УТВЕРЖДЕНО

RU.86432418.00001-01 90 01-1 - ЛУ

Программное обеспечение «Deckhouse Platform Certified Security Edition»

Руководство администратора

RU.86432418.00001-01 90 01-1

Листов 41

2024

Содержание

Список используемых обозначений и сокращений

ОО	Объект оценки
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 01-1».

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;
- проверка целостности дистрибутива ПО «Deckhouse Platform»;
- проверка целостности установленного ПО «Deckhouse Platform».

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

Проверка целостности ПО «Deckhouse Platform» на USB флэш-накопителе информации выполняется путем снятия контрольных сумм с помощью с использованием 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 USB флэш-накопитель информации, соответствует значению контрольной суммы, приведенной в Формуляре – требуется выполнить установку ПО «Deckhouse Platform» в соответствии с разделом 3 настоящего документа. Если по результатам проверки контрольная сумма дистрибутива ПО «Deckhouse Platform», записанного на USB флэш-накопитель информации, 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 4.4. 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 для работы в среде функционирования ПО «Deckhouse Platform»;
- действия, проводимые при инсталляции ПО «Deckhouse Platform», а также при инициализации ПО «Deckhouse Platform», подлежат логированию;
- установка и конфигурирование ПО «Deckhouse Platform» должны осуществляться в соответствии с эксплуатационной документацией;
- 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. Установка и настройка

Перед установкой необходимо убедиться, что выполнены минимальные требования к аппаратным и программным средствам (п. 3.2.4 ТУ) (Таблица 1).

Таблица 1 – Минимальные требования к аппаратным и программным средствам

Изделие	**Требования к аппаратной части**	**Требования к программной части**
ПО «Deckhouse Platform»	− Архитектура процессора x86-64; − Не менее 4 ядер CPU; − не менее 8 GB RAM; − не менее 40 ГБ дискового пространства	− ОС РЕД ОС (7.3), ОС Astra Linux Special Edition (1.7), Альт 8 СП.

Перед установкой необходимо проверить следующие условия:

- ОС сервера находится в списке поддерживаемых ОС и до сервера есть SSH-доступ по ключу;
- На сервере обновлены пакеты ОС до последних версий;
- Наличие доступа к хранилищу образов контейнеров (container registry) для загрузки образов ПО «Deckhouse Platform».

3.1. Файл конфигурации установки

Для установки ПО «Deckhouse Platform» нужно подготовить YAML-файл конфигурации установки и, при необходимости, YAML-файл ресурсов, которые нужно создать после успешной установки ПО «Deckhouse Platform».

YAML-файл конфигурации установки содержит параметры нескольких ресурсов (манифесты):

- InitConfiguration – начальные параметры конфигурации ПО «Deckhouse Platform». С этой конфигурацией ПО «Deckhouse Platform» запустится после установки.

```
apiVersion: deckhouse.io/v1
kind: InitConfiguration
deckhouse:
  imagesRepo: registry.company.my/deckhouse/cse
  registryDockerCfg:
eyJhdXRocyI6IHsgIm5leHVzLmNvbXBhbnkubXkiOiB7InVzZXJuYW1lIjoibmV4dXMtdXNlci
IsInBhc3N3b3JkIjoibmV4dXMtcEBzc3cwcmQiLCJhdXRoIjoiYm1WNGRYTXRkWE5sY2pwdVpY
aDFjeTF3UUhOemR6QnlaQW89In19fQo=
  registryScheme: HTTPS
  registryCA: |
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
```

В этом ресурсе, в частности, указываются параметры, без которых ПО «Deckhouse Platform» не запустится, или будет работать некорректно. Например, параметры размещения компонентов ПО «Deckhouse Platform», используемый storageClass, параметры доступа к container registry, шаблон используемых DNS-имен и другие.

- ClusterConfiguration – общие параметры кластера, такие как версия control plane, сетевые параметры, параметры CRI и т.д.

```
apiVersion: deckhouse.io/v1
kind: ClusterConfiguration
podSubnetNodeCIDRPrefix: '24'
podSubnetCIDR: 10.244.0.0/16
serviceSubnetCIDR: 192.168.0.0/16
kubernetesVersion: '1.27'
clusterDomain: k8s.internal
clusterType: Static
proxy:
  httpProxy: https://user:password@proxy.company.my:8443
  httpsProxy: https://user:password@proxy.company.my:8443
  noProxy:
  - company.my
```

- StaticClusterConfiguration – параметры кластера Kubernetes, разворачиваемого на серверах bare metal или на виртуальных машинах.

