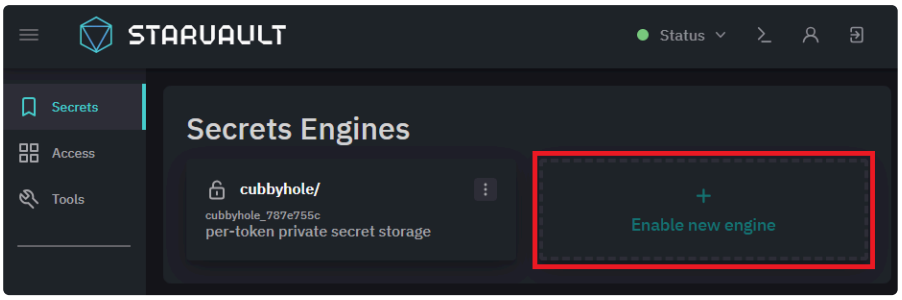
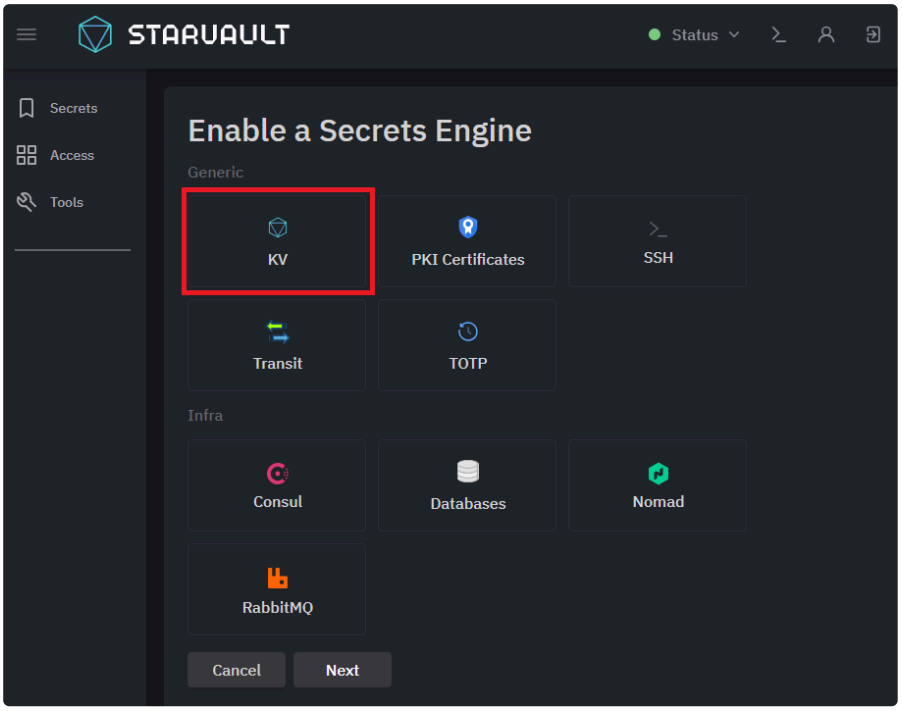
Использование секрета из StarVault

1. Добавление секрета и настройка доступа в StarVault

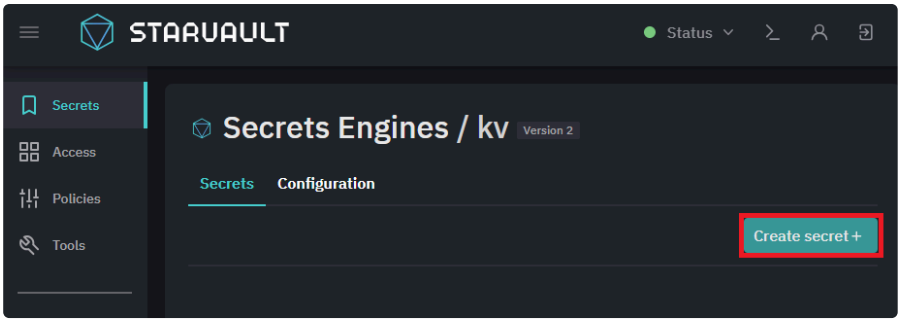
1. Зайдите в StarVault с *root token*.
2. Перейдите на страницу *Secrets* и нажмите на *Enable new engine +*, чтобы создать новое хранилище секретов.



