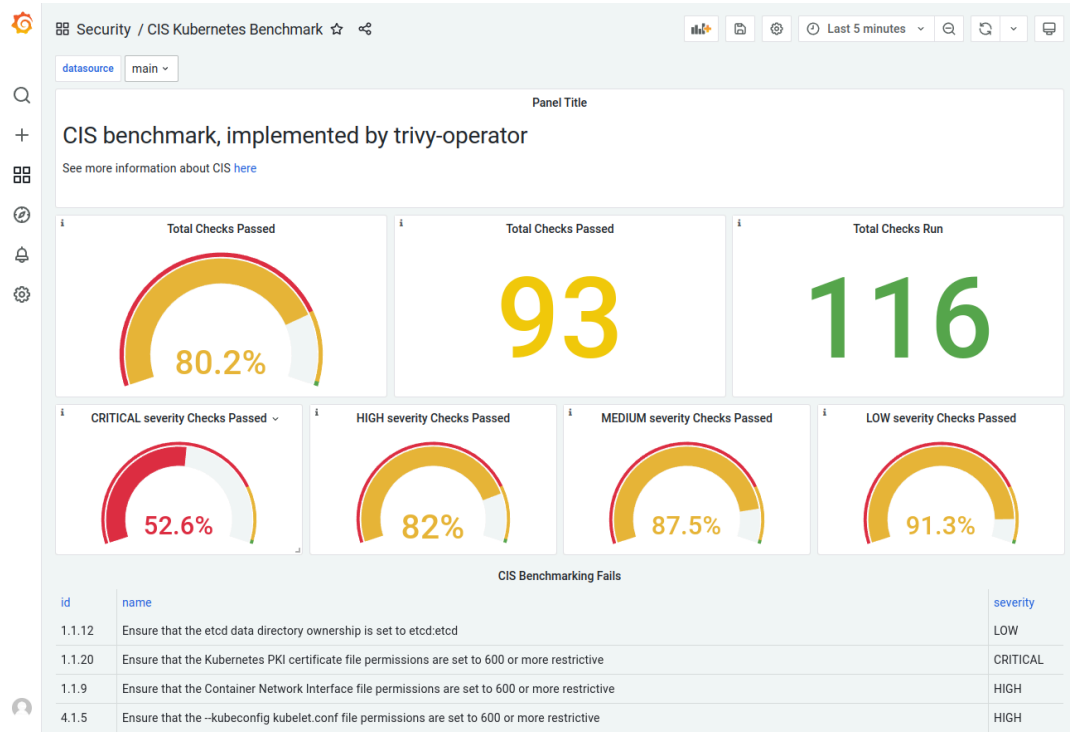

Рисунок 2. Пример дашборда CIS Kubernetes Benchmark.

- *Kubernetes audit logs.* Журнал регистрации обращений к API-серверу. Содержит записи о всех обращениях к API-серверу кластера в JSON-формате.

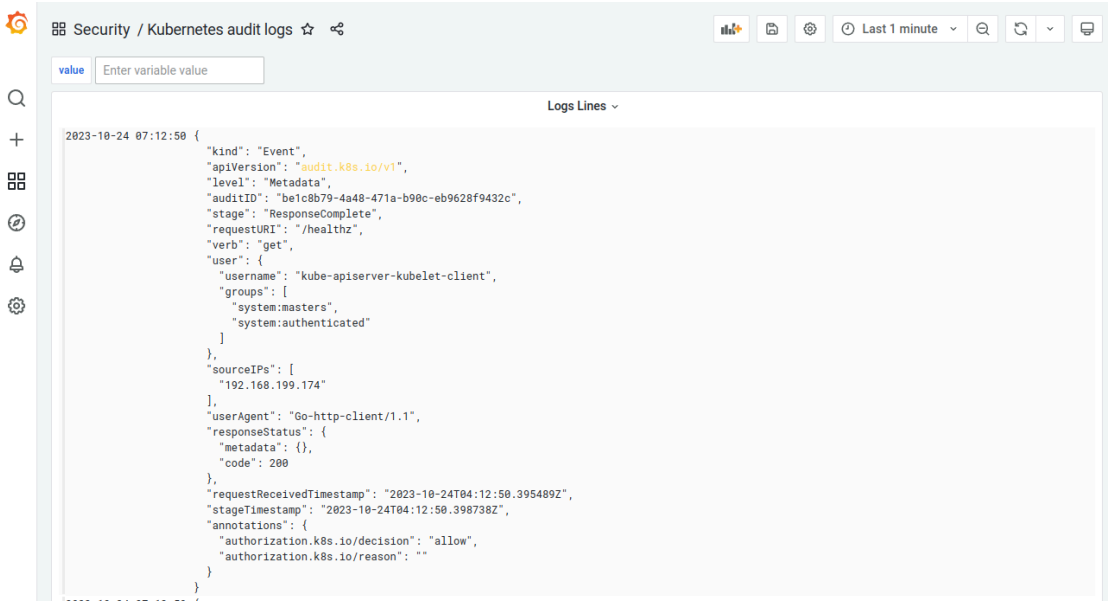

Рисунок 3. Пример дашборда Kubernetes audit logs.

- *Runtime audit engine logs.* Журнал регистрации событий безопасности аудита работы ядра Linux и API-сервера кластера.

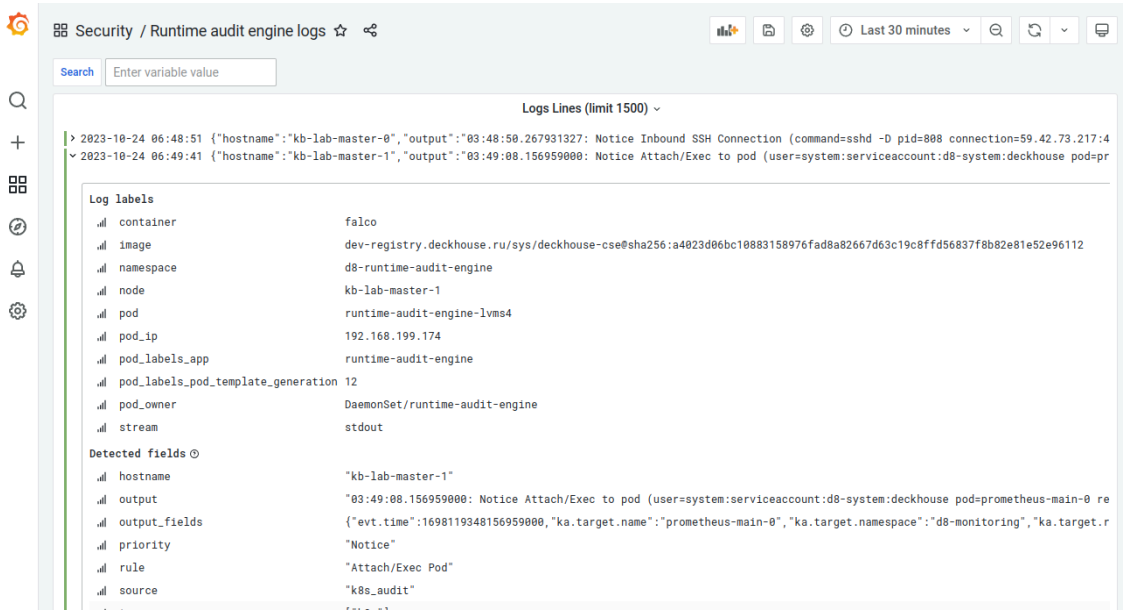

Рисунок 4. Пример дашборда Runtime audit engine logs.

- *Trivy Image Vulnerability Overview.* Дашборд со сводной и детализированной информацией о сканировании образов контейнеров подов в пространствах имен, отмеченных аннотацией *security-scanning.deckhouse.io/enabled* (см. п.5.1).

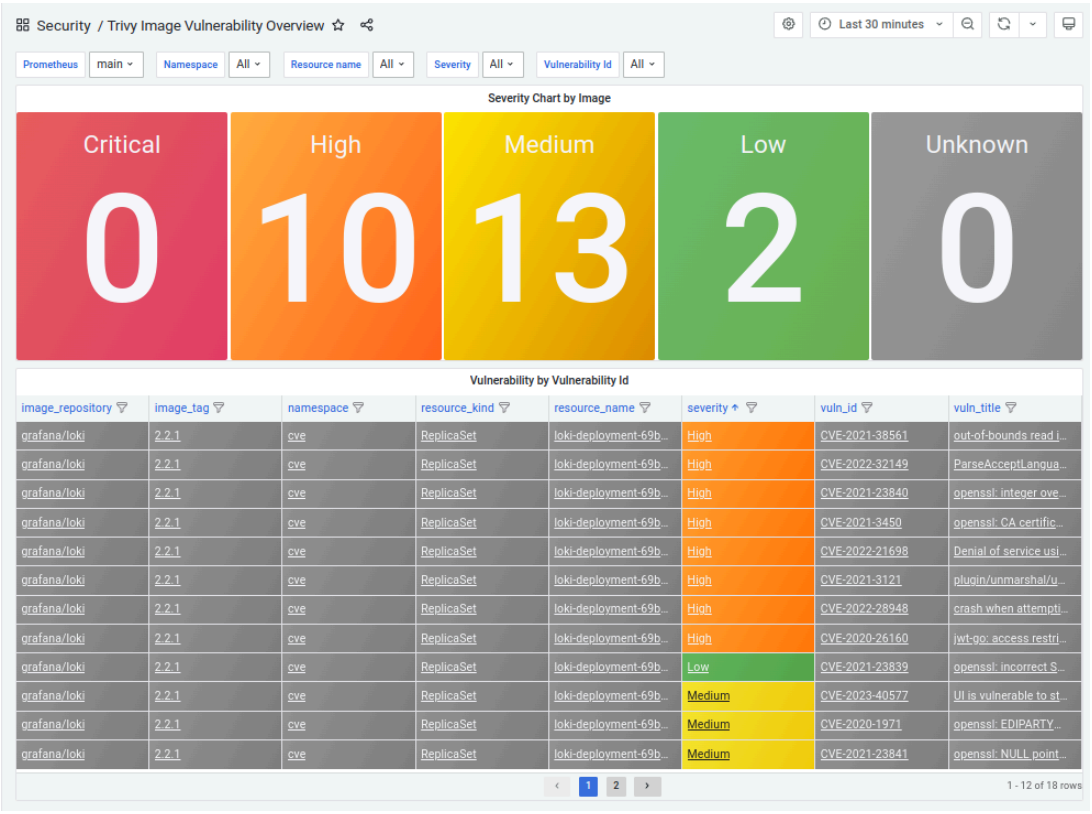

Рисунок 5. Пример дашборда Trivy Image Vulnerability Overview.

5.6 Хранение журналов событий безопасности

В ПО «Deckhouse Platform» хранение журналов событий безопасности реализовано с помощью модуля loki.

Объем выделенного хранилища указывается параметром diskSizeGigabytes в объекте ModuleConfig и соответствует размеру PersistentVolume, который может быть изменен администратором при необходимости.

Встроенное автоматизированное архивирование локальных данных по умолчанию отсутствует. Удаление устаревших данных производится автоматически согласно механизму ротации при достижении ограничений по размеру хранилища или по завершению срока хранения (retentionPeriodHours). Освобождаемая память становится доступна для новых данных.

Пример манифеста с указанием объёма выделенного хранилища (diskSizeGigabytes) и срока хранения (retentionPeriodHours):

```
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: loki
spec:
  settings:
    storageClass: ceph-csi-rbd
    diskSizeGigabytes: 30
    retentionPeriodHours: 168
  enabled: true
  version: 1
---
apiVersion: deckhouse.io/v1alpha2
kind: ClusterLoggingConfig
metadata:
  name: development-logs
spec:
  type: KubernetesPods
  kubernetesPods:
    namespaceSelector:
      labelSelector:
        matchExpressions:
        - key: "kubernetes.io/metadata.name"
          operator: In
          values: [development]
  destinationRefs:
    - d8-loki
```

Для организации долговременного хранения журналов аудита или создания архивов реализована поддержка ручного резервного копирования с помощью утилиты d8 backup loki. Использование данного механизма позволяет переносить и создавать резервную копию журнала

событий в соответствии с внутренними процедурами, согласованными с требованиями организации.

В случае необходимости длительного хранения или передачи данных во внешние системы может использоваться модуль log-shipper либо интеграция с внешним экземпляром Loki или объектным хранилищем.

Таким образом, требования по контролю выделяемого пространства и последующей очистке выполняются штатными средствами модуля. Возможность переноса или архивирования информации обеспечивается дополнительными средствами платформы вне рамок автоматизированного процесса.

5.6.1 Выгрузка логов

Команда `d8 backup loki` предназначена для выгрузки логов из встроенного Loki. Это диагностическая выгрузка: полученные данные нельзя восстановить обратно в Loki.

Для успешной выгрузки d8 обращается к Loki API от имени ServiceAccount loki в пространстве имён d8-monitoring, используя секрет с токеном.

ServiceAccount loki создаётся автоматически. Однако для работы команды `d8 backup loki` необходимо вручную создать секрет и назначить Role и RoleBinding, если они ещё не заданы.

Примените манифесты перед запуском d8 backup loki, чтобы команда корректно получала токен и могла обращаться к Loki API.