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

- ModuleConfig – набор ресурсов, содержащих параметры глобальной конфигурации и конфигурации встроенных модулей Deckhouse.

```
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: deckhouse
spec:
  version: 1
  enabled: true
  settings:
    bundle: Default
    releaseChannel: Stable
    logLevel: Info
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: global
spec:
  version: 1
  settings:
    modules:
      publicDomainTemplate: "%s.kube.company.my"
```

```
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: cni-flannel
spec:
  version: 1
  enabled: true
  settings:
    podNetworkMode: VXLAN
```

3.2. Установка ПО «Deckhouse Platform»

Для установки ПО «Deckhouse Platform» требуется наличие в локальной сети хранилища образов контейнеров (container registry). В случае отсутствия registry, выполните шаги по его установке, приведенные в п. 3.8 Развертывание registry.

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

С помощью утилиты 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 deckhouse-cse-1.58.2.tar и сравните полученные результаты соответственно с содержимым файлов d8.gostsum и deckhouse-cse-1.58.2.tar.gostsum. Полученные результаты расчета контрольных сумм и контрольные суммы в указанных файлах поставки должны совпадать.

Загрузите данные в хранилище образов контейнеров, выполнив следующую команду:

```
d8 mirror push deckhouse-cse-1.58.2.tar <REGISTRY_ULR>/<PATH>' \
  --registry-login='<USERNAME>' --registry-password='<PASSWORD>'
```

, где

- `<REGISTRY_URL>` - адрес хранилища образов контейнеров в локальной сети;
- `<PATH>` - путь в хранилище образов коннтейнеров, 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.

Подробнее, загрузка образов в registry рассмотрена в п. 3.8 Загрузка образов в изолированный registry.

Запустите контейнер установщика:

```
docker run --pull=always -it [<MOUNT_OPTIONS>]
<REGISTRY_URL>/deckhouse/cse/install:<RELEASE_CHANNEL> bash
```

, где:

- `<REGISTRY_URL>` – адрес container registry с образами ПО «Deckhouse Platform».
- `<RELEASE_CHANNEL>` - тэг версии ПО «Deckhouse Platform» - v1.58.2
- `<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/resources.yml:/resources.yml" \
  -v "$PWD/dhctl-tmp:/tmp/dhctl" \
  -v "$HOME/.ssh/:/tmp/.ssh/" registry.deckhouse.io/deckhouse/cse/install:
v1.58.2 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 --resources=/resources.yml
```

, где:

- `/config.yml` — файл конфигурации установки;
- `/resources.yml` — файл манифестов ресурсов;
- `<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.

3.3. Откат установки и удаление ПО «Deckhouse Platform»

При установке в случае прерывания установки или возникновения проблем во время установки, могут остаться созданные ресурсы. Для удаления созданных 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.

3.4. 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:

kubectl -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: '1.27'
clusterDomain: k8s.internal
clusterType: Static
proxy:
  httpProxy: https://user:password@proxy.company.my:8443
  httpsProxy: https://user:password@proxy.company.my:8443
  noProxy:
  - company.my
```

- apiVersion *строка*

  Обязательный параметр

  Используемая версия API Deckhouse.

  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’ах) по умолчанию.

Если используется значение NotManaged, то ПО «Deckhouse Platform» не будет управлять (устанавливать и настраивать) container runtime. В этом случае образы, используемые в NodeGroup’ах, должны содержать уже установленный container runtime.

По умолчанию: "Containerd"

Допустимые значения: Containerd, NotManaged

- kind *строка*

  Обязательный параметр

  Допустимые значения: ClusterConfiguration

- kubernetesVersion *строка*

  Обязательный параметр

  Версия control plane кластера Kubernetes.

  Допустимые значения: 1.27

- 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.

