Настройка проверки цифровой подписи образов контейнеров с помощью локальной инфраструктуры Sigstore

Для проверки цифровой подписи в Neuvector используются компоненты и методики проекта https://www.sigstore.dev/. Инфраструктура для осуществления полного цикла цифровой подписи по умолчанию доступа в сети Интернет, однако вы можете развернуть данные компоненты локально:

- **Rekor** — сервер журналирования артефактов,
- **Fulcio** — служба выдачи временных сертификатов,
- **Cosign** — инструмент подписи образов контейнеров и артефактов.

Далее представлено краткое руководство по запуску инфраструктуры для реализации процессов цифровой подписи.

1. Подготовка сервера

1. Подготовьте отдельный сервер для установки компонентов Sigstore.
2. Выполните установку на сервер ПО Docker.

2. Установка Cosign

Выполните установку последней доступной версии из репозитория: https://github.com/sigstore/cosign/releases/tag/v2.4.3

3. Установка сервера Rekor

При развертывании сервисов **Rekor** и **Fulcio** на одной машине, порт у **Rekor** должен отличатся от порта **Fulcio**.
Измените порт по умолчанию (`2112`) в `docker-compose.yml` **Rekor**, например, на `2113`.

1. Выполните сборку и запуск Rekor с помощью Docker Compose:

```bash
git clone https://github.com/sigstore/rekor.git
cd rekor
```

```
git checkout tags/v1.3.9
docker compose -f docker-compose.yml up -d
```

2. После установки проверьте, что сервер инициализирован, и его лог доступен:

```
BASH
curl http://localhost:3000/api/v1/log
```

Пример лога:

```
TERMINAL
{"inactiveShards":null,"rootHash":"e3b0c44298fc1c149afbf4c8996fb92427ae41e
4649b934ca495991b7852b855","signedTreeHead":"a1d947edf108 -
6903076924831886960\n0\n47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=\n\n—
a1d947edf108
bmbN4zBFAiArgMWxaNRdNZX3Zfvz2Vxyv4rdO/fvSh/WFKnkudhmogIhAPJpNQj4wUM6WkWLUA
QNa9AUV19DDa+YyDWG5pw7URZ3\n","treeID":"6903076924831886960","treeSize":0}
```

3. Получите публичный ключ Rekor, выполнив команду:

```
BASH
curl localhost:3000/api/v1/log/publicKey
```

Пример публичного ключа Rekor:

```
TERMINAL
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE/UTpADGGAmntz/Vi7LMfRXqgcS6l
je7WTMm2zHNpj4/WT8XSrExq00Cb5vrgisSLufJkxkPaRLFCLH5rAt3w4w==
-----END PUBLIC KEY-----
```

4. Установка сервера Fulcio

1. Выполните сборку и запуск Fulcio с помощью Docker Compose:

```
BASH
git clone https://github.com/sigstore/fulcio.git
cd fulcio
git checkout tags/v1.6.6
docker compose -f docker-compose.yml up -d
```

2. Получите корневой сертификат Fulcio, выполнив команду:

```
BASH
curl -X GET http://localhost:5555/api/v1/rootCert
```

Пример корневого сертификата Fulcio:

```
TERMINAL
-----BEGIN CERTIFICATE-----
MIICFjCCAb2gAwIBAgIUFlLnHDwqponlw4BE+FD3W0pZ40YwCgYIKoZIzj0EAwIw
```

```
aDEMMAoGA1UEBhMDVVNBMQswCQYDVQQIEwJXQTERMA8GA1UEBxMIS2lya2xhbmQx
FTATBgNVBAkTDDc2NyA2dGggU3QgUzEOMAwGA1UEERMFOTgwMzMxETAPBgNVBAoT
CHNpZ3N0b3JlMB4XDTI1MDIyNjE0NTAxMFoXDTM1MDIyNjE0NTAxMFowaDEMMAoG
A1UEBhMDVVNBMQswCQYDVQQIEwJXQTERMA8GA1UEBxMIS2lya2xhbmQxFTATBgNV
BAkTDDc2NyA2dGggU3QgUzEOMAwGA1UEERMFOTgwMzMxETAPBgNVBAoTCHNpZ3N0
b3JlMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE9UryrQoVkO/18FeXpR7/BYdJ
FuGMEq5V3ssnUEMZURGOa6GLxQsFKkSPL8xHKUoMkiCyA+r6sXXwTMxFedQECqNF
MEMwDgYDVR0PAQH/BAQDAgEGMBIGA1UdEwEB/wQIMAYBAf8CAQEwHQYDVR0OBBYE
FAOgPppZOXpC2bew48TuabKNfGH+MAoGCCqGSM49BAMCA0cAMEQCIB/2EBYLHce0
nr/5ZomjiKRCoFMvxvAVaicpzOCoELIlAiADa3/cjVJuHdeo85EUC13MX1cT9uHI
FZCkzktJN2lxmw==
-----END CERTIFICATE-----
```