Пример манифеста:

```
---
apiVersion: v1
kind: Secret
metadata:
  name: loki-api-token
  namespace: d8-monitoring
  annotations:
    kubernetes.io/service-account.name: loki
type: kubernetes.io/service-account-token
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: access-to-loki-from-d8
  namespace: d8-monitoring
rules:
  - apiGroups: ["apps"]
    resources:
      - "statefulsets/http"
    resourceNames: ["loki"]
```

```
    verbs: ["create", "get"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: access-to-loki-from-d8
  namespace: d8-monitoring
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: access-to-loki-from-d8
subjects:
  - kind: ServiceAccount
    name: loki
    namespace: d8-monitoring
```

Для создания резервной копии выполните команду:

```
d8 backup loki [флаги]
```

Пример:

```
d8 backup loki --days 1 > ./loki.log
```

Флаги:

- *--start, --end* — временные метки в формате "YYYY-MM-DD HH:MM:SS";

- *--days* — ширина временного окна выгрузки (по умолчанию 5 дней);

- *--limit* — максимум строк в одном запросе (по умолчанию 5000).

Список доступных флагов можно получить через следующую команду:

```
d8 backup loki --help
```

5.7 Контроль целостности

Контроль целостности — совокупность механизмов проверки контейнеров или данных, которые обеспечивают их безопасность и соответствие заданной конфигурации.

В ПО «Deckhouse Platform» контроль целостности реализован на трех уровнях:

- при запуске контейнеров приложений;
- во время работы контейнеров приложений;
- при взаимодействии контейнеров с данными, хранимыми в кластере в etcd.

5.7.1 Контроль целостности при запуске контейнеров

ПО «Deckhouse Platform» выполняет проверку образов на уровне контейнерного рантайма (CRI).

После загрузки образа проверяется его хеш-сумма SHA-256. Запуск возможен только при успешной верификации.

Последовательность контроля целостности при запуске:

1. Загрузка образа в локальное хранилище узла.
2. Извлечение метаданных образа, включая хеш-сумму SHA-256.
3. Верификация SHA-256 путем сравнения с эталонной.
4. Если хеш совпадает, проверка пройдена. Если хеш не совпадает, образ не запускается.

Для повышения уровня безопасности можно настраивать политики загрузки образов при использовании политик безопасности, чтобы запретить, например, использование образов контейнеров имеющих известные уязвимости. Настройки политик безопасности описаны в п. 5.2.

5.7.2 Контроль целостности работающих контейнеров

Аудит событий безопасности в ПО «Deckhouse Platform» включает анализ событий ядра Linux и аудита событий Kubernetes API. Это позволяет отслеживать, что приложения в подах работают в неизменном виде, соответствуют ожидаемому состоянию и не были модифицированы.

Для аудита используются:

- встроенные правила;
- пользовательские правила, которые можно добавлять с использованием синтаксиса условий Falco.

Для контроля целостности работающих контейнеров применяются встроенные и пользовательские правила.

В процессе контроля целостности работающих контейнеров могут выявляться такие угрозы, как запуск оболочек командной строки в контейнерах или подах, обнаружение контейнеров, работающие в привилегированном режиме, монтирование небезопасных путей в контейнеры, попытки чтения секретных данных.

5.7.3 Контроль целостности хранимых в etcd данных

Контроль целостности хранимых данных в ПО «Deckhouse Platform» реализован посредством преобразования информации, сохраняемой во внутренней базе данных etcd.

Хранение данных в etcd осуществляется в формате структуры, включающей данные и электронную подпись.

Процесс управления данными в etcd включает два этапа: запись и чтение.

Запись данных. Для каждого набора данных вычисляется электронная подпись, которая добавляется к записи. Это обеспечивает возможность проверки того, что данные не подвергались изменению после их первоначальной записи.

Чтение данных. При извлечении данных осуществляется процедура верификации, в ходе которой данные проверяются на соответствие электронной подписи. При выявлении несоответствия между данными и подписью система действует в соответствии с предопределенными параметрами конфигурации.

Настройки контроля целостности данных хранимых в etcd предусматривают три режима работы:

- *Enforce* — запрещает обработку данных с недействительной подписью.

  В случае, если данные не соответствуют электронной подписи, они будут отброшены, а система сгенерирует предупреждение и создаст запись в аудит-логе. В этом режиме обеспечивается высокий уровень надежности и безопасности хранения данных, минимизируя вероятность их несанкционированного изменения или компрометации в процессе хранения.
- *Migrate* — режим миграции и обеспечения совместимости формата данных при контроле целостности. Несоответствие подписи не блокирует обработку данных, однако система инициирует генерацию предупреждения и запись в аудит-лог.
- *Rollback* (значение по умолчанию) — преобразование хранимых данных в стандартный формат (формат по умолчанию для etcd).

  Данные без подписи, а также с невалидной подписью допускаются к обработке, генерируется предупреждение и запись в аудит-лог. Данные преобразуются в стандартный формат только при записи объекта.

Внимание! *Переход к режиму работы Enforce в кластере, содержащем данные в etcd, не преобразованные в новый формат, приведет к сбою функционирования кластера. Алгоритм миграции данных рассмотрен в разделе 4.16.*

Режим работы контроля целостности хранимых данных указывается параметром apiserver.signature в параметрах модуля control-plane-manager (объект ModuleConfig control-plane-manager).

Пример манифеста ModuleConfig control-plane-manager с указанием режима работы:

```
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: control-plane-manager
spec:
  settings:
    apiserver:
      signature: Enforce
```

5.7.4 Проверка целостности образа

Для подписания образов в репозитории пользователя в ПО «Deckhouse Platform» используются два механизма:

- Cosign,
- imagedigest (устаревший).

5.7.4.1 Проверка целостности образа с помощью Cosign

Утилита Cosign входит в состав поставки.

Чтобы подписать образ с помощью Cosign, выполните следующее:

1. Сгенерируйте пару ключей (публичный и приватный):

```
cosign generate-key-pair
```

2. Подпишите образ в хранилище образов контейнеров с помощью сгенерированного приватного ключа:

```
cosign sign --key <KEY> <REGISTRY_IMAGE_PATH>
```

Здесь:

- `<REGISTRY_IMAGE_PATH>` - путь к образу, который нужно указать при запуске, например: registry.private.ru/labs/application/image:latest.

3. Чтобы включить проверку подписи образов контейнеров в кластере ПО «Deckhouse Platform», используйте параметр policies.verifyImageSignatures ресурса SecurityPolicy, указав публичный ключ, сгенерированный на шаге 1.

   Пример конфигурации SecurityPolicy для проверки подписи образов контейнеров в хранилище registry.private.ru, размещенные по пути /labs/application/:

```
apiVersion: deckhouse.io/v1alpha1
kind: SecurityPolicy
```

```
metadata:
  name: verify-image-test
spec:
  enforcementAction: Deny
  match:
    namespaceSelector:
      labelSelector:
        matchLabels:
          kubernetes.io/metadata.name: test-namespace
  policies:
    allowHostIPC: true
    allowHostNetwork: true
    allowHostPID: false
    allowPrivilegeEscalation: true
    allowPrivileged: false
    allowRbacWildcards: true
    verifyImageSignatures:
    - publicKeys:
      - |-
        -----BEGIN PUBLIC KEY-----
        MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEhpqaufY9JSY+g4JZmmEWCxYp4BSj
        YAzTW+LBJa6GwiJ+iWHMEw2w8aiVk7NSayEp5ZDZaBTmspT/dyuWSpazPQ==
        -----END PUBLIC KEY-----
      reference: registry.private.ru/labs/application/*
```

4. Создайте ресурс OperationPolicy, ограничивающий запуск подов со сторонних registry:

```
apiVersion: deckhouse.io/v1alpha1
kind: OperationPolicy
metadata:
  name: test-operation-policy
spec:
  enforcementAction: Deny
  match:
    namespaceSelector:
      labelSelector:
        matchLabels:
          operation-policy.deckhouse.io/enabled: "true"
  policies:
    allowedRepos:
    - registry.private.ru
```

5. Для проверки работы механизма подписи образов разверните поды в пространстве имён, с подписанным и неподписанным образами (укажите нужное пространство имён):

```
kubectl  -n <NAMESPACE> run signed-pod --image=<ПОДПИСАННЫЙ_ОБРАЗ>
kubectl  -n <NAMESPACE> run unsigned-pod --image=<НЕПОДПИСАННЫЙ_ОБРАЗ>
```

Согласно данной политике, если адрес какого-либо образа контейнера совпадает со значением параметра reference и образ не подписан или подпись не соответствует указанным ключам, создание пода будет запрещено.

Пример вывода ошибки при создании пода с образом контейнера, не прошедшим проверку подписи:

```
[verify-image-signatures] Image signature verification failed: nginx:1.17.2
```

5.7.4.2 Проверка целостности образа с помощью imagedigest

Для проверки используется контрольная сумма, рассчитанная по алгоритму Стрибог (ГОСТ Р 34.11-2012).

Для работы проверки целостности образа по умолчанию включен модуль gost-integrity-controller. Если модуль не включен, то для включения модуля используйте команду:

```
d8 system module enable gost-integrity-controller
```

После включения модуля проверьте, что под `gost-digest-webhook` запустился:

```
d8 k get po -A |grep gost
```

Пример вывода:

```
d8-system                    gost-digest-webhook-56f59c48bb-b5njj                2/2
Running   0              160m
```

Чтобы образы контейнеров проверялись, необходимо добавить метку gost-integrity-controller.deckhouse.io/gost-digest-validation-enabled: true на пространство имен кластера, где необходимо производить контроль целостности образа.

Пример команды:

```
d8 k label ns test-gost
gost-integrity-controller.deckhouse.io/gost-digest-validation-enabled=true
```

Если в ходе проверки контрольная сумма образа окажется некорректной, запуск контейнера с таким образом будет запрещен, также будет выведено соответствующее сообщение.

Для расчета контрольной суммы берется список контрольных сумм слоев образа. Список сортируется в порядке возрастания и склеивается в одну строку. Затем производится расчет контрольной суммы от этой строки по алгоритму Стрибог (ГОСТ Р 34.11-2012).

Пример расчета контрольной суммы образа nginx:1.25.2:

Контрольные суммы слоев отсортированные в порядке возрастания:

```
[
"sha256:27e923fb52d31d7e3bdade76ab9a8056f94dd4bc89179d1c242c0e58592b4d5c",
"sha256:360eba32fa65016e0d558c6af176db31a202e9a6071666f9b629cb8ba6ccedf0",
```

```
"sha256:72de7d1ce3a476d2652e24f098d571a6796524d64fb34602a90631ed71c4f7ce",
"sha256:907d1bb4e9312e4bfeabf4115ef8592c77c3ddabcfddb0e6250f90ca1df414fe",
"sha256:94f34d60e454ca21cf8e5b6ca1f401fcb2583d09281acb1b0de872dba2d36f34",
"sha256:c5903f3678a7dec453012f84a7d04f6407129240f12a8ebc2cb7df4a06a08c4f",
"sha256:e42dcfe1730ba17b27138ea21c0ab43785e4fdbea1ee753a1f70923a9c0cc9b8"
]
```

Объединённая строка всех контрольных сумм:

```
"sha256:27e923fb52d31d7e3bdade76ab9a8056f94dd4bc89179d1c242c0e58592b4d5c
sha256:360eba32fa65016e0d558c6af176db31a202e9a6071666f9b629cb8ba6ccedf0
sha256:72de7d1ce3a476d2652e24f098d571a6796524d64fb34602a90631ed71c4f7ce
sha256:907d1bb4e9312e4bfeabf4115ef8592c77c3ddabcfddb0e6250f90ca1df414fe
sha256:94f34d60e454ca21cf8e5b6ca1f401fcb2583d09281acb1b0de872dba2d36f34
sha256:c5903f3678a7dec453012f84a7d04f6407129240f12a8ebc2cb7df4a06a08c4f
sha256:e42dcfe1730ba17b27138ea21c0ab43785e4fdbea1ee753a1f70923a9c0cc9b8"
```

Контрольная сумма образа:

```
2f538c22adbdb2ca8749cdafc27e94baed8645c69d4f0745fc8889f0e1f5a3f9
```

С помощью утилиты imagedigest выполняется расчет контрольной суммы образа, добавление контрольной суммы в метаданные образа и проверка контрольной суммы образа:

- imagedigest calculate <имя_образа> - расчет контрольной суммы образа.
- imagedigest add <имя_образа> - добавление контрольной суммы в метаданные образа.
- imagedigest validate <имя_образа> - проверка контрольной суммы.

Для каждого образа администратор должен фиксировать в журнал идентификатор пользователя, подписавшего образ, и вычисленную контрольную сумму образа.

5.7.4.3 Работа с контрольной суммой образов

Для ручного расчёта, добавления и проверки контрольной суммы образов контейнеров с используется утилита `d8 tools imagedigest`, входящая в состав ПО «Deckhouse Platform Certified Security Edition».

С помощью утилиты можно заранее вычислить и добавить контрольную сумму образа в его метаданные.