3.5. Ресурс InitConfiguration

Version: deckhouse.io/v1

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

Пример:

```
apiVersion: deckhouse.io/v1
kind: InitConfiguration
deckhouse:
  imagesRepo: registry.company.my/deckhouse/cse
  registryDockerCfg:
eyJhdXRocyI6IHsgIm5leHVzLmNvbXBhbnkubXkiOiB7InVzZXJuYW1lIjoibmV4dXMtdXNlciIsInBhc3N3b3JkIjoibmV4dXMtcEBzc3cwcmQiLCJhdXRoIjoiYm1WNGRYTXRkWE5sY2pwdVpYaDFjeTF3UUhOemR6QnlaQW89In19fQo=
  registryScheme: HTTPS
  registryCA: |
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
```

- apiVersion *строка*

Обязательный параметр

Используемая версия API Deckhouse.

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

3.6. 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
```

- apiVersion *строка*

  Обязательный параметр.

  Используемая версия API Deckhouse.

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

- internalNetworkCIDRs *массив строк*

  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

3.7. Создание образов

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

Описания создания образов приведено ниже

- Astra Linux Special Edition
  `https://wiki.astralinux.ru/pages/viewpage.action?pageId=158601444&sclid=lwsxujpv4r224070482`
- ОС РЕД ОС
  `https://redos.red-soft.ru/base/arm/arm-other/docker-install/?ysclid=lwsy07s1ux387721779`
- Альт 8 СП
  `https://www.altlinux.org/Docker`

При включенном и настроенном operator-trivy п. 4.1, развертывание контейнеров из уязвимых образов будет запрещено.

3.8. Развертывание registry

Для развертывания registry (хранилища образов контейнеров) выполните следующие шаги:

Установите на сервер пакеты ОС docker-registry apache2-utils:

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

Сгенерируйте пользователя для доступа в registry (в примере, пользователь deckhouse с паролем deckhnouse):

```
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

Перезапустите сервис docker-registry:

```
systemctl restart docker-registry
```

Используйте адрес registry при загрузке образов перед установкой ПО «Deckhouse Platform» (п. 3.2). Пример:

```
d8 mirror push /root/d8.tar 10.128.0.37:5000/deckhouse --insecure --
registry-login=deckhouse --registry-password=deckhouse
```

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

3.9. 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 Deckhouse) и других файлов.

ПО «Deckhouse Platform» настраивается с помощью:

- **Глобальных настроек.** Глобальные настройки хранятся в custom resource ModuleConfig/global. Глобальные настройки можно рассматривать как специальный модуль global, который нельзя отключить.
- **Настроек модулей.** Настройки каждого модуля хранятся в custom resource ModuleConfig, имя которого совпадает с именем модуля (в kebab-case).
- **Custom resource'ов.** Некоторые модули настраиваются с помощью дополнительных custom resource'ов.

Пример набора custom resource'ов 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 custom resource’ов ModuleConfig, состояние модуля (включен/выключен) и его статус можно с помощью команды kubectl get moduleconfigs.

Список и состояние модулей можно также получить с помощью команды kubectl 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:

```
kubectl -n d8-system edit moduleconfig/upmeter
```

После завершения редактирования изменения применяются автоматически.

3.10. Настройка модуля

Модуль настраивается с помощью custom resource ModuleConfig, имя которого совпадает с именем модуля (в kebab-case). Custom resource 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 custom resource’а 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 включён по умолчанию в одном из наборов модулей.

ПО «Deckhouse Platform» не изменяет custom resource’ы 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 Helm, kubectl и другие привычные инструменты.

Пример custom resource 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 custom resource’ов. Воспользуйтесь поиском (вверху страницы) или выберите модуль в меню слева, чтобы просмотреть документацию по его настройкам и используемым custom resource’ам.

3.11. 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 custom resource `ModuleConfig`. Если для модуля нет такого custom resource `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 `kubectl get moduleconfig <ИМЯ_МОДУЛЯ>`.

3.12. 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. спецификацию в п. 3.12.1).

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 ПО «Deckhouse Platform Certified Security Edition»).