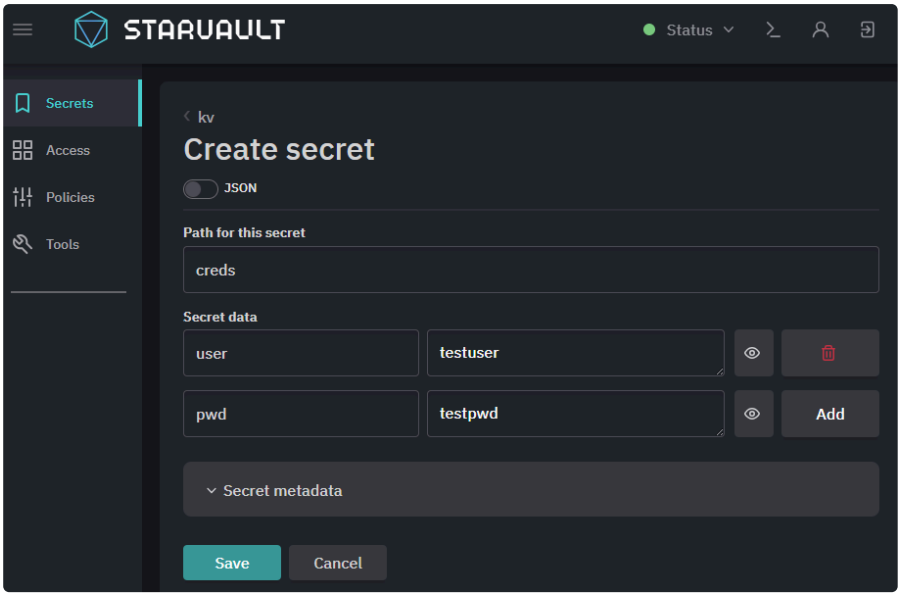
3. Выберите хранилище Key-Value.



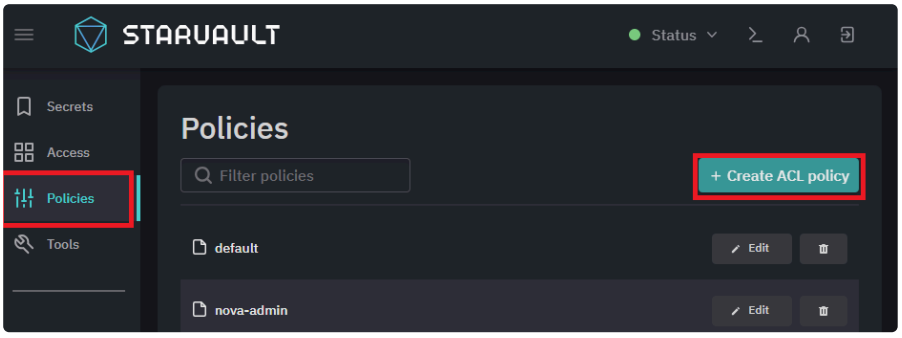
4. В новом хранилище нажмите на *Create secret +*, чтобы создать секрет.



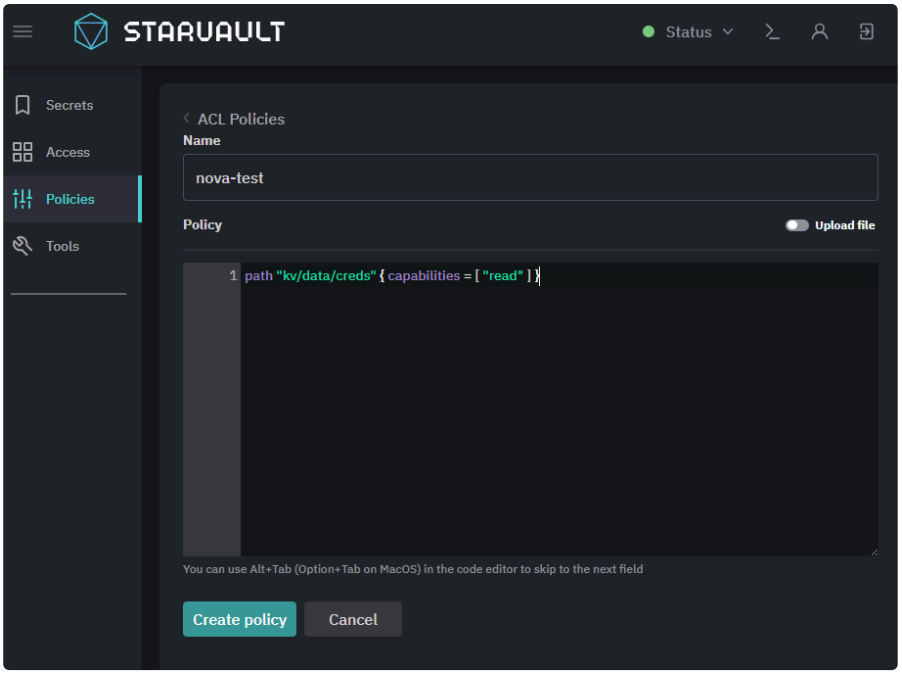
5. Укажите название для обращения к секрету и пары ключ-значение. Сохраните созданный секрет.