Для расчёта контрольной суммы образа выполните команду:

```
d8 tools imagedigest calculate <image>
```

Для расчёта контрольной суммы файла выполните команду:

```
d8 tools imagedigest calculate-from-file <file>
```

Для добавления контрольной суммы в метаданные образа выполните команду:

```
d8 tools imagedigest add <image>
```

После выполнения команды в метаданные образа будет добавлена контрольная сумма, рассчитанная по алгоритму Стрибог (ГОСТ Р 34.11-2012).

Для проверки контрольной суммы, записанной в метаданных образа, выполните команду:

```
d8 tools imagedigest validate <image>
```

Если контрольная сумма некорректна, утилита завершится с ошибкой. Для автоматического исправления контрольной суммы используйте параметр —fix.

```
d8 tools imagedigest validate <image> --fix
```

В этом случае утилита пересчитает контрольную сумму и обновит её в метаданных образа. Использование параметра `--fix` изменяет метаданные образа.

Утилита поддерживает дополнительные параметры:

- --insecure — разрешает небезопасные подключения к реестрам (без проверки TLS);
- --debug — включает режим отладки;
- --json — выводит логи в формате JSON.

5.8 Управление информационными потоками

5.8.1 Фильтрация

Фильтрация информационных потоков осуществляется в соответствии с правилами управления потоками, установленными администратором безопасности. Реализуется фильтрация средствами модуля cni-cilium. Политики позволяют ограничивать доступ между подами, пространствами имён и внешними системами.

Ниже приведены базовые сценарии, соответствующие требованиям по ограничению сетевых взаимодействий между компонентами.

- Полный запрет всех исходящих потоков из пространства имён:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
  namespace: default
spec:
  podSelector: {}
  policyTypes:
  - Egress
  egress: []
```

- Разрешение входящих потоков только на определённый порт. В примере поды с ролью db принимают соединения от backend только на порт 5432.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-db-traffic
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          role: backend
    ports:
    - protocol: TCP
      port: 5432
```

- Разрешение исходящих потоков только на DNS и HTTP/HTTPS. В примере под-client может обращаться к DNS внутри кластера и в интернет только по протоколам HTTP и HTTPS.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-dns-http
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: client
  policyTypes:
```

```
  - Egress
  egress:
  - to:
    - namespaceSelector: {}
      podSelector:
        matchLabels:
          k8s-app: kube-dns
    ports:
    - protocol: UDP
      port: 53
  - to:
    - ipBlock:
        cidr: 0.0.0.0/0
    ports:
    - protocol: TCP
      port: 80
    - protocol: TCP
      port: 443
```

5.8.2 Использование заданных маршрутов

В ПО «Deckhouse Platform» маршрутизация передачи информации определяется на этапе настройки системы. Использование заданных маршрутов для передачи информации разрешается администратором безопасности и реализуется средствами модуля cni-cilium и политик NetworkPolicies.

Примеры:

- Разрешение входящих потоков только от подов с определённой меткой. Пример, когда поды с ролью backend принимают входящие потоки только от подов с ролью frontend.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-frontend
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: backend
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          role: frontend
```

- Ограничение исходящих потоков в определённое пространство имён. В примере поды приложения my-app могут передавать данные только в пространство имён monitoring и в кластерный DNS.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-egress-to-monitoring
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: my-app
  policyTypes:
  - Egress
  egress:
  - to:
    - namespaceSelector: {}
      podSelector:
        matchLabels:
          k8s-app: kube-dns
    ports:
    - protocol: UDP
      port: 53
  - to:
    - namespaceSelector:
        matchLabels:
          purpose: monitoring
```

5.8.3 Перенаправление маршрута

Возможность изменения (перенаправления) маршрутов передачи информации для исходящего трафика реализуется с помощью модуля cni-cilium и позволяет администратору безопасности задать централизованный выходной шлюз (EgressGateway), через который будут передаваться все запросы из кластера во внешние информационные системы. EgressGateway используется совместно с политиками EgressGatewayPolicy для применения правил к конкретным приложениям или пространствам имён.

Для работы шлюза должны быть выполнены условия:

- узлы, назначенные для работы EgressGateway, находятся в состоянии Ready и не переведены в обслуживание (cordon);
- на узлах успешно работает агент сетевого взаимодействия (cilium-agent).

Администратор безопасности определяет параметры объекта EgressGateway в спецификации (spec).

Основные параметры:

- *nodeSelector* — выбор группы узлов, которые будут обслуживать исходящий трафик. Среди них система автоматически выбирает активный узел. При выходе из строя узла производится автоматическое переключение.
- *sourceIP.mode* — способ назначения исходящего IP-адреса:
    - *PrimaryIPFromEgressGatewayNodeInterface* — используется основной адрес сетевого интерфейса узла. В этом режиме при переключении узлов адрес отправителя изменится.
    - *VirtualIPAddress* — используется виртуальный IP-адрес, общий для группы узлов. При переключении активного узла адрес отправителя сохраняется неизменным.

Для настройки EgressGateway можно использовать следующий пример:

```
apiVersion: network.deckhouse.io/v1alpha1
kind: EgressGateway
metadata:
  name: my-egressgw
spec:
  nodeSelector:
    matchLabels:
      role: egress-gateway
  sourceIP:
    mode: VirtualIPAddress
    interfaceNames:
    - eth1
    virtualIP: 192.168.1.100
```

- В режиме PrimaryIP необходимо указать имя сетевого интерфейса (например, eth1).
- В режиме Virtual IP указываются:
    - список интерфейсов, доступных для работы с виртуальным IP;
    - сам виртуальный IP-адрес.

Далее EgressGateway используется совместно с EgressGatewayPolicy – для назначения, какой исходящий трафик должен направляться через данный шлюз. Пример политики EgressGatewayPolicy:

```
apiVersion: network.deckhouse.io/v1alpha1
kind: EgressGatewayPolicy
metadata:
  name: my-egressgw-policy
```