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
```

3.12.1. 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 юзера 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"

- spec.oidc.clientID *строка*

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

- spec.oidc.clientSecret *строка*

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

- spec.oidc.getUserInfo *булевый*

  Запрашивать дополнительные данные об успешно подключенном пользователе.
  Подробнее…

  По умолчанию: false

- 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. Описание параметров (настроек) безопасности

4.1. 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. 4.1.1)
- установить на пространствах имен, содержащих подлежащие сканированию поды, аннотацию security-scanning.deckhouse.io/enabled.

Для включения модуля operator-trivy выполните следующую команду (требуется kubectl, настроенный на работу с кластером):

kubectl -ti -n d8-system exec deploy/deckhouse -- deckhouse-controller 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 kubectl, настроенный на работу с кластером):

kubectl 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.4.5):

4.1.1. Обновление базы уязвимостей.

Обновление базы уязвимостей необходимо выполнять периодически. Рекомендуемая частота обновления баз – один раз в сутки.

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 мастер-узле (укажите имя файла и лицензионный ключ):

d8 mirror vuln-db pull –source registry-cse.deckhouse.ru -l <LICENSE_KEY> <DB_FILE_PATH>

Для закачивания обновлений базы уязвимостей из файла, выполните следующую команду на мастер-узле (укажите имя файла с базой обновлений и данные хранилища образа контейнеров):

d8 mirror vuln-db push <DB_FILE_PATH> <REGISTRY_URL> --registry-login <REGISTRY_LOGIN> --registry-password <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.

4.2. Настройка политик безопасности.

В ПО «Deckhouse Platform» предусмотрено использование следующих политик безопасности:

- *Privileged* — неограничивающая 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 жесткие 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. 4.2.1).

При необходимости изменить политику на конкретном пространстве имен, на него нужно установить лейбл *security.deckhouse.io/pod-policy=<НАЗВАНИЕ_ПОЛИТИКИ>*.

Пример команды установки политики Restricted на пространство имен my-namespace (требуется kubectl, настроенный на работу с кластером):

kubectl 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 kubectl;
- *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. 4.2.1). 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>* на соответствующем namespace. Список допустимых режимом политик состоит из: “dryrun”, “warn”, “deny”.

Пример команды установки режима *Warn* на пространство имен my-namespace (требуется kubectl, настроенный на работу с кластером):

kubectl label ns my-namespace security.deckhouse.io/pod-policy-action=warn

4.2.1. Ресурс ModuleConfig

- apiVersion: deckhouse.io/v1alpha1
- kind: ModuleConfig
- metadata:
  - name: admission-policy-engine
- spec:
  - version: 1
  - enabled: true
  - settings
    - 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.

    По умолчанию:
    - Baseline — при первичной установке Deckhouse версии v1.58 и выше;
    - Privilged — при первичной установке Deckhouse версии ниже v1.55 (обновление Deckhouse в кластере на версию v1.58 и выше **не меняет** политику по умолчанию на Baseline).

    Допустимые значения: Privileged, Baseline, Restricted

- 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 kubectl;
  - 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"

  Допустимые значения: Warn, Deny, Dryrun

- 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

4.3. Настройка уведомлений о событиях безопасности на почту.

Чтобы настроить уведомлений о событиях безопасности на почту, необходимо выполнить следующую команду:

kubectl 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
```

4.4. Настройка доступа к журналам событий безопасности

Чтобы настроить доступ пользователям к Grafana и Prometheus, необходимо выполнить следующую команду:

kubectl apply -f prometheus.yaml

Пример файла prometheus.yaml:

```
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
spec:
  enabled: true
  settings:
    auth:
      allowedUserGroups:
      - administrator
      - security
  version: 2
status:
  state: Enabled
  status: Ready
  type: ""
  version: "2"
```

- auth

Опции, связанные с аутентификацией или с авторизацией в приложении.

- auth.allowedUserGroups

Массив групп, пользователям которых позволен доступ в Grafana и Prometheus.

4.5. 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.