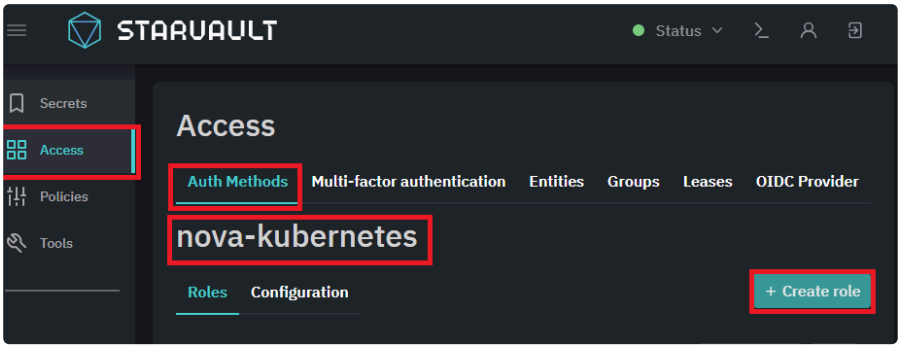
6. Перейдите на вкладку *Policies* для настройки правил доступа и нажмите на *Create ACL policy +*.



7. В этой политике указываются права на созданный ранее секрет.
   Пример правила: `path "kv/data/creds" { capabilities = [ "read" ] }`



8. Теперь перейдите на вкладку *Access* для настройки доступа к секрету используя созданную в прошлом пункте политику.
9. Перейдите в *Auth Methods* → *nova-kubernetes* и нажмите на *Create role +*.



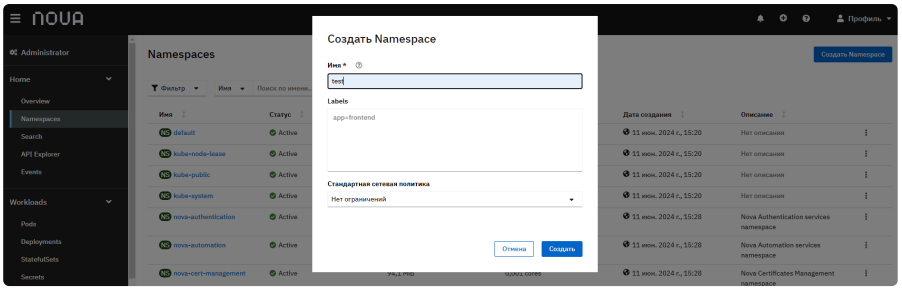
10. Заполните поля с соответствующими значениями:

```bash
Name: nova-test ①
Bound service account names: nova-test ②
Bound service account namespaces: test ③
Generated Tokens Policies: nova-test ④
```

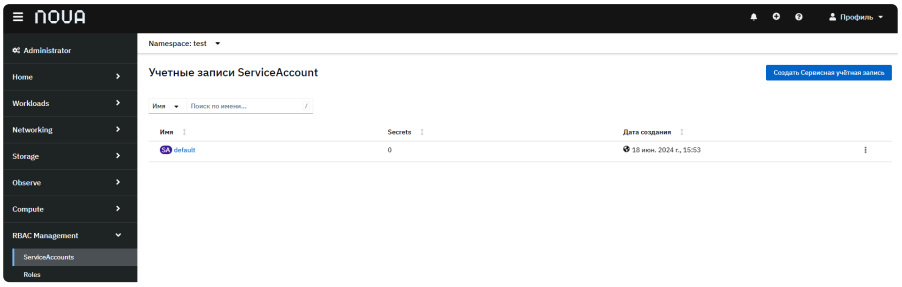
1. Имя роли. Может быть любым.
2. Имя сервисного аккаунта, который будет создан в Nova Console для доступа к секрету.
3. Пространство имён в Nova Console, откуда можно обращаться к секрету.
4. Созданная ранее политика доступа к секрету.

2. Настройка доступа со стороны Nova Console

1. Перейдите на страницу Nova Console и создайте новое пространство имён. Перейдите на вкладку *Home → Namespaces* и нажмите на *Создать Namespace*.