```
spec:
  destinationCIDRs:
  - 0.0.0.0/0
  egressGatewayName: my-egressgw
  selectors:
  - podSelector:
      matchLabels:
        app: backend
        io.kubernetes.pod.namespace: my-ns
```

5.8.4 Визуализация сетевых взаимодействий

Для оперативной диагностики и анализа сетевого взаимодействия в ПО «Deckhouse Platform» используется веб-интерфейс визуализации сетевого стека кластера. Он даёт возможность отслеживать сетевые взаимодействия между подами, сервисами и внешними ресурсами, анализировать сетевую активность и выявлять проблемы с сетью. Реализация этого интерфейса осуществляется модулем cilium-hubble.

Для перехода в веб-интерфейс визуализации сетевого стека кластера откройте адрес hubble.<ШАБЛОН_ИМЕН_КЛАСТЕРА>, где <ШАБЛОН_ИМЕН_КЛАСТЕРА> – строка, соответствующая шаблону DNS-имен кластера, указанному в глобальном параметре modules.publicDomainTemplate.

При переходе по адресу hubble.<ШАБЛОН_ИМЕН_КЛАСТЕРА> откроется экран выбора пространства имен, для которого будет отображаться сетевой стек.

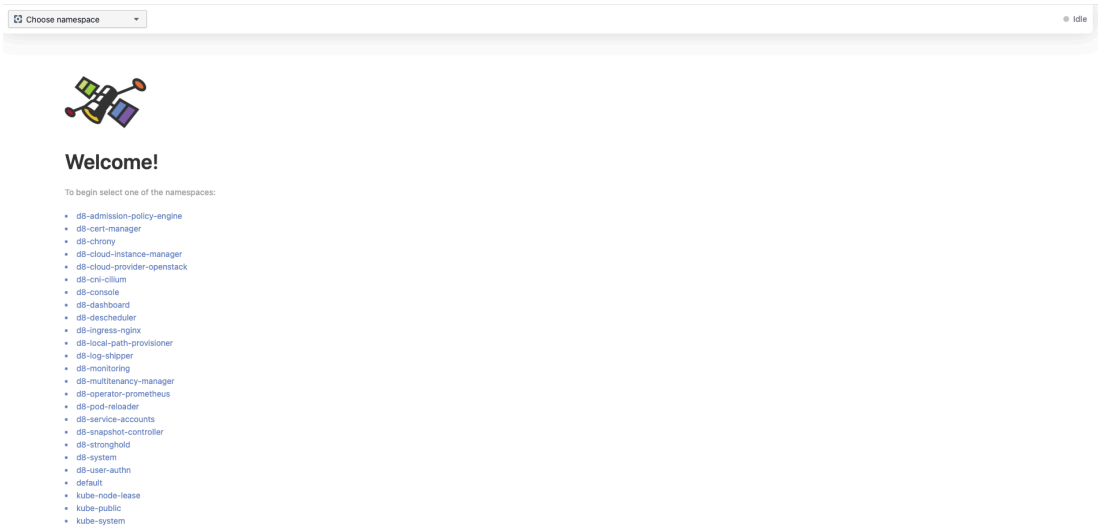

Рисунок 6. Выбор пространства имён.

Выберите пространство имен с помощью выпадающего списка в левой верхней части экрана или нажав на название нужного пространства имен в списке в центре экрана.

После выбора пространства имен откроется экран с визуализацией сетевого стека и средствами анализа. Он состоит из следующих частей:

- Верхняя панель с фильтрами и краткой сводкой по кластеру (количество потоков и количество узлов).
- Схема сетевых потоков.
- Таблица сетевых потоков и событий.

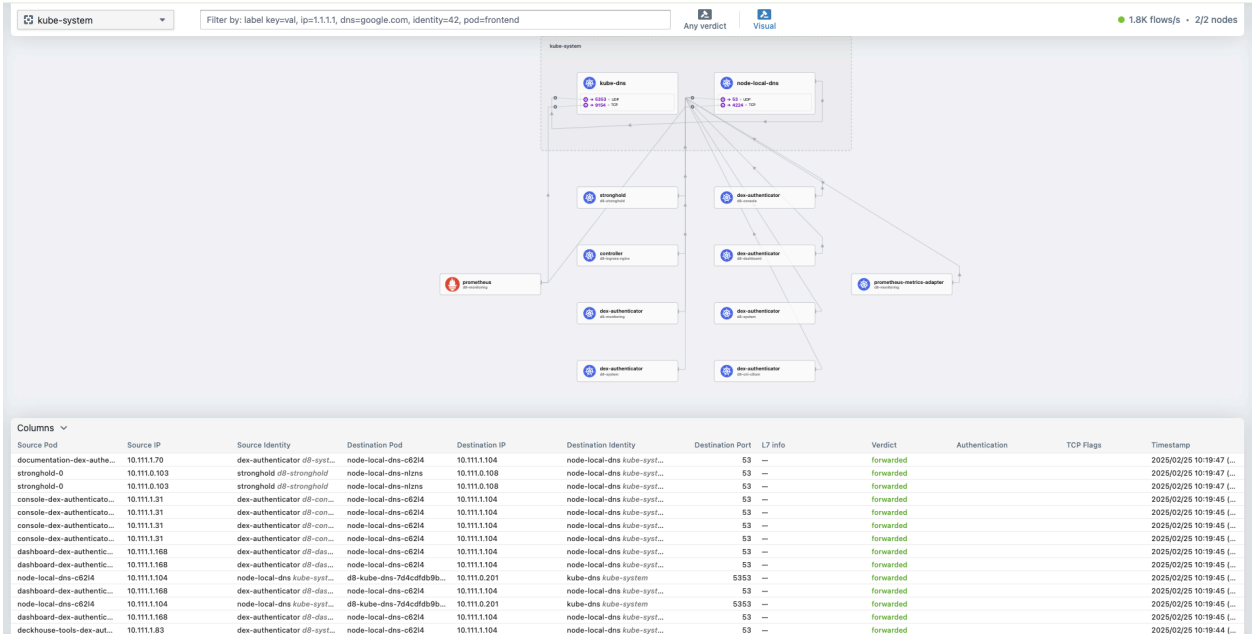

Рисунок 7. Визуализация сетевого стека.

Данные на схеме и в таблице сетевых потоков отображаются в реальном времени.

5.8.4.1 Фильтрация данных для отображения

Чтобы отфильтровать отображаемые данные о сетевом стеке и потоках, воспользуйтесь верхней панелью с фильтрами. Здесь расположены фильтры:

- для выбора пространства имен (выпадающий список в левой части панели);

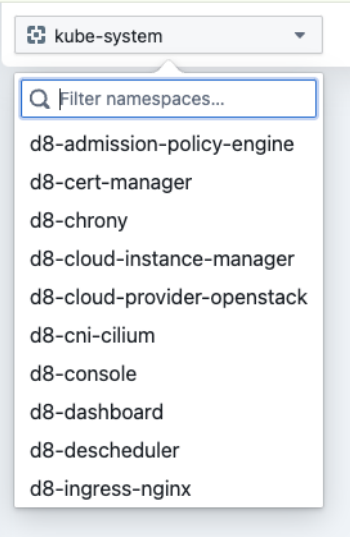

Рисунок 8. Фильтр для выбора пространства имён.

- для выбора ресурсов пространства имен, для которых нужно отобразить потоки (поле ввода в центральной части панели);

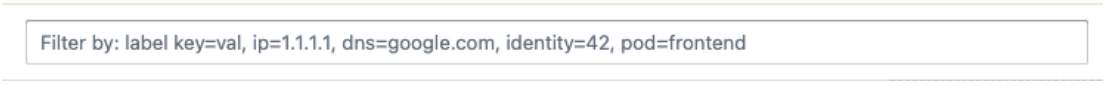

Рисунок 9. Фильтр для выбора ресурсов пространств имён.

- для выбора сетевых потоков на основе решения («вердикта»), принятого по ним Cilium;

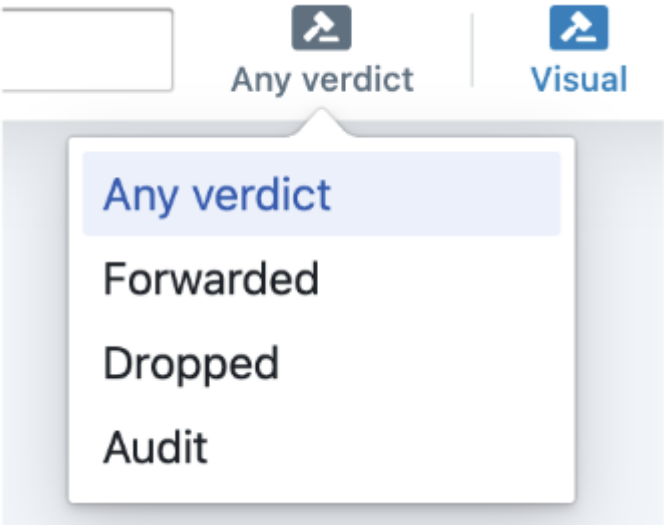

Рисунок 10. Фильтр для выбора сетевых потоков по «вердикту».

- для выбора элементов схемы анализируемого пространства имен.

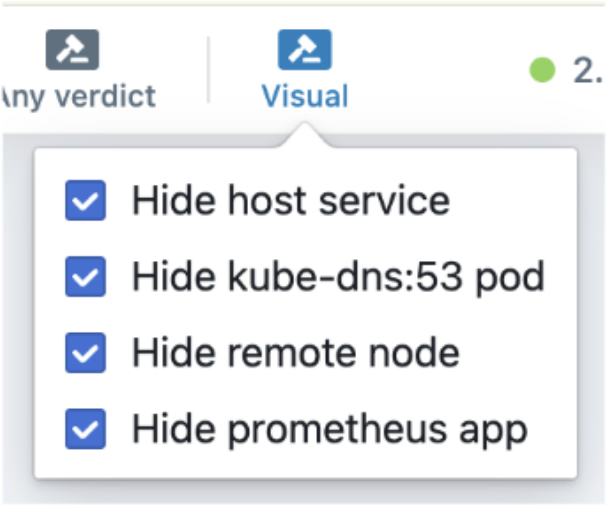

Рисунок 11. Фильтр для выбора элементов схемы анализируемого пространства имен.

5.8.4.2 Работа со схемой сетевых потоков

Схема сетевых потоков для выбранного пространства имен отображается в средней части экрана с визуализацией сетевого стека и средствами анализа. На схеме отображаются ресурсы выбранного пространства имен, расположенные в прямоугольнике с названием пространства имен, и внешние элементы, с которыми они взаимодействуют.

Чтобы посмотреть детальную информацию (список лейблов, сетевые взаимодействия и т.д.) по конкретному ресурсу на схеме, нажмите на него.

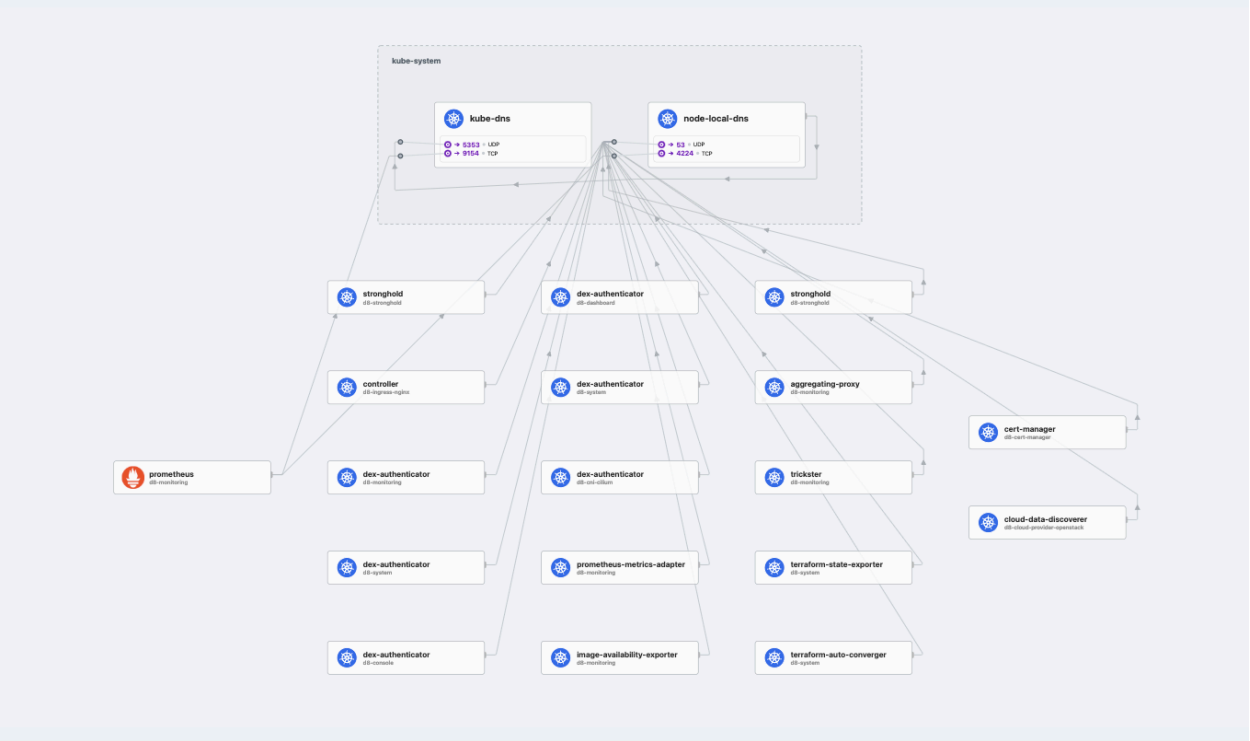

Рисунок 12. Схема сетевых потоков.

5.8.4.3 Работа с таблицей сетевых потоков и событий

Каждая строка таблицы содержит следующую информацию о сетевом потоке:

- имя пода — источника потока (столбец «Source Pod»);
- IP-адрес пода — источника потока (столбец «Source IP»);
- идентификатор сущности — источника потока (столбец «Source Identity»);
- имя пода — получателя потока (столбец «Destination Pod»);
- IP-адрес пода — получателя потока (столбец «Destination IP»);
- идентификатор сущности-получателя (столбец «Destination Identity»);
- номер порта назначения (столбец «Destination Port»);
- информация о прикладном уровне (Layer 7), если поток использует протоколы HTTP (столбец «L7 info»);
- результат («вердикт») обработки сетевого потока Cilium (столбец «Verdict»);

- информация о результатах проверки подлинности сетевого потока, если такая проверка выполнялась (столбец «Authentication»);
- флаги TCP, связанные с потоком (столбец «TCP Flags»);
- временная метка потока (столбец «Timestamp»).

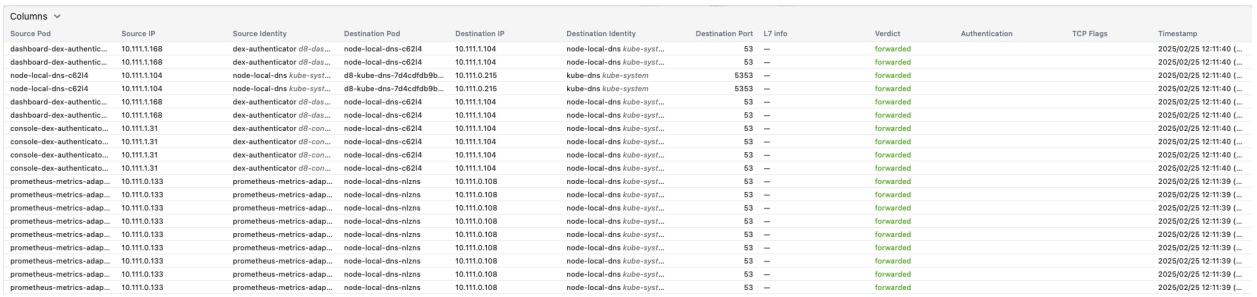

Рисунок 13. Таблица сетевых потоков и событий.

Чтобы настроить набор столбцов, отображаемых в таблице, нажмите на кнопку «Columns» в ее левой верхней части и выберете нужные.

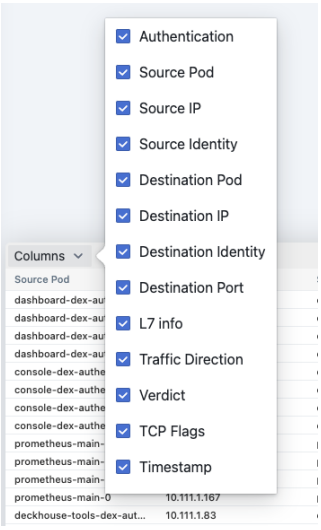

Рисунок 14. Настройка столбцов.

Чтобы посмотреть информацию о записи таблицы в текстовом виде, нажмите в любой части соответствующей строки. Информация отобразится в правой части таблицы. Данные здесь отображаются независимо от того, какой набор столбцов выбран для отображения в таблице.

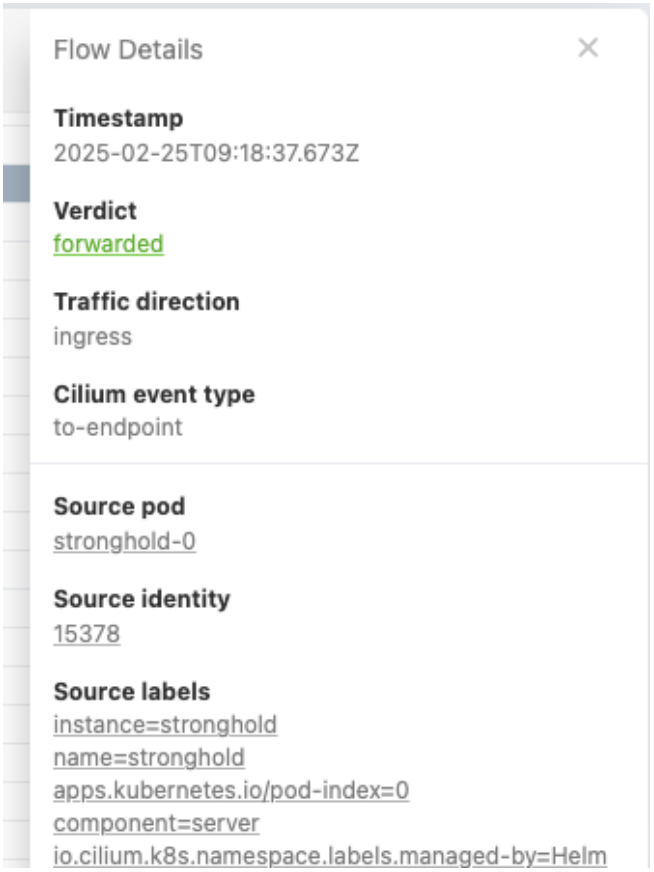

Рисунок 15. Информация о записи.

6 Действия по реализации функций безопасности среды функционирования средства

В среде функционирования ПО «Deckhouse Platform» должны быть реализованы следующие функции безопасности:

- физическая защита;
- доверенная загрузка ПО «Deckhouse Platform»;
- обеспечение условий безопасного функционирования ПО «Deckhouse Platform»;
- обеспечение доверенного маршрута;
- обеспечение доверенного канала.

Для реализации функций безопасности среды функционирования ПО «Deckhouse Platform» должны выполняться следующие действия:

- необходимо настроить SSH-доступ по ключу. Для этого необходимо подготовить и скопировать публичный ключ в каталог `"$HOME/.ssh/`
- отключить маршруты по умолчанию (default route) и оставить только маршрут (route) к registry и к bastion host для подключения по ssh. Для этого в файле /etc/rc.local прописать параметры, пример которых представлен ниже:

```
# tf.bastion
ip.ro add 95.217.68.252/32 via 192.168.199.1
# registry
ip.ro add 5.182.5.140/32 via 192.168.199.1
# cluster networks
ip.ro add 10.111.0.0/16 via 192.168.199.1
ip.ro add 10.222.0.0/16 via 192.168.199.1
# remove all routes
ip.ro add 5.182.5.140/32 via 192.168.199.1
ip ro del default
```

- необходимо регулярное обновление всех сред функционирования ПО «Deckhouse Platform» до актуальных версий с применением всех необходимых патчей безопасности с официальных сайтов разработчиков сред функционирования;
- установка, конфигурирование и управление ПО «Deckhouse Platform» должно осуществляться в соответствии с эксплуатационной документацией;
- доступ к объектам доступа ПО «Deckhouse Platform» должен осуществляться с учетом минимально необходимых прав и привилегий в соответствии с ролевой моделью ПО «Deckhouse Platform»;

- должна быть обеспечена физическая сохранность серверной платформы с установленным ПО «Deckhouse Platform» и исключение возможности физического доступа к ней посторонних лиц;
- ПО «Deckhouse Platform» должно использоваться только на совместимых с ним аппаратных мощностях и средствах;
- предоставление пользователям прав доступа к объектам доступа информационной системы обеспечивается, основываясь на задачах, решаемых пользователями в ПО «Deckhouse Platform» и взаимодействующими с ней информационными системами;
- каналы передачи данных ПО «Deckhouse Platform» должны быть либо расположены в пределах контролируемой зоны и защищены с использованием организационно-технических мер, либо, в случае их выхода за пределы контролируемой зоны, должны быть защищены путем применения средств криптографической защиты информации, сертифицированных в системе сертификации ФСБ России.

7 Модули и параметры, отвечающие за реализацию функций безопасности

7.1 Список модулей, отвечающих за реализацию функций безопасности

Список модулей, выключение которых не позволяет ПО «Deckhouse Platform» выполнять заявленные функции безопасности. Выключение указанных модулей недопустимо:

- admission-policy-engine
- control-plane-manager
- log-shipper
- loki
- operator-prometheus
- operator-trivy
- prometheus
- runtime-audit-engine
- user-authn
- user-authz
- gost-integrity-controller
- virtualization
- sds-node-configurator
- cni-cilium
- deckhouse
- ingress-nginx
- kube-dns
- node-manager
- registrypackages
- registry-packages-proxy
- registry
- common
- multitenancy-manager
- csi-nfs
- sds-local-volume
- csi-scsi-generic

7.2 Список параметров модулей, отвечающих за реализацию функций безопасности

Параметры модулей или ресурсы, изменение которых влияет на реализацию ПО «Deckhouse Platform» заявленных функций безопасности.

Модуль	**Параметры модуля или ресурс (объект)**	**Функция безопасности**	**Пояснения**
admission-policy-engine	- denyVulnerableImages.enabled - podSecurityStandards.enforcementAction	- запрет на создание образов контейнеров, содержащих известные уязвимости критического и высокого уровня опасности	- параметр podSecurityStandards.enforcementAction должен быть установлен в Deny (значение по умолчанию). - параметр denyVulnerableImages.enabled должен быть установлен в true
admission-policy-engine	- podSecurityStandards.defaultPolicy - podSecurityStandards.enforcementAction	- ограничение прав прикладного программного обеспечения, выполняемого внутри контейнера, на использование периферийных устройств, устройств хранения данных и съемных машинных носителей информации (блочных устройств), входящих в состав информационной (автоматизированной) системы - ограничение прав прикладного программного обеспечения, выполняемого внутри контейнера, на использование вычислительных ресурсов (оперативной памяти, операций ввода-вывода за период времени) хостовой операционной системы - монтирование корневой файловой системы хостовой операционной системы в режиме «только для чтения»	- параметр podSecurityStandards.enforcementAction должен быть установлен в Deny (значение по умолчанию). - параметр podSecurityStandards.defaultPolicy НЕ должен быть установлен в Privileged

Модуль	Параметры модуля или ресурс (объект)	Функция безопасности	Пояснения
loki	- storageClass - storeSystemLogs	- контроль целостности сведений о событиях безопасности самостоятельно - сбор и хранение записей в журнале событий безопасности, которые позволяют определить, когда и какие действия происходили	- параметр storageClass должен быть определен, если не определен глобальный параметр modules.storageClass. - параметр storeSystemLogs должен быть установлен в true
operator-trivy	- severities	- выявление известных уязвимостей при создании, установке образа контейнера в информационной (автоматизированной) системе и хранении образов контейнеров - оповещение о выявленных уязвимостях в образах контейнеров разработчика образов контейнеров и администратора безопасности информационной (автоматизированной) системы	- параметр severities должен содержать набор всех уровней критичности - UNKNOWN, LOW, MEDIUM, HIGH, CRITICAL (значение по умолчанию)
runtime-audit-engine, prometheus	CustomPrometheusRules	- контроль целостности образов контейнеров и исполняемых файлов контейнеров - контроль целостности параметров настройки ПО «Deckhouse Platform Certified Security Edition»	Должен быть создан объект CustomPrometheusRules, настраивающий отправку информации о событии аудита в систему мониторинга
gost-integrity-controller		- контроль целостности образов контейнеров и параметров настройки ПО «Deckhouse Platform Certified Security Edition» при установке образа	На объект Namespace, необходимого пространства имен, необходимо установить метку ***gost-integrity-controller.deckhouse.io/gost-digest-validation-enabled: true***, чтобы выполнялась функция безопасности

Модуль	Параметры модуля или ресурс (объект)	Функция безопасности	Пояснения
		контейнера в информационной (автоматизированной) системе и далее периодически за счет применения цифровой подписи самостоятельно	
cni-cilium	CiliumClusterWideNetworkPolicy	- фильтрация информационных потоков в соответствии с правилами управления потоками, установленными администратором безопасности; - разрешение передачи информации в ПО «Deckhouse Platform Certified Security Edition» только по маршруту, установленному администратором безопасности; - возможность записи во временное хранилище информации о сетевых взаимодействиях для анализа администратором безопасности	Конфигурационная опция policyAuditMode должна быть в значении false
cni-cilium	- EgressGateway - EgressGatewayPolicy	- изменение (перенаправление) маршрута передачи информации в случаях, установленных администратором безопасности	

Модуль	Параметры модуля или ресурс (объект)	Функция безопасности	Пояснения
control-plane-manager	apiserver.signature	- режим работы контроля целостности данных хранимых в etcd	Значение параметра apiserver.signature должно быть установлено следующим образом: - ‘Enforce’: для кластеров, развертываемых впервые, а также для существующих кластеров, у которых данные в etcd были полностью преобразованы в новый формат. - ‘Migrate’: для кластеров, в которых, на момент активации функции, данные в etcd присутствуют в стандартном формате. - ‘Rollback’: предназначен для кластеров, которым необходима обратная совместимость данных etcd и их резервных копий с классической версией Kubernetes (полное отключение функции безопасности). Данный режим также необходимо использовать при обновлении уже развернутого кластера Deckhouse до версии, в которой появилась функция контроля целостности данных etcd, перед переключением на любой другой режим. (значение по умолчанию)
virtualization	- .spec.enabled	- управление состоянием модуля	Значение параметра должно быть: - true, чтобы включить модуль; - false, чтобы выключить модуль.

7.3 Список событий, отвечающих за реализацию функций безопасности

В целях контроля целостности, аудита и предотвращения несанкционированных операций в среде функционирования ПО «Deckhouse Platform» фиксируются ключевые события безопасности, которые регистрируются в журнале событий безопасности Security Events в папке Security (п.5.6).

Список событий, регистрация которых обеспечивает реализацию заявленных функций безопасности:

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода
События безопасности средства контейнеризации		

Неуспешные попытки аутентификации пользователей средства контейнеризации	Событие:
	Неуспешная попытка аутентификации в веб-интерфейсе платформы
	
	Фильтр отбора событий:
	`failed login attempt: Invalid credentials`
		{
		"time": "2025-11-18T07:52:55.091407385Z",
		"level": "ERROR",
		"msg": "failed login attempt: Invalid credentials.",
		"user": "user@test.ru",
		"client_remote_addr": "192.168.0.246",
		"request_id": "1d80f05a-1601-410a-8195-cc9fdfee75bc"
		}

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода

	Событие:
	Неуспешная попытка аутентификации в сервисе kube-apiserver
	
	Фильтр отбора событий:
	`Unauthorized K8s API request detected`
		{
		"hostname": "cse-2-master-1",
		"output": "13:38:46.928533000: Warning Unauthorized K8s API request detected\nuser=<NA>\nagents=curl/8.14.1\nverb=get\nuri=/1\nsourceips=[\"192.168.0.246\",\"192.168.0.12\"]",
		"output_fields": {
		"evt.time": 1763473126928533000,
		"jevt.value[/requestURI]": "/1",
		"jevt.value[/sourceIPs]": "[\"192.168.0.246\",\"192.168.0.12\"]",
		"jevt.value[/user/username]": null,
		"jevt.value[/userAgent]": "curl/8.14.1",
		"jevt.value[/verb]": "get"
		},
		"priority": "Warning",
		"rule": "Unauthorized request in Kubernetes API",
		"source": "k8s_audit",
		"tags": [
		"k8s_auth_issues",
		"unauthorized"
		],
		"time": "2025-11-18T13:38:46.928533000Z"
		}

Запуск и остановка контейнеров с указанием причины остановки	Событие:
	Запуск контейнера
	
	Фильтр отбора событий:
	`K8s Pod Created`
		{
		"hostname": "cse-2-master-0",
		"output": "10:04:22.866760000: Informational K8s Pod Created (user=kubernetes-admin pod=test-pod0 ns=default resource=pods resp=201 decision=allow reason=RBAC: allowed by ClusterRoleBinding \"kubeadm:cluster-admins\" of ClusterRole \"cluster-admin\" to Group \"kubeadm:cluster-admins\")",
		"output_fields": {
		"evt.time": 1763546662866760000,
		"ka.auth.decision": "allow",
		"ka.auth.reason": "RBAC: allowed by ClusterRoleBinding \"kubeadm:cluster-admins\" of

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода
		ClusterRole \"cluster-admin\" to Group \"kubeadm:cluster-admins\"", "ka.response.code": "201", "ka.target.name": "test-pod0", "ka.target.namespace": "default", "ka.target.resource": "pods", "ka.user.name": "kubernetes-admin" }, "priority": "Informational", "rule": "K8s Pod created", "source": "k8s_audit", "tags": ["container_drift", "fstec"], "time": "2025-11-19T10:04:22.866760000Z" }

	<u>Событие:</u>
	Остановка (удаление) контейнера
	
	<u>Фильтр отбора событий:</u>
	`K8s Pod Deleted`
		{
		"hostname": "cse-2-master-2",
		"output": "10:12:49.236482000:
		Informational K8s Pod Deleted
		(user=system:node:cse-2-worker-2
		pod=test-pod0 ns=default resource=pods
		resp=200 decision=allow reason=)",
		"output_fields": {
		"evt.time": 1763547169236482000,
		"ka.auth.decision": "allow",
		"ka.auth.reason": "",
		"ka.response.code": "200",
		"ka.target.name": "test-pod0",
		"ka.target.namespace": "default",
		"ka.target.resource": "pods",
		"ka.user.name":
		"system:node:cse-2-worker-2"
		},
		"priority": "Informational",
		"rule": "K8s Pod deleted",
		"source": "k8s_audit",
		"tags": [
		"container_drift",
		"fstec"
		],
		"time":
		"2025-11-19T10:12:49.236482000Z"
		}

Модификация запускаемых контейнеров	<u>Событие:</u> В контейнере запущен процесс управления пакетами	{ "hostname": "cse-2-worker-1", "output": "12:02:43.366871682: Error Package management process launched in

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода

	Фильтр отбора событий:
	`Package management process launched in container`
		`container (user=root user_loginuid=-1 command=apt pid=772663 container_id=9d8dd712bf1b container_name=test123 image=docker.io/library/ubuntu:latest exe_flags=EXE_WRITABLE|EXE_LOWER_LAYER)"`
		`,`