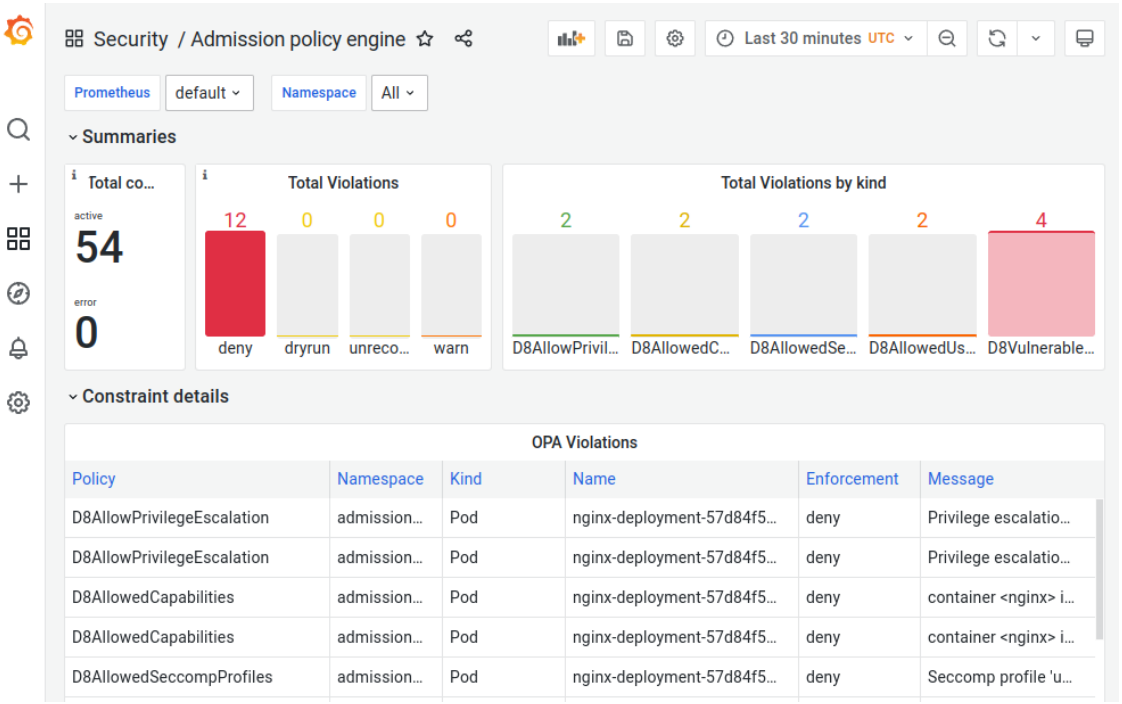

Рис. 4.5.1. Пример дашборда Admission policy engine

- *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.4.1.).

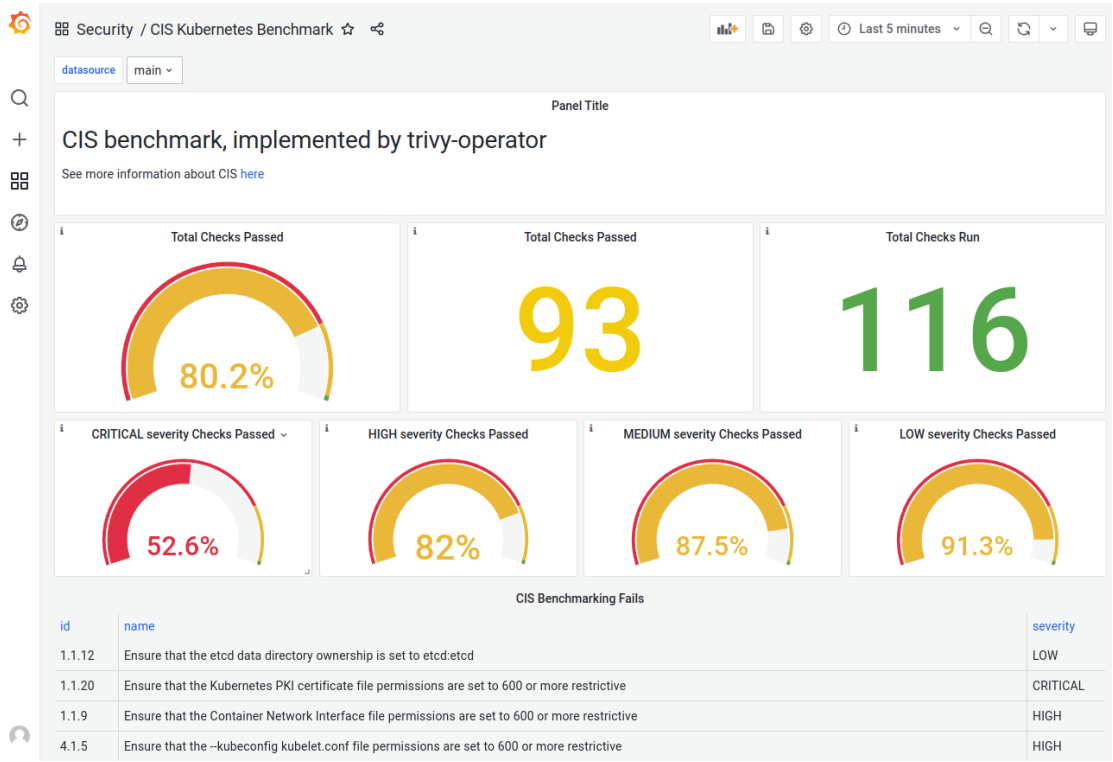

Рис. 4.5.3. Пример дашборда CIS Kubernetes Benchmark

- *Kubernetes audit logs.* Журнал регистрации обращений к API-серверу. Содержит записи о всех обращениях к API-серверу кластера в JSON-формате.