5. Использование Cosign

1. Сгенерируйте ключевую пару для дальнейшей подписи образов.
2. Подпишите необходимый образ, выполнив команду:

```
BASH
cosign sign --key cosign.key --fulcio-url=http://FULCIO_SERVER:5555 --rekor-
url=http://REKOR_SERVER:3000 harbor.domain.local/public/nginx-signed:latest
```

где:

- `172.31.100.160` — IP-адрес сервера, на который выполнялась установка компонентов Sigstore;
- `harbor.domain.local/public/nginx-signed:latest` — путь к образу контейнера в хранилище.

  Далее введите приватный ключ:

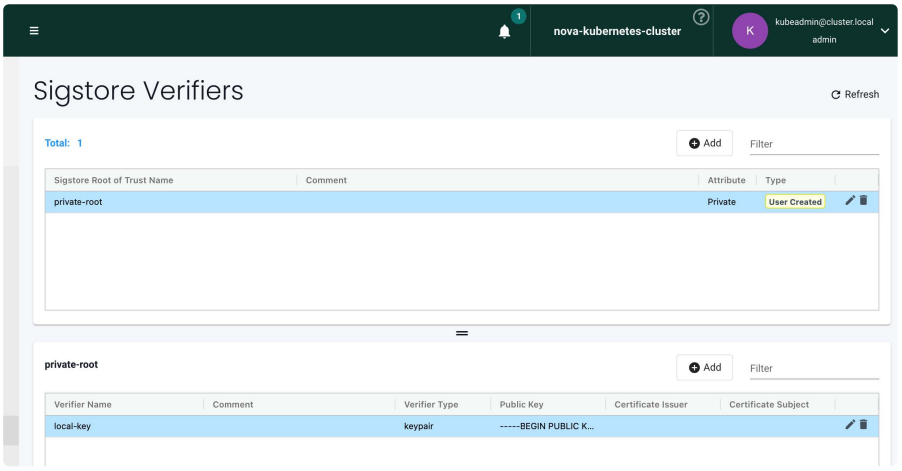
```
TERMINAL
Enter password for private key:
WARNING: Image reference harbor.domain.local/public/nginx-signed:latest
uses a tag, not a digest, to identify the image to sign.

This can lead you to sign a different image than the intended one.
Please use a digest (example.com/ubuntu@sha256:abc123...) rather than
tag (example.com/ubuntu:latest) for the input to cosign. The ability to
refer to images by tag will be removed in a future release.
The sigstore service, hosted by sigstore a Series of LF Projects, LLC,
is provided pursuant to the Hosted Project Tools Terms of Use, available
at https://lfprojects.org/policies/hosted-project-tools-terms-of-use/.
Note that if your submission includes personal data associated with this
signed artifact, it will be part of an immutable record.
This may include the email address associated with the account with
which you authenticate your contractual Agreement.
This information will be used for signing this artifact and will be
```

```
stored in public transparency logs and cannot be removed later, and is
subject to the Immutable Record notice at
https://lfprojects.org/policies/hosted-project-tools-immutable-records/.
By typing 'y', you attest that (1) you are not submitting the personal
data of any other person; and (2) you understand and agree to the
statement and the Agreement terms at the URLs listed above. Are you sure
you would like to continue? [y/N] y tlog entry created with index: 0
Pushing signature to: harbor.domain.local/public/nginx-signed
```