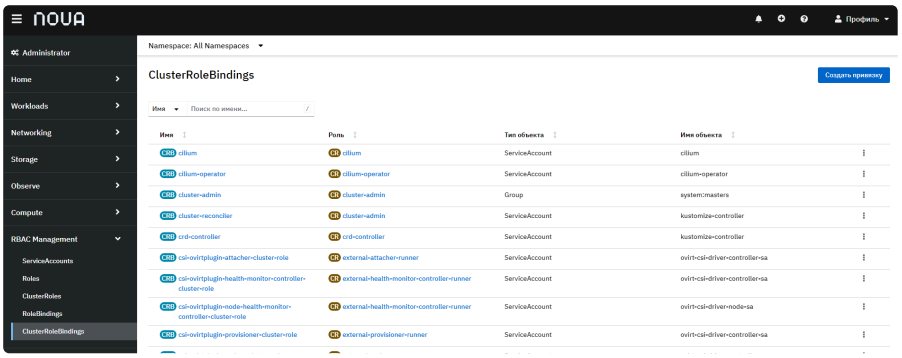
2. Создайте сервисный аккаунт в новом пространстве имён, который будет иметь доступ к созданному нами секрету в StarVault.
   Перейдите на вкладку *RBAC Management → ServiceAccounts* и нажмите на *Создать Сервисная учётная запись*.



   Далее представлен пример манифеста для сервисного аккаунта. Нужно учитывать, что имя сервисного аккаунта должно совпадать с именем, которое было указано при настройке доступа в StarVault.

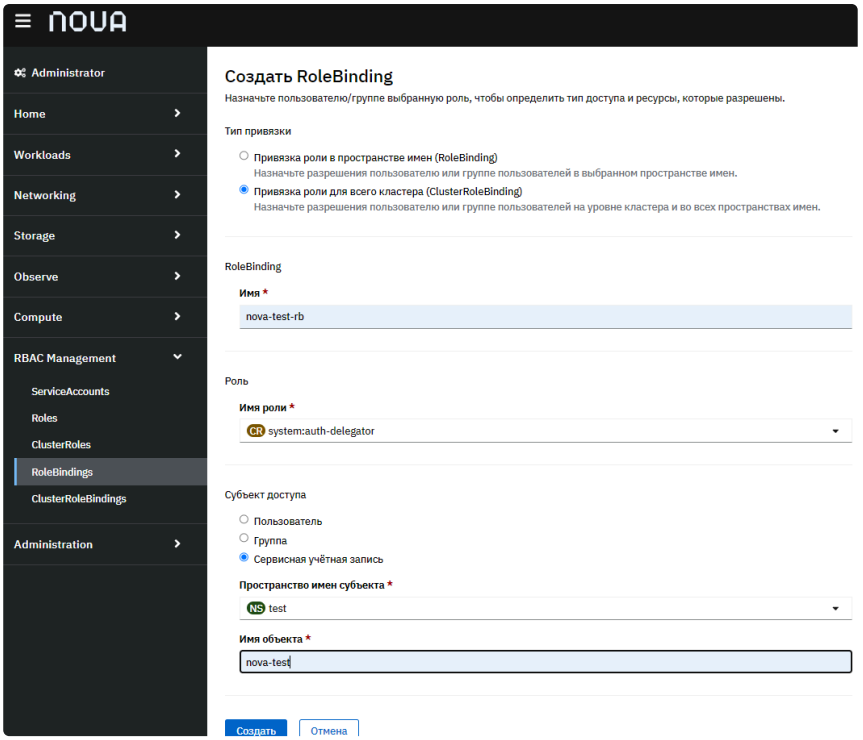
```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: nova-test
  namespace: test
```

3. Создайте привязку к уже существующей роли. Перейдите на вкладку *RBAC Management → ClusterRoleBindings* и нажмите *Создать привязку*.



4. При создании Role Binding указываются следующие параметры:
   - привязка нужна для всего кластера;
   - роль для привязки - `system:auth-delegator`

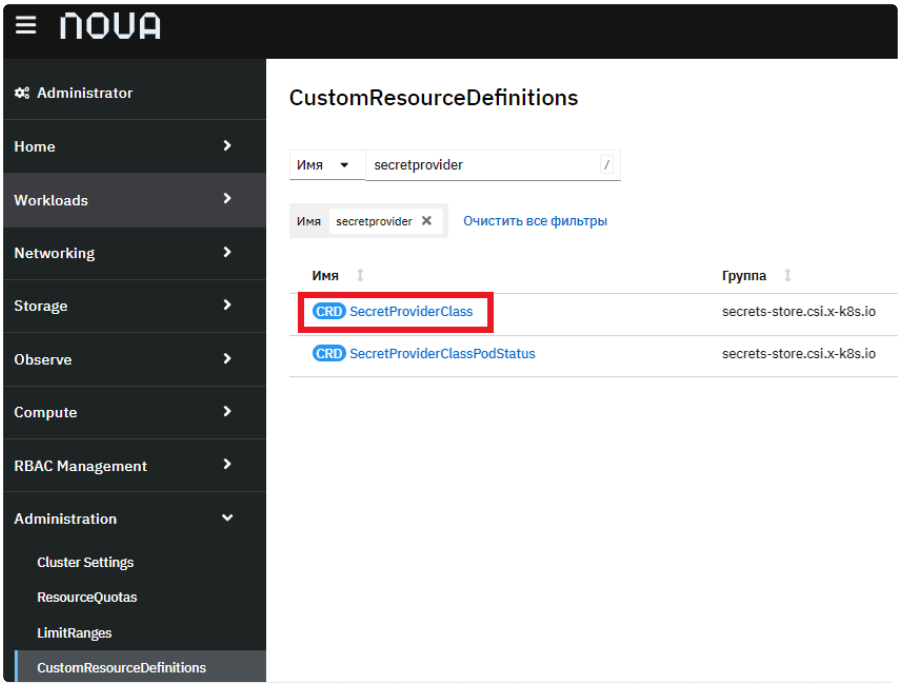
- созданный ранее сервисный аккаунт в пространстве имён `test`