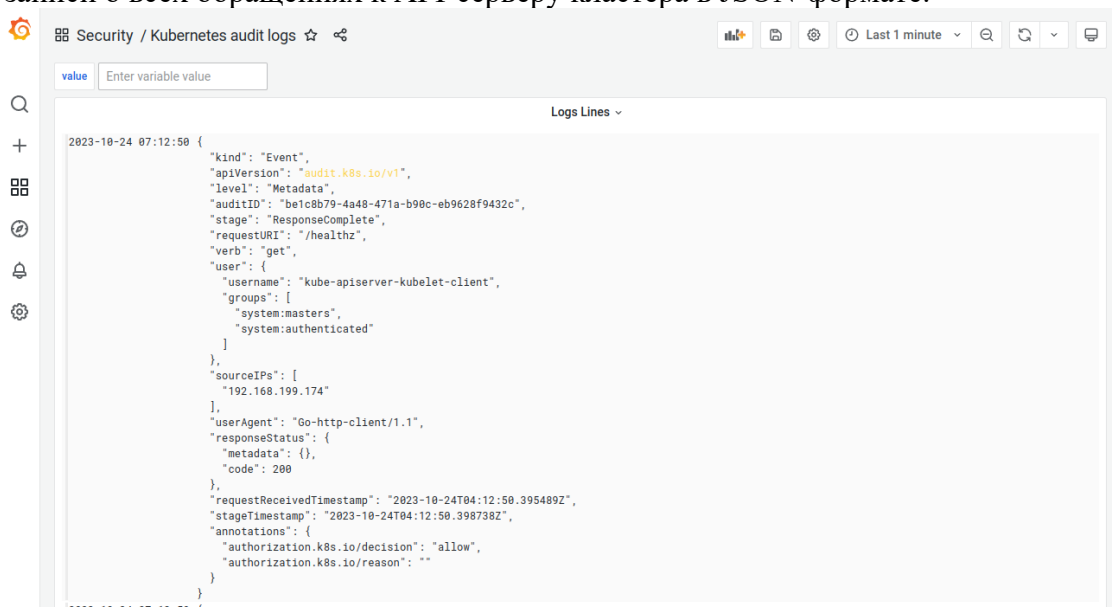

Рис. 4.5.4. Пример дашборда Kubernetes audit logs

- *Runtime audit engine logs.* Журнал регистрации событий безопасности аудита работы ядра Linux и API-сервера кластера.