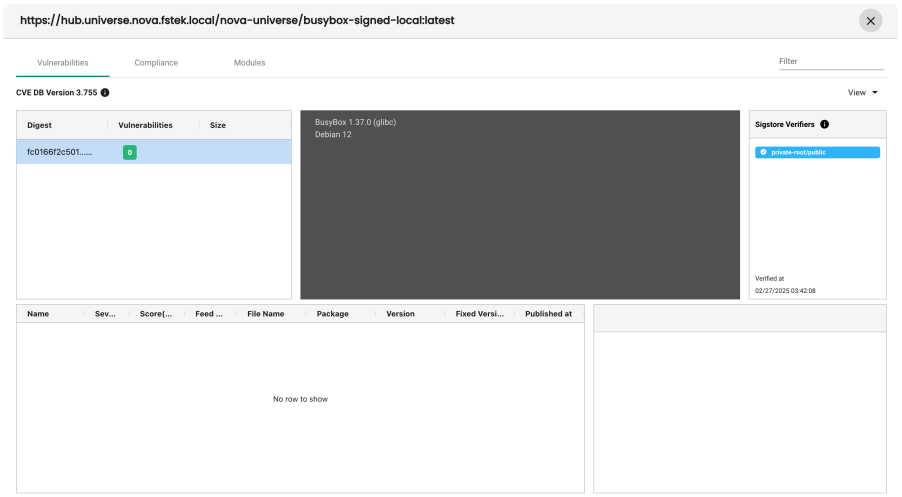
6. Настройка Neuvector

В разделе Sigstore Verifiers добавить новый доверенный источник:

1. Укажите имя, например, **private-root**
2. Атрибут — `Private`
3. В качестве Rekor Public Key укажите публичный ключ сервера Rekor, полученный ранее.
4. В качестве Root Certificate укажите сертификат сервера Fulcio, полученный ранее.
5. Добавьте публичный ключ, которым осуществлялась цифровая подпись образа.



6. Для добавленного хранилища выполните повторное сканирование образов и проверите результаты подписи.



Настройка проверки подписи образов

1. Предварительные условия

1. Установлена утилита для подписи образов Cosign
2. Установлен сервис NeuVector
3. Создано пространство имён (в котором будет разрешен запуск только подписанных образов).
   Откройте в браузере веб-интерфейс Nova Console, перейдите на страницу *Home* → *Namespaces* и нажмите *"Create Namespace"*
   В открывшемся окне введите имя для неймспейса, например `nova-cosign` и нажмите *"Create"*.

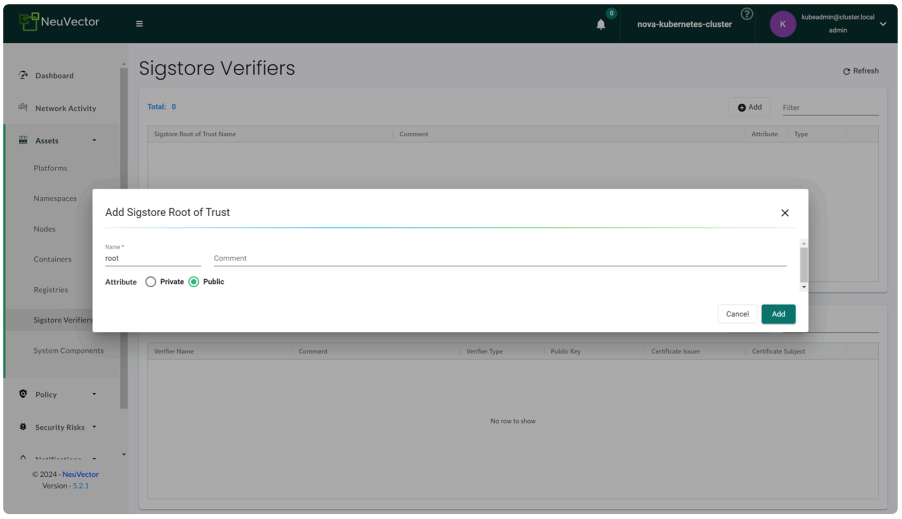
   Примечание: Если вы выберете другое имя, которое не начинается с *"nova-"*, то создание пода будет неуспешным из-за более раннего условия NeuVector (№1002).