3. Использование секрета в контейнере

3.1. Использование секрета в контейнере с копированием секрета из StarVault в Nova Console

1. Перейдите на страницу Nova Console на вкладку *Administration* → *CustomResourceDefinitions*, найдите *SecretProviderClass* и зайдите в него.



2. Перейдите на вкладку *Экземпляры*, нажмите *Создать SecretProviderClass* и используйте следующий манифест:

```yaml
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: nova-test
  namespace: test
spec:
  provider: vault
  parameters:
    objects: |
      - objectName: "pwd"
        secretPath: "kv/data/creds"
        secretKey: "pwd"
      - objectName: "user"
        secretPath: "kv/data/creds"
        secretKey: "user"
    roleName: nova-test
  secretObjects:
    - data:
        - key: password
          objectName: pwd
        - key: username
          objectName: user
      secretName: example-secret ①
      type: Opaque
```

1. Параметр `secretName` создаст секрет в Nova Console с указанным именем и данными из секрета в StarVault.

3. Создайте под, в который будет прокинут секрет из StarVault. Используйте следующий манифест:

```yaml
apiVersion: v1
kind: Pod
metadata:
  annotations: ①
    vault.security.banzaicloud.io/vault-path: nova-kubernetes
    vault.security.banzaicloud.io/vault-role: nova-test
    vault.security.banzaicloud.io/vault-tls-secret: nova-oauth-ca
  name: example
  namespace: test
spec:
  containers:
    - name: httpd
      image: 'httpd:latest'
      ports:
        - containerPort: 8080
      env:
```

```
        - name: USERNAME
          valueFrom:
            secretKeyRef:
              name: example-secret ②
              key: username
        - name: PASSWORD
          valueFrom:
            secretKeyRef:
              name: example-secret ②
              key: password
      volumeMounts:
        - name: data
          mountPath: /data
        - name: secrets-store-inline
          mountPath: "/mnt/secrets-store"
  serviceAccountName: nova-test
  serviceAccount: nova-test
  volumes:
    - name: data
      emptyDir: {}
    - name: secrets-store-inline
      csi:
        driver: secrets-store.csi.k8s.io
        readOnly: true
        volumeAttributes:
          secretProviderClass: "nova-test"
```

1. Аннотация нужна для указания кластеру, что будет использоваться доступ к StarVault.
2. Секрет `example-secret` создавать не нужно. Он создастся автоматически.

4. Проверьте, что создался секрет. Зайдите в терминал пода и убедитесь, что появились переменные.

3.2. Использование секрета в контейнере БЕЗ копированием секрета из StarVault в Nova Console

1. Создайте под, в который будет прокинут секрет из StarVault. Используйте следующий манифест:

YAML

```yaml
apiVersion: v1
kind: Pod
metadata:
  annotations: ①
    vault.security.banzaicloud.io/vault-path: nova-kubernetes
    vault.security.banzaicloud.io/vault-role: nova-test
    vault.security.banzaicloud.io/vault-tls-secret: nova-oauth-ca
  name: example
```

```
  namespace: test
spec:
  serviceAccountName: nova-test ②
  serviceAccount: nova-test ②
  containers:
  - name: main-container
    image: busybox
    command: [ "sh", "-c", "tail -f /dev/null" ]
    volumeMounts:
    - name: config-volume
      mountPath: /etc/config
  initContainers: ③
  - name: init-container
    image: busybox
    command:
      - sh
      - -c
      - 'echo $USERNAME > /etc/config/my-env-file'
    env:
    - name: USERNAME
      value: 'vault:kv/data/creds#user'
    volumeMounts:
    - name: config-volume
      mountPath: /etc/config
  volumes:
  - name: config-volume
    emptyDir: {}
```

1. Аннотация нужна для указания кластеру, что будет использоваться доступ к StarVault.
2. Параметры `serviceAccount` и `serviceAccountName` нужны для доступа пода к секрету в StarVault.
3. В под автоматически добавится ещё один *initContainer* `copy-vault-env`, который копирует секрет из StarVault.