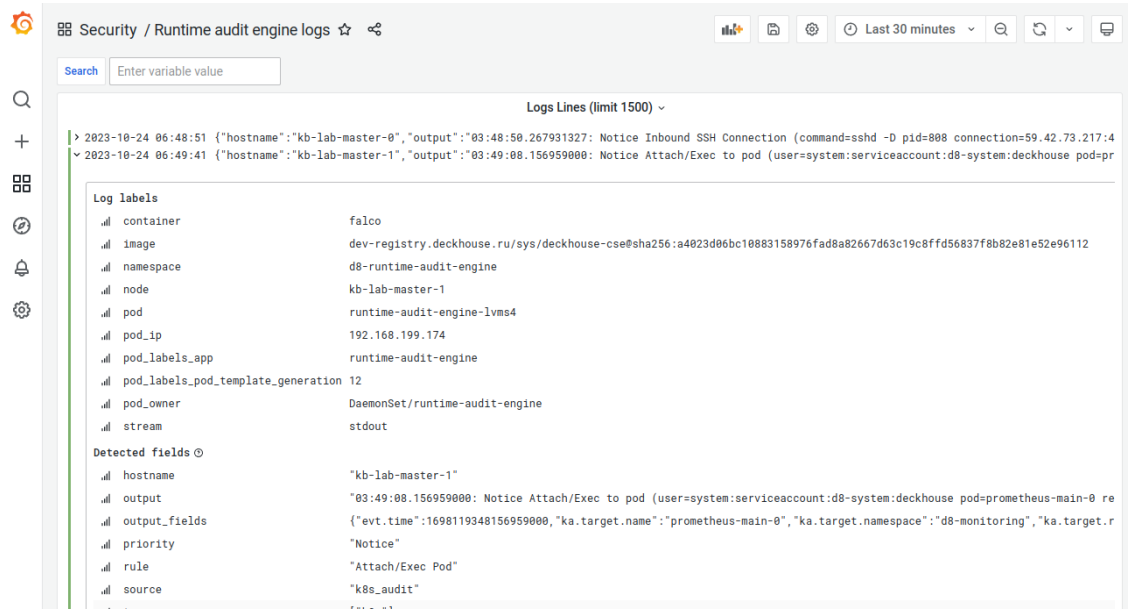

Рис. 4.5.5. Пример дашборда Runtime audit engine logs

- *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.4.1).

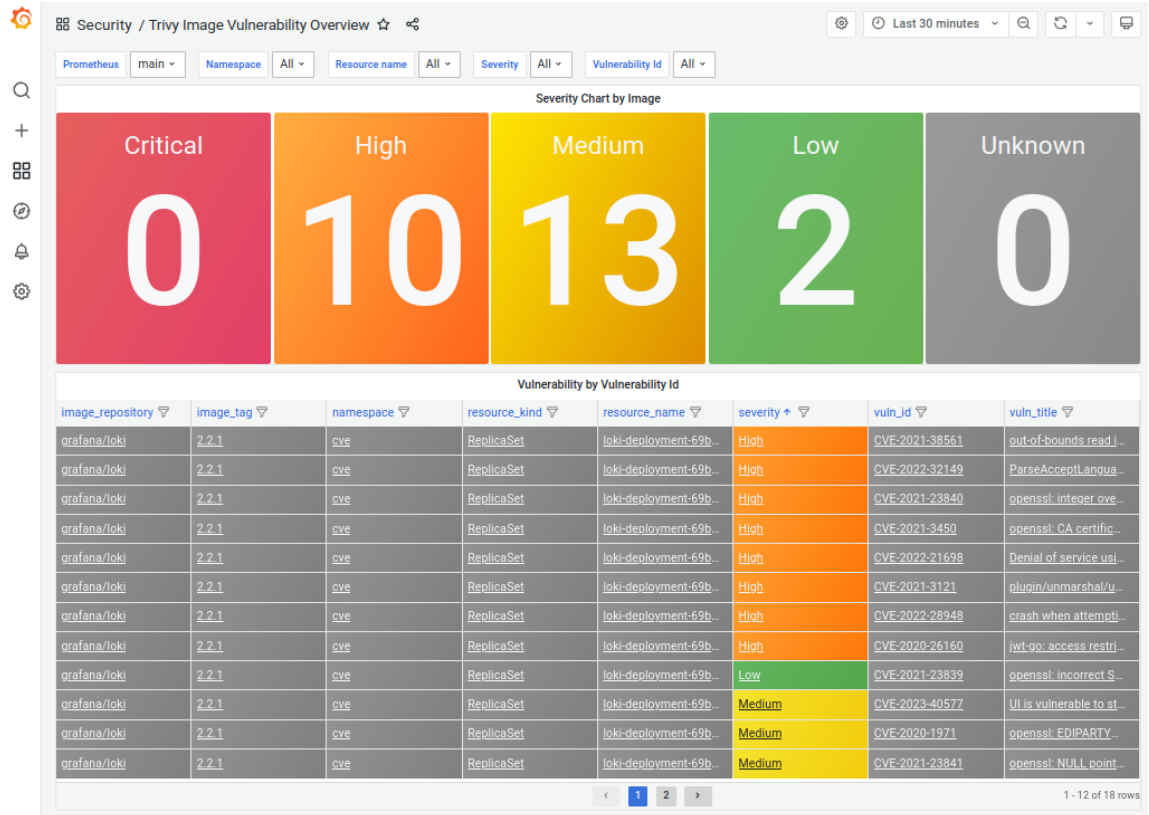

Рис. 4.5.6. Пример дашборда Trivy Image Vulnerability Overview

4.6. Контроль целостности

Для проверки целостности образа используется контрольная сумма, рассчитанная по алгоритму Стрибог (ГОСТ Р 34.11-2012).