2. Настройка

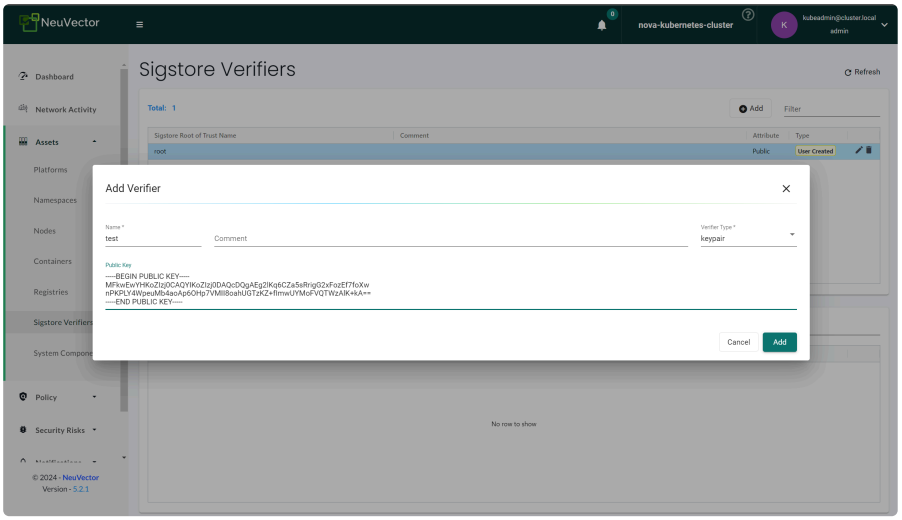
1. Сгенерируйте пару ключей выполнив следующую команду (в данном примере ключи будут сохранены в рабочую директорию, их так же можно сохранить в Vault или в секрет K8S, подробнее https://docs.sigstore.dev/cosign/key_management/overview/):

   ```
   cosign generate-key-pair
   ```

2. В браузере откройте веб-интерфейс Neuvector и перейдите на страницу *Assets* → *Sigstore Verifiers*.
3. Для начала необходимо добавить корневой доверенный сертификат (*Sigstore Root of Trust*). Можно использовать свой сертификат или публичный (в данном примере будет использован публичный).
   Для добавления сертификата нажмите кнопку *"+ Add"* в правом-верхнем углу. В открывшемся окне выберите *public* и в поле *Name* введите *root*, после чего нажмите *"Add"*

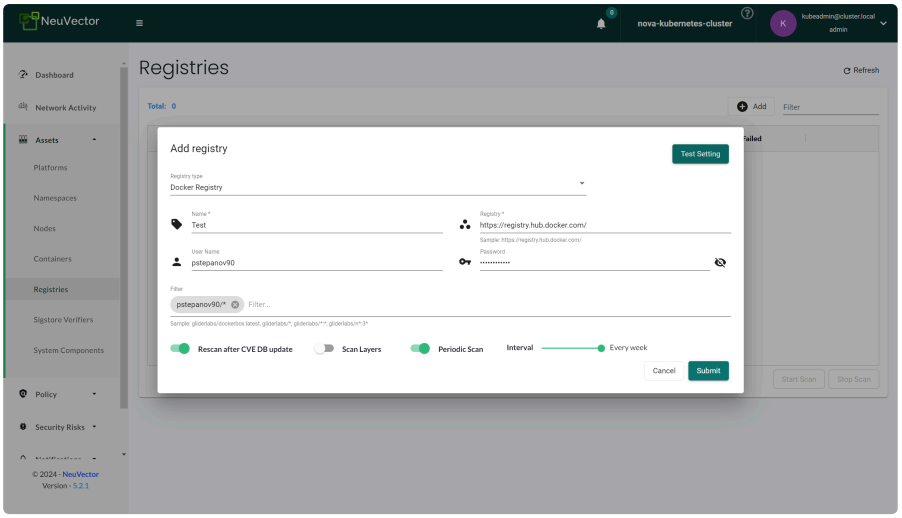


4. Теперь необходимо добавить ключ, сгенерированный в п.1 в качестве верификатора. Для этого, нажмите *"+ Add"* в разделе *Verifiers*. В открывшемся окне введите имя (например *test*), вставьте публичный ключ *cosign.pub*, сгенерированный в п.1 и нажмите *"Add"*.