2. Заёдите в терминал пода и проверьте содержимое файла `/etc/config/my-env-file`, а также, что данные не хранятся в открытом виде в переменной `echo $USERNAME`.

Управление секретами платформы

Данный раздел содержит статьи полезные для управления секретами в Nova Container Platform.

1. Подключение к StarVault

Для выполнения различных задач по администрированию аутентификации и авторизации в Nova Container Platform требуется подключение к StarVault с привилегиями администратора.

Вы можете получить адрес и подключиться к StarVault, используя процедуру ниже.

Необходимые условия

- ✓ У вас есть токен доступа к хранилищу секретов StarVault или учетная запись с привилегиями `root`.
- ✓ У вас есть доступ к Kubernetes API с привилегиями администратора кластера.

Процедура

1. Получите адрес StarVault:

   ▸ **Web UI**

   ▸ **CLI**

2. Перейдите по полученному адресу и авторизуйтесь в StarVault, указав токен доступа к хранилищу секретов StarVault или параметры собственной учетной записи с привилегиями `root`.

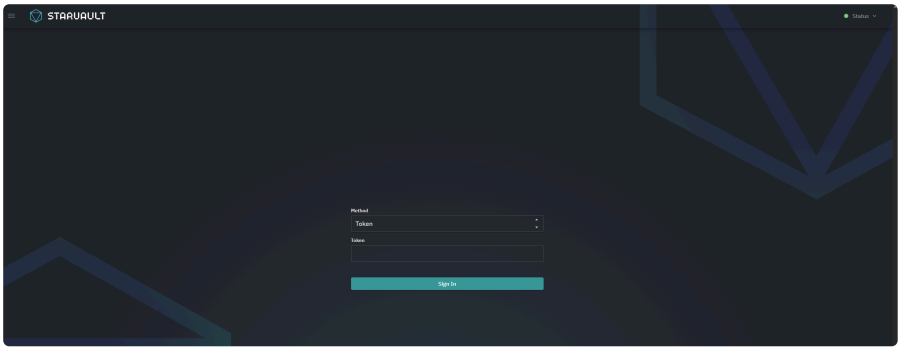


Рисунок 1. Страница входа в StarVault

2. Приложения OAuth

Данный раздел содержит статьи полезные для управления OAuth приложениями в Nova Container Platform.

2.1. Настройка доступа к приложениям OAuth

Доступ к каком-либо компоненту Nova Container Platform для конечного пользователя выполняется по протоколу OpenID Connect (OIDC). Поскольку StarVault является основным OIDC-провайдером в платформе, то для доступа к приложениями, зарегистрированным в StarVault, необходимо выполнить процедуру назначения приложения определенной группе пользователей или конкретным пользователям.

2.1.1. Настройка назначений

Для настройки назначения приложений воспользуйтесь процедурой ниже.

1. Откройте веб консоль StarVault.
2. Перейдите в раздел **Access**, далее **OIDC Provider**.
3. Перейдите в список **Assignments** и нажмите **Create assignment**.
    - В поле **Name** укажите имя назначения. Это может быть, например, имя группы пользователей в каталоге LDAP-сервера.
    - В поле **Entities** укажите ранее созданные сущности пользователей.
    - В поле **Groups** укажите ранее созданную группу.
    - Нажмите **Create**, чтобы создать назначение.

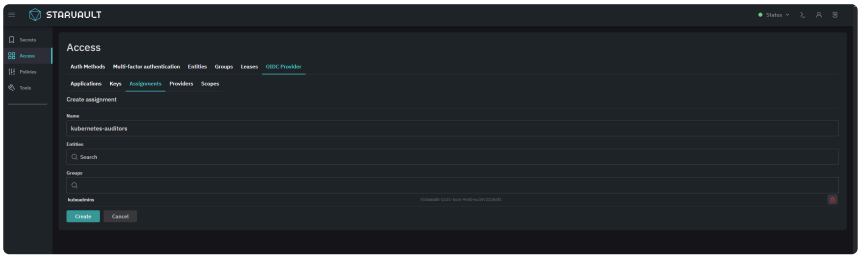


Рисунок 2. Настройка назначений в StarVault

2.1.2. Привязка назначений к приложениям

Для привязки назначения к приложению воспользуйтесь процедурой ниже.

1. Откройте веб консоль StarVault.
2. Перейдите в раздел **Access**, далее **OIDC Provider**.
3. Перейдите в список **Applications** и выберете необходимое приложение. Например, для добавления пользователям возможности выполнять аутентификацию в утилите kubectl

или веб-интерфейсе Nova Console, выберете приложение `oidc-kubernetes-client`.