Чтобы устанавливаемые образы проверялись, необходимо добавить метку gost-integrity-controller.deckhouse.io/gost-digest-validation-enabled: true в пространство имен кластера, где необходимо производить контроль целостности образа.

Пример:

```
apiVersion: v1
kind: Namespace
metadata:
labels:
gost-integrity-controller.deckhouse.io/gost-digest-validation-enabled: true
name: default
```

Если в ходе проверки контрольная сумма образа окажется некорректной, установка образа будет отклонена, и вы получите 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"
]
```

Склеенная строка из контрольных сумм

```
"sha256:27e923fb52d31d7e3bdade76ab9a8056f94dd4bc89179d1c242c0e58592b4d5c
sha256:360eba32fa65016e0d558c6af176db31a202e9a6071666f9b629cb8ba6ccedf0
sha256:72de7d1ce3a476d2652e24f098d571a6796524d64fb34602a90631ed71c4f7ce
sha256:907d1bb4e9312e4bfeabf4115ef8592c77c3ddabcfddb0e6250f90ca1df414fe
sha256:94f34d60e454ca21cf8e5b6ca1f401fcb2583d09281acb1b0de872dba2d36f34
sha256:c5903f3678a7dec453012f84a7d04f6407129240f12a8ebc2cb7df4a06a08c4f
sha256:e42dcfe1730ba17b27138ea21c0ab43785e4fdbea1ee753a1f70923a9c0cc9b8"
```

Контрольная сумма образа

```
2f538c22adbdb2ca8749cdafc27e94baed8645c69d4f0745fc8889f0e1f5a3f9
```

Скопировать бинарный файл imagedigest из дистрибутива ПО «Deckhouse Platform Certified Security Edition». Из образа install. Для этого перейти в каталог с дистрибутивом, распаковать архив deckhouse-cse-1.58.2.tar с помощью следующий команды:

tar -xvf /distr/deckhouse-cse-1.58.2.tar -C /distr/deckhouse-cse-1.58.2

Перейти в каталог deckhouse-cse-1.58.2 распаковать архив f67338b4fb93a72e06b8d0a0b768b2f358e766a20530ab7243fc7eac2ea73ca с помощью следующий команды:

tar -xvf /distr/deckhouse-cse-1.58.2/install/blob/sha256/f67338b4fb93a72e06b8d0a0b768b2f358e766a20530ab7243fc7eac2ea73ca

Из каталога /distr/deckhouse-cse-1.58.2/install/blob/sha256/usr/bin/ скопировать бинарный файл imagedigest к себе на ЭВМ.

С помощью утилиты imagedigest выполняется расчет контрольной суммы образа, подпись и проверка.

imagedigest calculate <имя_образа> - расчет контрольной суммы образа

imagedigest add <имя_образа> - подпись

imagedigest validate <имя_образа> -проверка

Для каждого образа администратор должен фиксировать в журнал идентификатор пользователя, подписавшего образ, и вычисленную контрольную сумму образа.

5. 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 vim 192.168.199.1
# registry
ip.ro add 5.182.5.140/32 vim 192.168.199.1
# cluster networks
ip.ro add 10.111.0.0/16 vim 192.168.199.1
ip.ro add 10.222.0.0/16 vim 192.168.199.1
# remove all routes
ip.ro add 5.182.5.140/32 vim 192.168.199.1
ip ro del default
```

- для осуществления безопасной настройки ОС на узлах кластера после установки ПО «Deckhouse Platform» необходимо применить в кластере YAML-файл, приведенный в электронном приложении к настоящему документу, каталог «Электронные приложения» - «Deckhouse Platform. Руководство администратора» - ngc.yaml. Для этого выполните следующую команду:

  kubectl create ngc.yaml

  Данный файл подготовлен в соответствии с методическим документом ФСТЭК России от 25 декабря 2022 г. «Рекомендации по безопасной настройке

операционных систем Linux» (https://fstec.ru/dokumenty/vse-dokumenty/spetsialnye-normativnye-dokumenty/metodicheskij-dokument-ot-25-dekabrya-2022-g)

- 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.

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	аннулированных					
1	2	3	4	5	6	7	8	9	10