5. Подготовим несколько образов для теста:
   - скачаем любой публичный образ, например nginx:
     `docker pull nginx`
   - добавим для данного образа тэг, чтобы потом загрузить его в свой реестр:
     `docker tag nginx pstepanov90/nginx-signed:latest`
   - загрузим новый тэг в свой репозиторий:
     `docker push pstepanov90/nginx-signed:latest`
   - подпишем тэг, используя ключ из п.1:
     `cosign sign --key cosign.key pstepanov90/nginx-signed:latest`
     Теперь у нас в реестре есть подписанный образ nginx.
6. Для того, чтобы Neuvector мог контролировать подписанные образы, необходимо добавить свой *Registry*. Для этого перейдите на страницу *Assets* → *Registries* и нажмите кнопку *"Add"*.

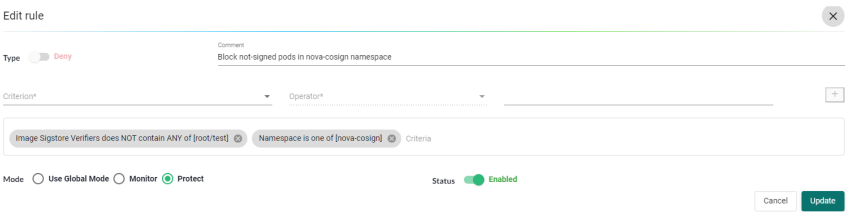
7. В открывшемся окне выберите тип реестра (в данном примере это *Docker registry*), задайте имя, введите адрес реестра, пользователя, пароль, фильтр (позволяет добавить только определённые образы) и настройки сканирования. Затем нажмите *"Submit"*.



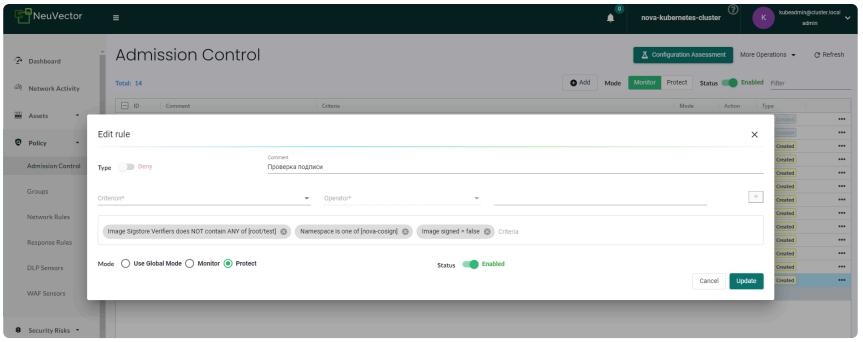
8. После добавления реестра необходимо его просканировать нажав кнопку *"Start scan"*.

9. Теперь создадим правило, запрещающее запуск неподписанных образов. Для этого перейдите на страницу *Policy* → *Admission Control* и нажмите кнопку *"Add"*.

   - В открывшемся окне введите в поле *Comment* описание для правила, например *"Проверка подписи"*.
   - Добавьте критерий проверки, для чего в поле *Criterion* выберите *"Image Sigstore Verifiers"*, в качестве оператора выберите *"does NOT contain ANY of"*, в поле *Value* выберите значение из п.4 (*root\test*) и нажмите *"+"*.
   - Добавьте еще один критерий проверки, для чего в поле *Criterion* выберите *"Name-space"*, в качестве оператора выберите *"contains one of"*, а в качестве *Value* выберите имя из п.3 (*Предварительные условия*), после чего нажмите *"+"*.
   - Убедитесь что режим *Mode* выбран *"Protect"* и *Status* переключен в состояние *"En-abled"*, после чего нажмите *Add*.



   Примечание: Если правило не подействует. то нужно добавить ещё одно условие в правило. В поле *Criterion* выберите *"Image signed"*, в поле *Value* выберите *"false"* и нажмите *"+"*.