		`"output_fields": {`
		`"container.id": "9d8dd712bf1b",`
		`"container.image.repository": "docker.io/library/ubuntu",`
		`"container.image.tag": "latest",`
		`"container.name": "test123",`
		`"evt.arg.flags": "EXE_WRITABLE|EXE_LOWER_LAYER",`
		`"evt.time": 1763553763366871800,`
		`"proc.cmdline": "apt",`
		`"proc.pid": 772663,`
		`"user.loginuid": -1,`
		`"user.name": "root"`
		`},`
		`"priority": "Error",`
		`"rule": "Launch Package Management Process in container",`
		`"source": "syscall",`
		`"tags": [`
		`"container_drift",`
		`"fstec"`
		`],`
		`"time": "2025-11-19T12:02:43.366871682Z"`
		`}`

	Событие:
	Исполнение бинарного файла, не входящего в базовый образ контейнера. Шаблон «drop and execute» часто наблюдается после получения злоумышленником первоначального доступа.
	
	Фильтр отбора событий:
	`Executing binary not part of base image`
		`{`
		`"hostname": "cse-2-worker-1",`
		`"output": "12:20:35.191327872: Critical Executing binary not part of base image (user=root user_loginuid=-1 user_uid=0 comm=curl ya.ru exe=curl container_id=9d8dd712bf1b image=docker.io/library/ubuntu proc.name=curl proc.sname=bash proc.pname=bash proc.aname[2]=containerd-shim exe_flags=EXE_WRITABLE|EXE_UPPER_LAYER proc.exe_ino=1580515 proc.exe_ino.ctime=1763554824302013490 proc.exe_ino.mtime=1733935459000000000 proc.exe_ino.ctime_duration_proc_start=10886267891 proc.exepath=/usr/bin/curl proc.cwd=/ proc.tty=34817 container.start_ts=1763553758811466336 proc.sid=15 proc.vpgid=3161`

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода

		evt.res=SUCCESS)",
		"output_fields": {
		"container.id": "9d8dd712bf1b",
		"container.image.repository":
		"docker.io/library/ubuntu",
		"container.start_ts":
		1763553758811466200,
		"evt.arg.flags":
		"EXE_WRITABLE|EXE_UPPER_LAYER",
		"evt.res": "SUCCESS",
		"evt.time": 1763554835191327700,
		"proc.aname[2]": "containerd-shim",
		"proc.cmdline": "curl ya.ru",
		"proc.cwd": "/",
		"proc.exe": "curl",
		"proc.exe_ino": 1580515,
		"proc.exe_ino.ctime":
		1763554824302013400,
		
		"proc.exe_ino.ctime_duration_proc_start"
		: 10886267891,
		"proc.exe_ino.mtime":
		1733935459000000000,
		"proc.exepath": "/usr/bin/curl",
		"proc.name": "curl",
		"proc.pname": "bash",
		"proc.sid": 15,
		"proc.sname": "bash",
		"proc.tty": 34817,
		"proc.vpgid": 3161,
		"user.loginuid": -1,
		"user.name": "root",
		"user.uid": 0
		},
		"priority": "Critical",
		"rule": "Drop and execute new binary
		in container",
		"source": "syscall",
		"tags": [
		"container_drift",
		"fstec"
		],
		"time":
		"2025-11-19T12:20:35.191327872Z"
		}

	Событие: В контейнере создан новый исполняемый файл путём изменения прав (chmod) Фильтр отбора событий:	{ "hostname": "cse-2-worker-1", "output": "12:39:43.680836047: Error Drift detected (chmod), new executable created in a container (user=root user_loginuid=-1 command=chmod +x 1.sh pid=973196 filename=/1.sh name=<NA>

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода

	Drift detected \\(chmod\\)	mode=S_IXOTH|S_IROTH|S_IXGRP|S_IRGRP|S_IXUSR|S_IWUSR|S_IRUSR event=fchmodat)",
		"output_fields": {
		"evt.arg.filename": "/1.sh",
		"evt.arg.mode": "S_IXOTH|S_IROTH|S_IXGRP|S_IRGRP|S_IXUSR|S_IWUSR|S_IRUSR",
		"evt.arg.name": null,
		"evt.time": 1763555983680836000,
		"evt.type": "fchmodat",
		"proc.cmdline": "chmod +x 1.sh",
		"proc.pid": 973196,
		"user.loginuid": -1,
		"user.name": "root"
		},
		"priority": "Error",
		"rule": "Container drift detected (chmod)",
		"source": "syscall",
		"tags": [
		"container_drift",
		"fstec"
		],
		"time": "2025-11-19T12:39:43.680836047Z"
		}

	Событие:
	В контейнере создан новый исполняемый файл посредством операций open+create.
	Поведение часто используется вредоносным программным обеспечением.
	
	Фильтр отбора событий:
	Drift detected \\(open\\+create\\)
		{
		"hostname": "cse-2-worker-1",
		"output": "13:15:03.343713276: Error Drift detected (open+create), new executable created in a container (user=root user_loginuid=-1 command=malw pid=1161187 filename=<NA> name=/tmp/drift_test mode=0755 event=openat)",
		"output_fields": {
		"evt.arg.filename": null,
		"evt.arg.mode": "0755",
		"evt.arg.name": "/tmp/drift_test",
		"evt.time": 1763558103343713300,
		"evt.type": "openat",
		"proc.cmdline": "malw",
		"proc.pid": 1161187,
		"user.loginuid": -1,
		"user.name": "root"
		},
		"priority": "Error",
		"rule": "Container drift detected (open+create)",
		"source": "syscall",
		"tags": [
		"container_drift",

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода
		"fstec"], "time": "2025-11-19T13:15:03.343713276Z" }

	Событие:
	Попытка изменить любой файл в пределах набора системных каталогов с бинарными файлами.
	
	Фильтр отбора событий:
	`File below known binary directory renamed/removed`
		{
		"hostname": "cse-2-worker-2",
		"output": "13:46:03.506551797: Error File below known binary directory renamed/removed (user=root user_loginuid=-1 command=ip6tables -t filter -S OLD_CILIUM_FORWARD pid=1204837 pcmdline=cilium-agent --config-dir=/tmp/cilium/config-map --prometheus-serve-addr=127.0.0.1:9092 operation=unlinkat file=<NA> res=-30(EROFS) dirfd=3(<f>/usr/sbin) name=iptables(/usr/sbin/iptables) flags=0 container_id=a85dcc476525 image=dev-registry-cse.deckhouse.ru/sys/deckhouse-cse)",
		"output_fields": {
		"container.id": "a85dcc476525",
		"container.image.repository": "dev-registry-cse.deckhouse.ru/sys/deckhouse-cse",
		"evt.args": "res=-30(EROFS) dirfd=3(<f>/usr/sbin) name=iptables(/usr/sbin/iptables) flags=0",
		"evt.time": 1763559963506551800,
		"evt.type": "unlinkat",
		"fd.name": null,
		"proc.cmdline": "ip6tables -t filter -S OLD_CILIUM_FORWARD",
		"proc.pcmdline": "cilium-agent --config-dir=/tmp/cilium/config-map --prometheus-serve-addr=127.0.0.1:9092",
		"proc.pid": 1204837,
		"user.loginuid": -1,
		"user.name": "root"
		},
		"priority": "Error",
		"rule": "Modify binary dirs",
		"source": "syscall",
		"tags": [
		"container_drift",
		"fstec"
		],
		"time":
		"2025-11-19T13:46:03.506551797Z"
		}

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода

	Событие:
	Попытка *attach/exec* в под.
	
	Фильтр отбора событий:
	`Attach/Exec to pod`
		{
		"hostname": "cse-2-master-0",
		"output": "06:16:12.365352000: Notice Attach/Exec to pod (user=kubernetes-admin pod=test123 resource=pods ns=default action=exec command=bash)",
		"output_fields": {
		"evt.time": 1763619372365352000,
		"ka.target.name": "test123",
		"ka.target.namespace": "default",
		"ka.target.resource": "pods",
		"ka.target.subresource": "exec",
		"ka.uri.param[command]": "bash",
		"ka.user.name": "kubernetes-admin"
		},
		"priority": "Notice",
		"rule": "Attach/Exec Pod",
		"source": "k8s_audit",
		"tags": [
		"container_image_access",
		"fstec"
		],
		"time": "2025-11-20T06:16:12.365352000Z"
		}

	Событие:
	Создание временного (ephemeral) контейнера.
	
	Фильтр отбора событий:
	`Ephemeral container is created in pod`
		{
		"hostname": "cse-2-master-0",
		"output": "06:27:03.543708000: Notice Ephemeral container is created in pod (user=kubernetes-admin pod=test123 resource=pods ns=default ephemeral_container_name=debugger-vrjf4 ephemeral_container_image=busybox)",
		"output_fields": {
		"evt.time": 1763620023543708000,
		
		"jevt.value[/requestObject/spec/ephemeralContainers/0/image]": "busybox",
		
		"jevt.value[/requestObject/spec/ephemeralContainers/0/name]": "debugger-vrjf4",
		"ka.target.name": "test123",
		"ka.target.namespace": "default",
		"ka.target.resource": "pods",
		"ka.user.name": "kubernetes-admin"
		},
		"priority": "Notice",
		"rule": "EphemeralContainers created",
		"source": "k8s_audit",
		"tags": [
		"container_image_access",

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода
		"fstec"], "time": "2025-11-20T06:27:03.543708000Z" }

	Событие:
	Попытка изменить файл образа контейнера в каталоге */var/lib/containerd/containerd*.
	
	Фильтр отбора событий:
	`File below a known containerd directory opened for writing`
		{
		"hostname": "cse-2-master-0",
		"output": "06:31:57.318994093: Error File below a known containerd directory opened for writing (user=root user_loginuid=1001 command=tee /var/lib/containerd/testfile pid=853499 file=/var/lib/containerd/testfile parent=sudo pcmdline=sudo tee /var/lib/containerd/testfile gparent=sudo container_id=host image=)",
		"output_fields": {
		"container.id": "host",
		"container.image.repository": "",
		"evt.time": 1763620317318994200,
		"fd.name":
		"/var/lib/containerd/testfile",
		"proc.aname[2]": "sudo",
		"proc.cmdline": "tee /var/lib/containerd/testfile",
		"proc.pcmdline": "sudo tee /var/lib/containerd/testfile",
		"proc.pid": 853499,
		"proc.pname": "sudo",
		"user.loginuid": 1001,
		"user.name": "root"
		},
		"priority": "Error",
		"rule": "Write below containerd images dir",
		"source": "syscall",
		"tags": [
		"container_image_drift",
		"fstec"
		],
		"time":
		"2025-11-20T06:31:57.318994093Z"
		}

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода

	Событие:
	Попытка прочесть файл образа контейнера в каталоге */var/lib/containerd/io.containerd.grpc.v1.cri/containers/*
	
	Фильтр отбора событий:
	`File below a known containerd directory opened for reading`
		{
		"hostname": "cse-2-worker-2",
		"output": "07:05:22.961388994: Notice File below a known containerd directory opened for reading (user=root user_loginuid=-1 command=containerd pid=708001 file=/var/lib/containerd/io.containerd.grpc.v1.cri/containers/7844711587a95f798f1a4ca5d0d9f86b5a98585e594b1cfc9ce9b60b06629891/.tmp-status209272341 parent=systemd pcmdline=systemd gparent=<NA> container_id=host image=)",
		"output_fields": {
		"container.id": "host",
		"container.image.repository": "",
		"evt.time": 1763622322961389000,
		"fd.name": "/var/lib/containerd/io.containerd.grpc.v1.cri/containers/7844711587a95f798f1a4ca5d0d9f86b5a98585e594b1cfc9ce9b60b06629891/.tmp-status209272341",
		"proc.aname[2]": null,
		"proc.cmdline": "containerd",
		"proc.pcmdline": "systemd",
		"proc.pid": 708001,
		"proc.pname": "systemd",
		"user.loginuid": -1,
		"user.name": "root"
		},
		"priority": "Notice",
		"rule": "Read below containerd images dir",
		"source": "syscall",
		"tags": [
		"container_image_access",
		"fstec"
		],
		"time": "2025-11-20T07:05:22.961388994Z"
		}

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода

	Событие:
	Использование в контейнере, запущенном в системном пространстве имён, тега образа, отличного от sha256-суммы. Свидетельствует о потенциальной ошибке настройки контроля целостности.
	
	Фильтр отбора событий:
	`Not all containers are running with the sha256`
		{
		"hostname": "cse-2-master-0",
		"output": "07:28:31.005030000: Notice Not all containers are running with the sha256 sum as a tag in a system namespace, which is a potential integrity control mechanism misconfiguration (user=kubernetes-admin binding=test resource=pods resp=201 decision=allow reason=RBAC: allowed by ClusterRoleBinding \"kubeadm:cluster-admins\" of ClusterRole \"cluster-admin\" to Group \"kubeadm:cluster-admins\" image=(nginx))",
		"output_fields": {
		"evt.time": 1763623711005030000,
		"ka.auth.decision": "allow",
		"ka.auth.reason": "RBAC: allowed by ClusterRoleBinding \"kubeadm:cluster-admins\" of ClusterRole \"cluster-admin\" to Group \"kubeadm:cluster-admins\"",
		"ka.req.pod.containers.image": [
		"nginx"
		],
		"ka.response.code": "201",
		"ka.target.name": "test",
		"ka.target.resource": "pods",
		"ka.user.name": "kubernetes-admin"
		},
		"priority": "Notice",
		"rule": "Container tag is not @sha256",
		"source": "k8s_audit",
		"tags": [
		"fstec",
		"integrity_control"
		],
		"time": "2025-11-20T07:28:31.005030000Z"
		}

Изменение ролевой модели	Событие:
	Попытка создания ServiceAccount в пространствах имён *kube-system*, *kube-public, default*
	
	Фильтр отбора событий:
		{
		"hostname": "cse-2-master-0",
		"output": "07:49:48.803021000: Warning Service account created in kube namespace (user=kubernetes-admin serviceaccount=test resource=serviceaccounts ns=kube-system)",
		"output_fields": {

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода
	`Service account created in kube namespace`	`"evt.time": 1763624988803021000, "ka.target.name": "test", "ka.target.namespace": "kube-system", "ka.target.resource": "serviceaccounts", "ka.user.name": "kubernetes-admin" }, "priority": "Warning", "rule": "ServiceAccount created in a system namespace", "source": "k8s_audit", "tags": ["fstec", "rbac_drift"], "time": "2025-11-20T07:49:48.803021000Z" }`
	<u>Событие:</u> Попытка создания *ClusterRoleBinding* к роли *cluster-admin*. <u>Фильтр отбора событий:</u> `Cluster Role Binding to cluster-admin role`	`{ "hostname": "cse-2-master-2", "output": "08:41:27.516128000: Warning Cluster Role Binding to cluster-admin role (user=kubernetes-admin binding=test444 resource=clusterrolebindings subjects=<NA> role=cluster-admin resp=201 decision=allow reason=RBAC: allowed by ClusterRoleBinding \"kubeadm:cluster-admins\" of ClusterRole \"cluster-admin\" to Group \"kubeadm:cluster-admins\")", "output_fields": { "evt.time": 1763628087516128000, "ka.auth.decision": "allow", "ka.auth.reason": "RBAC: allowed by ClusterRoleBinding \"kubeadm:cluster-admins\" of ClusterRole \"cluster-admin\" to Group \"kubeadm:cluster-admins\"", "ka.req.binding.role": "cluster-admin", "ka.req.binding.subjects": null, "ka.response.code": "201", "ka.target.name": "test444", "ka.target.resource": "clusterrolebindings", "ka.user.name": "kubernetes-admin" }, "priority": "Warning", "rule": "Attach to cluster-admin Role",`

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода
		"source": "k8s_audit", "tags": ["fstec", "rbac_drift"], "time": "2025-11-20T08:41:27.516128000Z" }

	<u>Событие:</u>
	Попытка создания *Role/ClusterRole* с подстановочными знаками («*») в ресурсах или действиях.
	
	<u>Фильтр отбора событий:</u>
	`Created Role/ClusterRole with wildcard`
		{
		"hostname": "cse-2-master-0",
		"output": "09:08:25.846971000: Warning Created Role/ClusterRole with wildcard (user=kubernetes-admin role=user-editor resource=clusterroles rules=({\"verbs\":[\"get\"],\"apiGroups\":[\"kuma.io\"],\"resources\":[\"*\"]}))",
		"output_fields": {
		"evt.time": 1763629705846971000,
		"ka.req.role.rules": [
		"{\"verbs\":[\"get\"],\"apiGroups\":[\"kuma.io\"],\"resources\":[\"*\"]}"
		],
		"ka.target.name": "user-editor",
		"ka.target.resource": "clusterroles",
		"ka.user.name": "kubernetes-admin"
		},
		"priority": "Warning",
		"rule": "ClusterRole with wildcard created",
		"source": "k8s_audit",
		"tags": [
		"fstec",
		"rbac_drift"
		],
		"time":
		"2025-11-20T09:08:25.846971000Z"
		}

	<u>Событие:</u>
	Попытка создания *Role/ClusterRole*, обладающую правами на операции записи(*create, modify, delete*).
	
	<u>Фильтр отбора событий:</u>
	`Created Role/ClusterRole with write privileges`
		{
		"hostname": "cse-2-master-0",
		"output": "09:28:35.509262000: Notice Created Role/ClusterRole with write privileges (user=kubernetes-admin role=user-editor resource=clusterroles rules=({\"verbs\":[\"get\",\"create\"],\"apiGroups\":[\"kuma.io\"],\"resources\":[\"*\"]}))",
		"output_fields": {
		"evt.time": 1763630915509262000,

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода
		"ka.req.role.rules": ["{\"verbs\":[\"get\",\"create\"],\"apiGroups\":[\"kuma.io\"],\"resources\":[\"*\"]}"], "ka.target.name": "user-editor", "ka.target.resource": "clusterroles", "ka.user.name": "kubernetes-admin" }, "priority": "Notice", "rule": "ClusterRole with write privileges created", "source": "k8s_audit", "tags": ["fstec", "rbac_drift"], "time": "2025-11-20T09:28:35.509262000Z" }

	Событие:
	Попытка создания *Role/ClusterRole* с возможностью выполнения команд (exec) в поде (с объектом *pods/exec* в массиве *resources*).
	
	Фильтр отбора событий:
	`Created Role/ClusterRole with pod exec privileges`
		{
		"hostname": "cse-2-master-0",
		"output": "09:34:10.109024000: Warning Created Role/ClusterRole with pod exec privileges (user=kubernetes-admin role=user-editor resource=clusterroles rules=({\"verbs\":[\"get\",\"create\"],\"apiGroups\":[\"kuma.io\"],\"resources\":[\"test\",\"pods/exec\"]}))",
		"output_fields": {
		"evt.time": 1763631250109024000,
		"ka.req.role.rules": [
		"{\"verbs\":[\"get\",\"create\"],\"apiGroups\":[\"kuma.io\"],\"resources\":[\"test\",\"pods/exec\"]}"
		],
		"ka.target.name": "user-editor",
		"ka.target.resource": "clusterroles",
		"ka.user.name": "kubernetes-admin"
		},
		"priority": "Warning",
		"rule": "ClusterRole with Pod Exec created",
		"source": "k8s_audit",
		"tags": [
		"fstec",
		"rbac_drift"
		],

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода
		"time": "2025-11-20T09:34:10.109024000Z" }
	Событие: Попытка изменить или удалить *ClusterRole/Role*, название которой начинается с *system*. Фильтр отбора событий: `System ClusterRole/Role modified or deleted`	{ "hostname": "cse-2-master-0", "output": "09:39:25.003993000: Warning System ClusterRole/Role modified or deleted (user=kubernetes-admin role=system:test resource=clusterroles action=delete)", "output_fields": { "evt.time": 1763631565003993000, "ka.target.name": "system:test", "ka.target.resource": "clusterroles", "ka.user.name": "kubernetes-admin", "ka.verb": "delete" }, "priority": "Warning", "rule": "System ClusterRole modified/deleted", "source": "k8s_audit", "tags": ["fstec", "rbac_drift"], "time": "2025-11-20T09:39:25.003993000Z" }
	Событие: Попытка создания ServiceAccount Фильтр отбора событий: `K8s Serviceaccount Created`	{ "hostname": "cse-2-master-1", "output": "09:42:26.177759000: Notice K8s Serviceaccount Created (user=system:node:cse-2-worker-1 serviceaccount=cert-manager ns=d8-cert-manager resource=serviceaccounts resp=201 decision=allow reason=)", "output_fields": { "evt.time": 1763631746177759000, "ka.auth.decision": "allow", "ka.auth.reason": "", "ka.response.code": "201", "ka.target.name": "cert-manager", "ka.target.namespace": "d8-cert-manager", "ka.target.resource": "serviceaccounts", "ka.user.name": "system:node:cse-2-worker-1" },

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода
		"priority": "Notice", "rule": "K8s ServiceAccount created", "source": "k8s_audit", "tags": ["fstec", "rbac_drift"], "time": "2025-11-20T09:42:26.177759000Z" }

	<u>Событие:</u>
	Попытка удаления ServiceAccount
	
	<u>Фильтр отбора событий:</u>
	`K8s Serviceaccount Deleted`
		{
		"hostname": "cse-2-master-0",
		"output": "09:44:58.733934000: Notice K8s Serviceaccount Deleted (user=kubernetes-admin serviceaccount=test ns=default resource=serviceaccounts resp=200 decision=allow reason=RBAC: allowed by ClusterRoleBinding \"kubeadm:cluster-admins\" of ClusterRole \"cluster-admin\" to Group \"kubeadm:cluster-admins\")",
		"output_fields": {
		"evt.time": 1763631898733934000,
		"ka.auth.decision": "allow",
		"ka.auth.reason": "RBAC: allowed by ClusterRoleBinding \"kubeadm:cluster-admins\" of ClusterRole \"cluster-admin\" to Group \"kubeadm:cluster-admins\"",
		"ka.response.code": "200",
		"ka.target.name": "test",
		"ka.target.namespace": "default",
		"ka.target.resource":
		"serviceaccounts",
		"ka.user.name": "kubernetes-admin"
		},
		"priority": "Notice",
		"rule": "K8s ServiceAccount deleted",
		"source": "k8s_audit",
		"tags": [
		"fstec",
		"rbac_drift"
		],
		"time":
		"2025-11-20T09:44:58.733934000Z"
		}

	<u>Событие:</u> Попытка создания *Role* или *ClusterRole*	{ "hostname": "cse-2-master-0", "output": "09:39:07.355383000: Notice K8s Cluster Role Created

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода

	Фильтр отбора событий:
	`K8s Cluster Role Created`
		`(user=kubernetes-admin role=system:test resource=clusterroles rules=({\"verbs\":[\"get\"],\"apiGroups\":[\"\"],\"resources\":[\"*\"]}) resp=201 decision=allow reason=RBAC: allowed by ClusterRoleBinding \"kubeadm:cluster-admins\" of ClusterRole \"cluster-admin\" to Group \"kubeadm:cluster-admins\")",
		"output_fields": {
		"evt.time": 1763631547355383000,
		"ka.auth.decision": "allow",
		"ka.auth.reason": "RBAC: allowed by ClusterRoleBinding \"kubeadm:cluster-admins\" of ClusterRole \"cluster-admin\" to Group \"kubeadm:cluster-admins\"",
		"ka.req.role.rules": [
		"{\"verbs\":[\"get\"],\"apiGroups\":[\"\"],\"resources\":[\"*\"]}"
		],
		"ka.response.code": "201",
		"ka.target.name": "system:test",
		"ka.target.resource": "clusterroles",
		"ka.user.name": "kubernetes-admin"
		},
		"priority": "Notice",
		"rule": "K8s Role/ClusterRole created",
		"source": "k8s_audit",
		"tags": [
		"fstec",
		"rbac_drift"
		],
		"time": "2025-11-20T09:39:07.355383000Z"
		}

	Событие:
	Попытка создания *ClusterRoleBinding*
	
	Фильтр отбора событий:
	`K8s Cluster Role Binding Created`
		{
		"hostname": "cse-2-master-1",
		"output": "10:08:06.223077000: Notice K8s Cluster Role Binding Created (user=kubernetes-admin binding=test444 resource=clusterrolebindings subjects=<NA> role=admins resp=201 decision=allow reason=RBAC: allowed by ClusterRoleBinding \"kubeadm:cluster-admins\" of ClusterRole \"cluster-admin\" to Group \"kubeadm:cluster-admins\")",
		"output_fields": {

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода
		"evt.time": 1763633286223077000, "ka.auth.decision": "allow", "ka.auth.reason": "RBAC: allowed by ClusterRoleBinding \"kubeadm:cluster-admins\" of ClusterRole \"cluster-admin\" to Group \"kubeadm:cluster-admins\"", "ka.req.binding.role": "admins", "ka.req.binding.subjects": null, "ka.response.code": "201", "ka.target.name": "test444", "ka.target.resource": "clusterrolebindings", "ka.user.name": "kubernetes-admin" }, "priority": "Notice", "rule": "K8s Role/ClusterRole binding created", "source": "k8s_audit", "tags": ["fstec", "rbac_drift"], "time": "2025-11-20T10:08:06.223077000Z" }

	<u>Событие:</u>
	Попытка удаления *ClusterRoleBinding*
	
	<u>Фильтр отбора событий:</u>
	`K8s Cluster Role Binding Deleted`
		{
		"hostname": "cse-2-master-2",
		"output": "10:08:04.368957000: Notice K8s Cluster Role Binding Deleted (user=kubernetes-admin binding=test444 resource=clusterrolebindings resp=200 decision=allow reason=RBAC: allowed by ClusterRoleBinding \"kubeadm:cluster-admins\" of ClusterRole \"cluster-admin\" to Group \"kubeadm:cluster-admins\")",
		"output_fields": {
		"evt.time": 1763633284368957000,
		"ka.auth.decision": "allow",
		"ka.auth.reason": "RBAC: allowed by ClusterRoleBinding \"kubeadm:cluster-admins\" of ClusterRole \"cluster-admin\" to Group \"kubeadm:cluster-admins\"",
		"ka.response.code": "200",
		"ka.target.name": "test444",
		"ka.target.resource": "clusterrolebindings",
		"ka.user.name": "kubernetes-admin"
		},
		"priority": "Notice",

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода
		"rule": "K8s Role/ClusterRole binding deleted", "source": "k8s_audit", "tags": ["fstec", "rbac_drift"], "time": "2025-11-20T10:08:04.368957000Z" }
Выявление известных уязвимостей в образах контейнеров и некорректности конфигурации	Событие: Создание ресурсов *ConfigAuditReports* и *VulnerabilityReports* Фильтр отбора событий: `K8s Security Reports Created`	{ "hostname": "cse-2-master-2", "output": "12:15:07.995425000: Notice K8s Security Reports Created. The report may contain vulnerability information. Check the object in the cluster (user=system:serviceaccount:d8-operator-trivy:operator-trivy, verb=create, ns=cve, resource=vulnerabilityreports, object=replicaset-loki-deployment-5549c44655-loki)", "output_fields": { "evt.time": 1763640907995425000, "ka.target.name": "replicaset-loki-deployment-5549c44655-loki", "ka.target.namespace": "cve", "ka.target.resource": "vulnerabilityreports", "ka.user.name": "system:serviceaccount:d8-operator-trivy:operator-trivy", "ka.verb": "create" }, "priority": "Notice", "rule": "Create Security Reports", "source": "k8s_audit", "tags": ["fstec", "security_reports"], "time": "2025-11-20T12:15:07.995425000Z" }
Факты нарушения целостности объектов контроля	Событие: Один из двоичных файлов, используемых компонентами ПО «Deckhouse Platform» в	{ "hostname": "cse-2-master-0", "output": "13:55:53.182087669: Error Deckhouse binary /opt/deckhouse/bin/ls is missing a proper digital signature", "output_fields": {