- Нажмите **Edit application**.
- В разделе **Assign access** добавьте ранее настроенное назначение в список разрешенных.
- Нажмите **Update**, чтобы обновить параметры назначения.

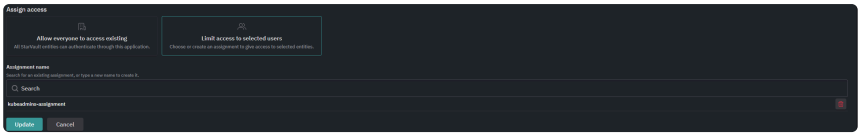


Рисунок 3. Настройка назначений в StarVault

2.2. Настройка доступа до OAuth приложений с использованием LDAP

В данном разделе описывается процедура использования провайдера идентификации в *StarVault* по протоколу LDAP для доступа к OAuth приложениям на примере **NeuVector**.

2.2.1. Необходимые условия

- ✓ У вас есть токен доступа к хранилищу секретов *StarVault* с привилегиями `root`.
- ✓ У вас есть доступ к кластеру с учетной записью, имеющей роль `cluster-admin` в Kubernetes.
- ✓ Модуль NeuVector установлен в вашем кластере.
- ✓ Провайдер идентификации LDAP подключен в *StarVault*.

2.2.2. Настройка доступа в StarVault

1. В веб-интерфейсе StarVault выберите вкладку **Access**, далее **Groups**.
2. Создайте группу и алиас.
   - Нажмите **Create group**.
     - В поле **Name** укажите имя группы так же, как группа названа в каталоге LDAP-сервера.
     - В поле **Type** укажите **External**.
     - Нажмите **Create**, чтобы создать группу. Откроется страница с параметрами созданной группы.
   - Нажмите **Add alias**.
     - В поле **Name** укажите имя алиаса так же, как группа названа в каталоге LDAP-сервера.
     - В поле **Auth Backend** выберете имя метода аутентификации LDAP.

- Нажмите **Create**, чтобы создать алиас.

Для удобства и простоты администрирования рекомендуется использовать один алиас на одну сущность StarVault.

3. В веб-интерфейсе StarVault перейдите на вкладку **Access → OIDC Provider → Assignments**.
4. Нажмите **Create assignment**
   - В поле **Name** укажите любое имя, например название приложения к которому предоставляется доступ.
   - В поле **Groups** выберите группу созданную ранее.
   - Нажмите **Create**.
5. Перейдите на вкладку **Access → OIDC Provider → Applications**.
   - Выберите нужное приложение. В нашем случае `oidc-auth-neuvector`.
   - Нажмите **Edit application**
   - В поле **Assignment name** выберите ранее созданный Assignment.
   - Нажмите **Update**.
6. Перейдите на вкладку **Access → OIDC Provider → Scopes**.
   - Нажмите на **Edit** у параметра `email`.
   - Измените значение на `{ "email": {{identity.entity.name}} }`
   - Нажмите **Update**

2.2.3. Настройка доступа в NeuVector

1. В веб-интерфейсе Nova Container Platform выберите вкладку **Ресурсы**, далее **Secrets**.
2. Скопируйте значение `oidcinitcfg.ctmpl` из секрета `neuvector-init-template`.
3. Перейдите на вкладку **Администрирование → CustomResourceDefinitions** и выберите **Kustomization**
4. Перейдите на вкладку **Инстансы** и выберите `nova-release-neuvector-main`
5. На вкладке **YAML** добавьте патч в блок **spec**. Значение файла `oidcinitcfg.ctmpl` должно быть аналогичным значению из пункта 2.
   В блок **group_mapped_roles** добавьте соответствие нужной группы и роли.

Всего в NeuVector есть 3 группы по умолчанию.

- admin
- reader
- ciops

Пример

```yaml
patches:
  - patch: |-
      kind: Secret
      apiVersion: v1
      metadata:
        name: neuvector-init-template
        namespace: nova-neuvector
      stringData:
        oidcinitcfg.ctmpl: |
          {{- with secret "identity/oidc/provider/nova" }}
          always_reload: false
          Issuer: {{ .Data.issuer }}
          {{- end -}}
          {{ with secret "identity/oidc/client/oidc-auth-neuvector" }}
          Client_ID: {{ .Data.client_id }}
          Client_Secret: {{ .Data.client_secret }}
          {{ end -}}
          GroupClaim: groups
          Scopes:
            - openid
            - profile
            - email
            - groups
          Enable: true
          Default_Role:
          group_mapped_roles:
            - group: kubeadmins ①
              global_role: admin
            - group: global-admins ②
              global_role: admin
          group_claim: groups
    target:
      kind: Secret
      name: neuvector-init-template
      namespace: nova-neuvector
```

1. Настройка доступа для kubeadmins. Изменять не нужно.
2. Пример добавления группы с определённой ролью.