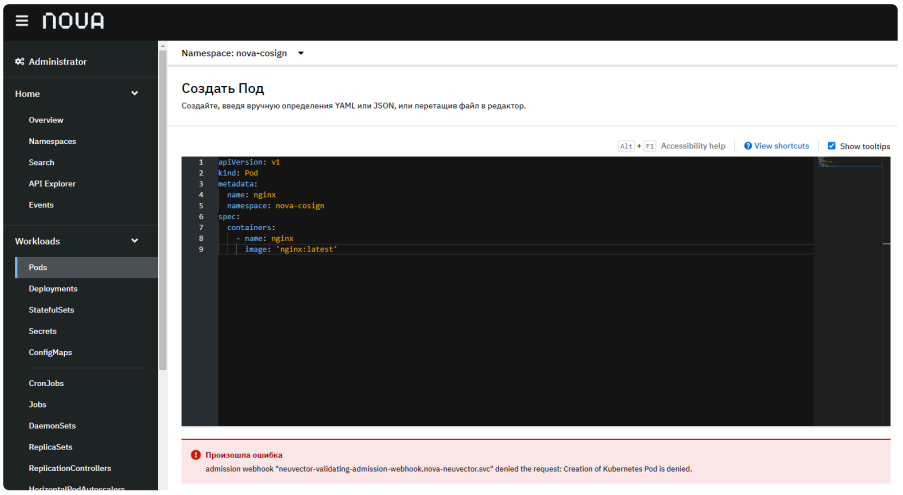


10. Теперь проверим работу данного правила. Для этого откройте в браузере веб-интерфейс Nova Console и нажмите на кнопку *"+"* в правом-верхнем углу.
11. В открывшемся окне введите следующий манифест и нажмите *"Create"*:

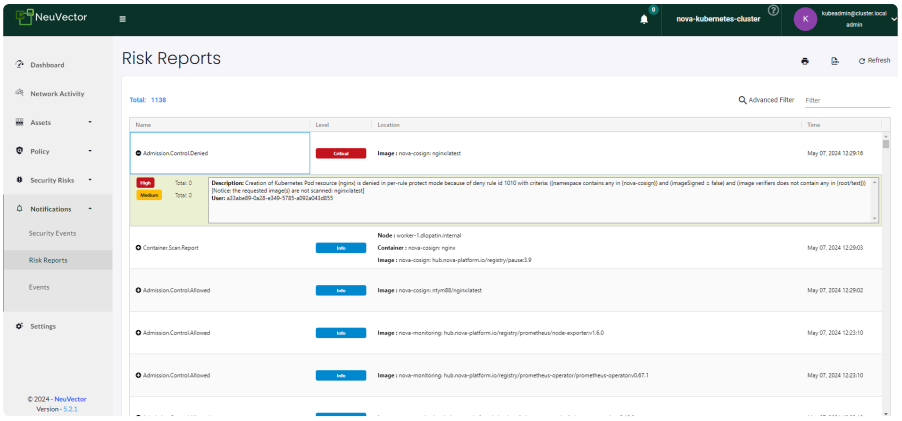
    Примечание: тут мы пытаемся создать под используя официальный репозиторий.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  namespace: nova-cosign
spec:
  containers:
    - name: nginx
      image: 'nginx:latest'
```

12. Убедитесь, что создание пода из неподписанного образа было заблокировано и на экран выведена соответствующая ошибка:



13. Проверьте в консоли Neuvector какое правило заблокировало создание пода.



14. Исправьте в манифесте имя образа на `nginx-signed` и нажмите *"Create"*:

YML

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  namespace: nova-cosign
spec:
  containers:
    - name: nginx
      image: 'pstepanov90/nginx-signed:latest'
```

Аутентификация и авторизация

Данный глоссарий содержит описание основных терминов, устоявшихся выражений, примитивов и определений, которые вы можете встретить в данной документации и при работе с механизмами аутентификации и авторизации в Nova Container Platform.

1. Глоссарий

1.1. Bearer token

bearer token

Bearer-токен используется для аутентификации в Kubernetes API. Токен устанавливается в HTTP-запрос в заголовок `Authorization: Bearer <token>`.