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода

	директории */opt/deckhouse/bin*, не прошёл проверку подписи
	
	Фильтр отбора событий:
	`Error Deckhouse binary`
		`"evt.time": 1763646953182087700,`
		`"fd.name": "/opt/deckhouse/bin/ls"`
		`},`
		`"priority": "Error",`
		`"rule": "missing_digital_signature",`
		`"source": "syscall",`
		`"tags": [`
		`"integrity_check",`
		`"subscription_check"`
		`],`
		`"time": "2025-11-20T13:55:53.182087669Z"`
		`}`

	Событие: Нарушение целостности объектов хранимых в etcd Фильтр отбора событий: `K8s audit event with deckhouse.io/signature annotation`	`{` `"hostname": "adyakonov-static-master-0",` `"output": "08:28:46.868800000: Notice [Falco] K8s audit event with deckhouse.io/signature annotation. Indicates that unsigned objects or objects with invalid signatures were found in the etcd database. (user=system:node:adyakonov-static-master-0 verb=create resource=serviceaccounts object=safe-agent-updater signature=Absent signature)",` `"output_fields": {` `"evt.time": 1763713726868800000,` `"jevt.value[/annotations/deckhouse.io~1signature]": "Absent signature",` `"ka.target.name": "safe-agent-updater",` `"ka.target.resource": "serviceaccounts",` `"ka.user.name": "system:node:adyakonov-static-master-0",` `"ka.verb": "create"` `},` `"priority": "Notice",` `"rule": "Event with deckhouse.io/signature annotation",` `"source": "k8s_audit",` `"tags": [` `"annotations",` `"audit",` `"deckhouse",` `"signature"` `],` `"time": "2025-11-21T08:28:46.868800000Z"` `}`

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода

	Событие:
	Попытка запуска контейнера с некорректной или отсутствующей контрольной суммой образа
	
	Фильтр отбора событий:
	`K8s Pod creation denied due to missing GOST digest`
		{
		"hostname": "adyakonov-static-master-0",
		"output": "09:02:17.183187000: Warning K8s Pod creation denied due to missing GOST digest (user=kubernetes-admin, ns=default, pod=po, reason=admission webhook \"gost-digest-webhook.deckhouse.io\" denied the request: the image does not contain gost digest, image=nginx:latest)",
		"output_fields": {
		"evt.time": 1763715737183187000,
		
		"jevt.value[/requestObject/spec/containers/0/image]": "nginx:latest",
		
		"jevt.value[/responseStatus/message]": "admission webhook \"gost-digest-webhook.deckhouse.io\" denied the request: the image does not contain gost digest",
		"ka.target.name": "po",
		"ka.target.namespace": "default",
		"ka.user.name": "kubernetes-admin"
		},
		"priority": "Warning",
		"rule": "Pod creation denied due to missing GOST digest",
		"source": "k8s_audit",
		"tags": [
		"integrity_check",
		"signature"
		],
		"time": "2025-11-21T09:02:17.183187000Z"
		}

События безопасности средства виртуализации		

Успешные и неуспешные попытки аутентификации пользователей ПО «Deckhouse Platform»	Событие:
	Неуспешная попытка аутентификации в веб-интерфейсе платформы
	
	Фильтр отбора событий:
	`failed login attempt`
		{
		"time": "2025-11-18T07:52:55.091407385Z",
		"level": "ERROR",
		"msg": "failed login attempt: Invalid credentials.",
		"user": "user@test.ru",
		"client_remote_addr": "192.168.0.246",
		"request_id": "1d80f05a-1601-410a-8195-cc9fdfee75bc"
		}

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода

	Событие:
	Успешная попытка аутентификации в веб-интерфейсе платформы
	
	Фильтр отбора событий:
	`login successful`
		{
		"time": "2025-11-21T09:57:28.066435654Z",
		"level": "INFO",
		"msg": "login successful",
		"connector_id": "local",
		"username": "user",
		"preferred_username": "",
		"email": "name.surname@test.ru",
		"groups": null,
		"client_remote_addr": "10.111.8.85",
		"request_id": "5644ed82-b5f5-453c-ad51-661bd73efdfc"
		}

Доступ пользователей ПО «Deckhouse Platform» к виртуальным машинам	Событие:
	Доступ пользователей к ВМ
	
	Фильтр отбора событий:
	`Access to VM`
		{
		"type": "Access to VM",
		"level": "info",
		"name": "Virtual machine 'test-vm' connection has been initiated via console by 'name.surname@test.ru'",
		"datetime": "2025-11-22T07:43:43Z",
		"uid": "a83162e5-1c85-4c57-9c22-ea508c75115e",
		"request_subject": "name.surname@test.ru",
		"action_type": "get",
		"node_network_address": "10.12.0.200",
		"virtualmachine_uid": "986d6e76-ec0d-43ab-a846-171d1ecae437",
		"virtualmachine_os": "unknown",
		"storageclasses": "local-storage-class",
		"qemu_version": "v9.2.0",
		"libvirt_version": "v10.9.0",
		"operation_result": "unknown"
		}

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода

Создание и удаление виртуальных машин	Событие:
	Создание ВМ
	
	Фильтр отбора событий:
	`Virtual machine .* has been created`
		{
		"type": "Manage VM",
		"level": "info",
		"name": "Virtual machine 'test-vm' has been created by 'kubernetes-admin'",
		"datetime": "2025-11-22T07:41:04Z",
		"uid": "dd7d9739-2980-4fa0-b21b-1967fff6bbc3",
		"request_subject": "kubernetes-admin",
		"action_type": "create",
		"node_network_address": "unknown",
		"virtualmachine_uid": "986d6e76-ec0d-43ab-a846-171d1ecae437",
		"virtualmachine_os": "unknown",
		"storageclasses": "",
		"qemu_version": "",
		"libvirt_version": "",
		"operation_result": "allow"
		}

	Событие:
	Удаление ВМ
	
	Фильтр отбора событий:
	`Virtual machine .* has been deleted`
		{
		"type": "Manage VM",
		"level": "warn",
		"name": "Virtual machine 'test-vm' has been deleted by 'kubernetes-admin'",
		"datetime": "2025-11-22T07:28:29Z",
		"uid": "45410e58-7bf0-4b13-87d2-b5b8b8f5f190",
		"request_subject": "kubernetes-admin",
		"action_type": "delete",
		"node_network_address": "10.12.0.200",
		"virtualmachine_uid": "f08d1586-ef97-44bd-bb13-f2b329bf43b3",
		"virtualmachine_os": "unknown",
		"storageclasses": "unknown",
		"qemu_version": "v9.2.0",
		"libvirt_version": "v10.9.0",
		"operation_result": "allow"
		}

Запуск и остановка ПО «Deckhouse Platform» с указанием причины остановки	Событие: Запуск\изменение настроек СВ Фильтр отбора событий: `Module 'virtualization' has been updated`	{ "type": "Module control", "level": "info", "name": "Module 'virtualization' has been updated by 'kubernetes-admin'", "datetime": "2025-11-24T09:10:29Z", "uid": "10075456-a04c-4fd0-8e38-60e1a62f81a3", "request_subject": "kubernetes-admin", "operation_result": "allow", "action_type": "patch", "component": "virtualization",

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода
		"node_network_address": "unknown", "virtualization_version": "v1.73.2", "virtualization_name": "Deckhouse Virtualization Platform", "qemu_version": "unknown", "libvirt_version": "unknown" }

	Событие:
	Остановка СВ
	
	Фильтр отбора событий:
	`Module 'virtualization' has been disabled`
		{
		"type": "Module control",
		"level": "warn",
		"name": "Module 'virtualization' has been disabled by 'kubernetes-admin'",
		"datetime": "2025-11-24T09:10:26Z",
		"uid": "be3259e5-e986-4ea5-b15f-36322fd0d257",
		"request_subject": "kubernetes-admin",
		"operation_result": "allow",
		"action_type": "patch",
		"component": "virtualization",
		"node_network_address": "unknown",
		"virtualization_version": "v1.73.2",
		"virtualization_name": "Deckhouse Virtualization Platform",
		"qemu_version": "unknown",
		"libvirt_version": "unknown"
		}

Запуск и остановка виртуальных машин с указанием причины остановки	Событие:
	Запуск ВМ
	
	Фильтр отбора событий:
	`Virtual machine .* has been started`
		{
		"type": "Control VM",
		"level": "info",
		"name": "Virtual machine 'test-vm' has been started by 'name.surname@test.ru'",
		"datetime": "2025-11-24T09:05:39Z",
		"uid": "cd88fc91-2c33-4bbb-90ca-4ff76e9c721d",
		"request_subject": "name.surname@test.ru",
		"action_type": "start",
		"node_network_address": "unknown",
		"virtualmachine_uid": "986d6e76-ec0d-43ab-a846-171d1ecae437",
		"virtualmachine_os": "",
		"storageclasses": "local-storage-class",
		"qemu_version": "v9.2.0",
		"libvirt_version": "v10.9.0",
		"operation_result": "allow"
		}

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода

	Событие:
	Остановка ВМ с указанием причины остановки
	
	Фильтр отбора событий:
	`Virtual machine .* has been stopped`
		{
		"type": "Control VM",
		"level": "warn",
		"name": "Virtual machine 'test-vm' has been stopped by 'name.surname@test.ru'",
		"datetime": "2025-11-24T08:46:02Z",
		"uid": "00e14c65-25e5-445f-a5fc-3347d5130d53",
		"request_subject": "name.surname@test.ru",
		"action_type": "stop",
		"node_network_address": "10.12.0.200",
		"virtualmachine_uid": "986d6e76-ec0d-43ab-a846-171d1ecae437",
		"virtualmachine_os": "",
		"storageclasses": "local-storage-class",
		"qemu_version": "v9.2.0",
		"libvirt_version": "v10.9.0",
		"operation_result": "allow"
		}

	Событие:
	Перезагрузка ВМ
	
	Фильтр отбора событий:
	`Virtual machine .* has been restarted`
		{
		"type": "Control VM",
		"level": "warn",
		"name": "Virtual machine 'test-vm' has been restarted by 'name.surname@test.ru'",
		"datetime": "2025-11-24T08:46:04Z",
		"uid": "9f768a9a-34ef-49d4-b12c-492c5f9aac25",
		"request_subject": "name.surname@test.ru",
		"action_type": "restart",
		"node_network_address": "10.12.0.200",
		"virtualmachine_uid": "986d6e76-ec0d-43ab-a846-171d1ecae437",
		"virtualmachine_os": "",
		"storageclasses": "local-storage-class",
		"qemu_version": "v9.2.0",
		"libvirt_version": "v10.9.0",
		"operation_result": "allow"
		}

Изменение ролевой модели	События аналогичны событиям средства контейнеризации	
Изменение конфигурации ПО «Deckhouse Platform»	Событие: Изменение конфигурации СВ	{ "type": "Module control", "level": "info", "name": "Module 'virtualization' has

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода
	Фильтр отбора событий: `Module 'virtualization' has been updated`	`been updated by 'kubernetes-admin'", "datetime": "2025-11-24T08:16:43Z", "uid": "f4177935-4b15-4437-a886-f2171fe52ae9", "request_subject": "kubernetes-admin", "operation_result": "allow", "action_type": "patch", "component": "virtualization", "node_network_address": "unknown", "virtualization_version": "v1.73.2", "virtualization_name": "Deckhouse Virtualization Platform", "qemu_version": "unknown", "libvirt_version": "unknown" }`
Изменение конфигураций виртуальных машин	Событие: Изменение конфигурации ВМ Фильтр отбора событий: `Virtual machine .* has been updated`	`{ "type": "Manage VM", "level": "info", "name": "Virtual machine 'test-vm' has been updated by 'name.surname@test.ru'", "datetime": "2025-11-24T09:05:01Z", "uid": "611f0dd3-cf37-4899-97d0-201170a95af7", "request_subject": "name.surname@test.ru", "action_type": "update", "node_network_address": "10.12.0.200", "virtualmachine_uid": "986d6e76-ec0d-43ab-a846-171d1ecae437", "virtualmachine_os": "unknown", "storageclasses": "local-storage-class", "qemu_version": "v9.2.0", "libvirt_version": "v10.9.0", "operation_result": "allow" }`
Факты нарушения целостности объектов контроля	Событие: Нарушение контроля целостности конфигурации ВМ Фильтр отбора событий: `Virtual machine .* config integrity check failed`	`{ "type": "Integrity check", "level": "critical", "name": "Virtual machine 'test-vm' config integrity check failed", "datetime": "2025-11-25T05:46:50Z", "uid": "a8344637-1ed8-4b06-b769-21fa4e73dd96", "request_subject": "system:serviceaccount:d8-virtualization:kubevirt-internal-virtualization-handler", "operation_result": "allow", "object_type": "Virtual machine configuration",`

Регистрируемые события безопасности	События безопасности ПО «Deckhouse Platform»	Пример вывода
		"virtual_machine_name": "test-vm", "control_method": "Integrity Check", "reaction_type": "info", "integrity_check_algo": "sha256", "reference_checksum": "a57fee12cd9af67dd3105026d6f20ad3f30e4de8598fb4600f233be34fcbcc62", "current_checksum": "4ca8a36fb8b3bf12a922530de0dae794acf5698bc4d77440c38a17bbcb0e97b5" }

Лист регистрации изменений

Номер изменения	Номера страниц				Всего страниц в документе	Наименование и номер документа, вводящего изменения	Подпись, Ф.И.О. внесшего изменения в данный экземпляр	Дата внесения изменения в данный экземпляр	Дата введения изменения
	измененных	замененных	новых	аннулиро-ванных					
1	2	3	4	5	6	7	8	9	10