2.2.4. Проверка

1. Убедитесь, что под `neuvector-controller-pod-0` работает.
2. Зайдите в веб-интерфейс NeuVector с использованием LDAP подключения.

CSI для S3

Это интерфейс контейнерного хранилища CSI для S3 (или S3-совместимого) хранилища. Он может динамически выделять бакеты и монтировать их с помощью fuse mount в любой контейнер.

1. Предварительные условия

- ✓ У вас есть доступ к кластеру с учетной записью, имеющей роль `cluster-admin` в Kubernetes.
- ✓ Вы установили утилиту `kubectl` для работы с Kubernetes.
- ✓ У вас есть доступ в S3 бакет с правами управления хранилища и настройки доступа к API.

2. Автоматическая установка

Установите Helm chart опубликованный в GitHub:

BASH

```
helm repo add yandex-s3 https://yandex-cloud.github.io/k8s-csi-s3/charts
helm install csi-s3 yandex-s3/csi-s3
```

3. Ручная установка

1. Клонируйте репозиторий:

   BASH

   ```
   git clone https://github.com/yandex-cloud/k8s-csi-s3.git
   ```

2. Создайте секрет с учетными данными S3.

   YAML

   ```
   apiVersion: v1
   kind: Secret
   metadata:
     name: csi-s3-secret
     namespace: kube-system
   stringData:
     accessKeyID: <project>:<client_id> ①
     secretAccessKey: <client_secret_key> ①
     endpoint: https://s3.k2.cloud ②
   ```

① Данные API.

② Ссылка на S3 endpoint в выбранном облаке.

3. Перейдите в репозиторий и установите драйвер в кластер.

```
kubectl create -f deploy/kubernetes/provisioner.yaml
kubectl create -f deploy/kubernetes/driver.yaml
kubectl create -f deploy/kubernetes/csi-s3.yaml
```

4. Создайте *Storage Class*.

```
kubectl create -f deploy/kubernetes/examples/storageclass.yaml
```

4. Тестирование

1. Создайте *PVC* используя новый *Storage Class*:

```
kubectl create -f deploy/kubernetes/examples/pvc.yaml
```

2. Проверьте привязки *PVC*:

```
$ kubectl get pvc csi-s3-pvc
NAME         STATUS    VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
csi-s3-pvc   Bound     pvc-c5d4634f-8507-11e8-9f33-0e243832354b   5Gi
RWO            csi-s3         9s
```

3. Создайте тестовый под, в который монтируется том:

```
kubectl create -f deploy/kubernetes/examples/pod.yaml
```

4. Проверьте состояние пода. Если он запущен, значит проблем с интеграцией нет.

5. Убедитесь, что там смонтирован:

```
$ kubectl exec -ti csi-s3-test-nginx bash
$ mount | grep fuse
pvc-035763df-0488-4941-9a34-f637292eb95c: on /usr/share/nginx/html/s3 type
fuse.geesefs
(rw,nosuid,nodev,relatime,user_id=65534,group_id=0,default_permissions,allow
_other)
$ touch /usr/share/nginx/html/s3/hello_world
```

5. Дополнительная конфигурация

5.1. Бакет

По умолчанию, csi-s3 создаст новый бакет для каждого тома. Имя бакета будет соответствовать ID тома. Если вы хотите, чтобы ваши тома находились в заранее созданном бакете, то вы можете просто указать бакет в параметрах *Storage Class*:

```yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: csi-s3-existing-bucket
provisioner: ru.yandex.s3.csi
parameters:
  mounter: geesefs
  options: "--memory-limit 1000 --dir-mode 0777 --file-mode 0666"
  bucket: some-existing-bucket-name
```

Если имя бакета указано, он будет создан, если не существует на бэкенде. Каждый том получит свой префикс в бакете, соответствующий идентификатору тома. При удалении тома также будет удален только префикс.

5.2. Статическая инициализация

Если вы хотите смонтировать уже существующий бакет или префикс внутри уже существующего бакета и не хотите, чтобы csi-s3 удалял его при удалении PV, вы можете использовать статическую инициализацию.

Для этого нужно исключить *storageClassName* в *PersistentVolumeClaim* и вручную создать *PersistentVolume* с соответствующим *claimRef*, как в следующем примере: deploy/kubernetes/examples/pvc-manual.yaml.

6. Устранение неполадок

6.1. Проблемы при создании PVC

Проверьте журналы провайдера:

```bash
kubectl logs -l app=csi-provisioner-s3 -c csi-s3
```

6.2. Проблемы при создании контейнеров

1. Убедитесь, что для параметра `MountPropagation` не установлено значение `false`.
2. Проверьте журналы S3-драйвера:

```bash
kubectl logs -l app=csi-s3 -c csi-s3
```