1.2. Cluster Role

Кластерная роль в RBAC Kubernetes, которая содержит набор правил, определяющих множество разрешений. Кластерная роль содержит только разрешающие правила и не может содержать запрещающих правил. Кластерная роль всегда определяет набор правил на уровне всего кластера, а не пространства имен (*Namespace*) в частности.

1.3. Cluster Role Binding

Объект *ClusterRoleBinding* (привязка кластерной роли) необходим для назначения каких-либо разрешений, определенных в роли, к пользователю или группе пользователей. В объекте *ClusterRoleBinding* содержится перечень субъектов (пользователей, групп, сервисных аккаунтов) и указание роли, которая им назначается. Объект *ClusterRoleBinding* используется только в контексте кластера и может ссылаться на любую кластерную роль.

1.4. Distinguished Name (DN)

Уникальное имя объекта, по которому данный объект может быть идентифицирован в каталоге LDAP-сервера.

1.5. Lightweight directory access protocol (LDAP)

LDAP является протоколом доступа к каталогам. С помощью данного протокола может быть запрошена информация о пользователе.

1.6. LDAPS (LDAP over SSL)

LDAP-подключения, защищенные с помощью SSL.

1.7. OpenID Connect

Протокол, позволяющий пользователю использовать единую учетную запись и сквозную аутентификацию (SSO) во множестве различных информационных систем.

1.8. Role

Роль в RBAC Kubernetes, которая содержит набор правил, определяющих множество разрешений. Роль содержит только разрешающие правила и не может содержать запрещающих правил. Роль всегда определяет набор правил на уровне пространства имен (*Namespace*).

1.9. RoleBinding

Объект *RoleBinding* (привязка роли) необходим для назначения каких-либо разрешений, определенных в роли, к пользователю или группе пользователей. В объекте *RoleBinding* содержится перечень субъектов (пользователей, групп, сервисных аккаунтов) и указание роли, которая им назначается. Объект *RoleBinding* используется только в контексте пространств имен (namespace) и может ссылаться на любую роль в том пространстве имен, в котором оно находится.

1.10. Аутентификация

Authentication

Процедура проверки подлинности пользователя различными методами, например, с помощью сравнения введенного и установленного пароля. Выполняется для того, чтобы гарантировать доступ к кластеру только подтвержденным пользователям.

1.11. Авторизация

authorization

Процедура предоставления прав идентифицированному пользователю. Выполняется для того, чтобы гарантировать выполнение пользователем только тех операций, которые ему

разрешены.

1.12. Группа

group

Набор пользователей. Группы удобно использовать, когда необходимо предоставить одинаковые привилегии нескольким пользователем одновременно.

1.13. Идентификация

Identification

Процедура проверки уникального идентификатора пользователя. Выполняется для того, чтобы однозначно определить существование пользователя в каталоге провайдера идентификации.

1.14. Клиент OAuth

OAuth client

Приложение или сервис, которому пользователь делегирует права доступа к своим данным на сервере OAuth. Используется для получения Bearer-токена.

1.15. Метод аутентификации

Auth method

Компонент StarVault, выполняющий задачи по аутентификации пользователей в каком-либо провайдере идентификации и установке пользовательского идентификатора с набором необходимых политик.

1.16. Пользователь

Сущность, которая может выполнять запросы к API.

1.17. Провайдер идентификации

Identity provider

Встроенный или внешний сервис, предназначенный для хранения и управления пользовательскими идентификационными данными, необходимыми для аутентификации пользователей в Nova Container Platform.

1.18. Сервер OAuth

OAuth server

Компонент StarVault в Nova Container Platform имеет встроенный OAuth-сервер, который отвечает за логику работы с провайдерами идентификации и выдачу новых токенов доступа.

1.19. Системные пользователи

Пользователи, созданные автоматически в ходе установки Nova Container Platform.

1.20. Служебные аккаунты

service accounts

Служебные аккаунты используются приложениями в кластере Kubernetes.

1.21. Управление доступом на основе ролей (RBAC)

role-based access control (RBAC)

Основной механизм управления разграничениями доступа к ресурсам кластера на основе ролей.

2. Содержание раздела