Упрощение управления политиками с помощью шаблонов политик ACL

StarVault работает по стандарту **secure by default**, и поэтому политика **no permissions** не дает никаких прав в системе. Поэтому необходимо создать политику, которая будет управлять поведением клиентов и обеспечивать контроль доступа на основе ролей (RBAC), определяя привилегии доступа (авторизацию).

Поскольку все в StarVault основано на путях, авторы политик должны знать все существующие пути, а также пути, которые будут созданы.

В учебном пособии "Политики" рассказывается о создании политик ACL в StarVault.

1. Задача

Единственным способом указать нестатические пути в ACL-политиках было использование символов (*****) в конце путей. Или использовать знак плюс (**+**) для подстановочного символа одного каталога.

HCL

```
path "transit/keys/*" {
  capabilities = [ "read" ]
}

path "secret/+/apikey" {
  capabilities = [ "create", "read", "update", "delete", "list" ]
}
```

Это делает многие задачи управления и делегирования сложными. Например, разрешение пользователю изменять свой собственный пароль путем обращения к конечной точке `auth/userpass/users/<имя_пользователя>/password` может потребовать политики для *каждого* пользователя.

2. Решение

В StarVault есть возможность шаблонизации ACL, позволяющая использовать подмножество пользовательской информации в путях политики ACL.

Эта функция использует идентификаторы StarVault Identities для введения значений в пути политики ACL.

3. Предпоссылки

Для выполнения задач, описанных в этом руководстве, необходимо иметь среду с StarVault. Для установки StarVault обратитесь к руководству по установке StarVault.

4. Введение в сценарий

Предположим, что заданы следующие требования к политике:

- Каждый *user* может выполнять все операции над выделенным ему секретным путем ключа/значения (`user-kv/data/<имя_пользователя>`).
- Education *group* имеет специальное хранилище key/value для каждого региона, где все операции могут выполняться членами группы (group-kv/data/education/<region>).
- Члены *group* могут обновлять информацию о группе, например метаданные о ней (`identity/group/id/<group_id>`).

В этом уроке вам предстоит выполнить следующие действия:

1. Лабораторная установка
2. Создание типовых политик ACL
3. Настройка сущности и группы
4. Протестируйте шаблонизацию ACL

5. Настройка лаборатории

5.1. StarVault

1. Откройте терминал и запустите сервер StarVault dev с `root` в качестве корневого токена.

   ```
   $ starvault server -dev -dev-root-token-id=root
   ```

   По умолчанию сервер StarVault dev работает по адресу `127.0.0.1:8200`. Сервер также инициализируется и снимается с печати.

   Небезопасная операция

   Не запускайте сервер StarVault dev в производстве. Этот подход используется здесь только для упрощения процесса снятия печати в данной демонстрации.

2. Экспортируйте переменную окружения для `starvault` CLI, чтобы обратиться к серверу StarVault.

```
$ export STARVAULT_ADDR=http://127.0.0.1:8200
```
BASH

3. Экспортируйте переменную окружения для starvault CLI, чтобы аутентифицироваться на сервере StarVault.

```
$ export STARVAULT_TOKEN=root
```
BASH

Для этих задач можно использовать `root` токен StarVault. Однако рекомендуется использовать корневой токен только для первоначальной настройки или в чрезвычайных ситуациях. Лучше всего использовать токены с соответствующим набором политик в зависимости от вашей роли в организации.

Сервер StarVault готов.

5.2. Требования к политике

Поскольку в этом руководстве демонстрируется создание политики `admin`, по возможности войдите в систему с токеном `root`. В противном случае обратитесь к требованиям к политике в учебном пособии "Политики".

6. Создание типовых политик ACL

Авторы политики могут передавать путь политики, содержащий двойные фигурные скобки в качестве разделителей шаблона: `{{<parameter>}}`.

6.1. Доступные параметры шаблонов

Name	Description
`identity.entity.id`	Идентификатор организации
`identity.entity.name`	Название организации
`identity.entity.metadata.<metadata key>`	Метаданные, связанные с сущностью по заданному ключу
`identity.entity.aliases.<mount accessor>.id`	Идентификатор псевдонима сущности для данного монтирования

Name	Description
`identity.entity.aliases.<mount accessor>.name`	Имя псевдонима сущности для данного монтирования
`identity.entity.aliases.<mount accessor>.metadata.<metadata key>`	Метаданные, связанные с псевдонимом для заданного монтирования и ключа метаданных
`identity.entity.aliases.<mount accessor>.custom_metadata.<custom metadata key>`	Пользовательские метаданные, связанные с псевдонимом сущности
`identity.groups.ids.<group id>.name`	Имя группы для заданного ID группы
`identity.groups.names.<group name>.id`	ID группы для заданного имени группы
`identity.groups.ids.<group id>.metadata.<metadata key>`	Метаданные, связанные с группой для данного ключа
`identity.groups.names.<group name>.metadata.<metadata key>`	Метаданные, связанные с группой для данного ключа

Группы идентификации не привязаны напрямую к токену, и сущность может быть связана с несколькими группами. Поэтому, чтобы сослаться на группу, необходимо указать **group ID** или **group name** (например, `identity.groups.ids.59f001d5-dd49-6d63-51e4-357c1e7a4d44.name`).

6.2. Пример

Политика позволяет пользователям изменять свой собственный пароль, если имя пользователя и пароль определены в методе `auth userpass`. Значение аксессора монтирования (`auth_userpass_6671d643` в данном примере) можно прочитать из конечной точки `sys/auth`.

HCL

```
path
"auth/userpass/users/{{identity.entity.aliases.auth_userpass_6671d643.name}}" {
  capabilities = [ "update" ]
  allowed_parameters = {
    "password" = []
  }
}
```

6.2.1. CLI команда

1. Напишите политику шаблона пользователя (`user-tmpl.hcl`).

   user-tmpl.hcl

```
$ tee user-tmpl.hcl <<EOF
# Grant permissions on user specific path
path "user-kv/data/{{identity.entity.name}}/*" {
    capabilities = [ "create", "update", "read", "delete", "list" ]
}

# For Web UI usage
path "user-kv/*" {
  capabilities = ["list"]
}
EOF
```

2. Напишите политику группового шаблона (`group-tmpl.hcl`).

group-tmpl.hcl

```
$ tee group-tmpl.hcl <<EOF
# Grant permissions on the group specific path
# The region is specified in the group metadata
path "group-
kv/data/education/{{identity.groups.names.education.metadata.region}}/*" {
    capabilities = [ "create", "update", "read", "delete", "list" ]
}

# Group member can update the group information
path "identity/group/id/{{identity.groups.names.education.id}}" {
  capabilities = [ "update", "read" ]
}

# For Web UI usage
path "group-kv/*" {
  capabilities = ["list"]
}

path "identity/group/id" {
  capabilities = [ "list" ]
}
EOF
```

3. Создайте новую политику под названием `user-tmpl` .

```
$ starvault policy write user-tmpl user-tmpl.hcl
```

4. Создайте новую политику под названием `group-tmpl` .

```
$ starvault policy write group-tmpl group-tmpl.hcl
```

6.2.2. Вызов API с помощью cURLs

1. Чтобы создать политику, используйте конечную точку `/sys/policies/acl`:

```
$ curl --header "X-Vault-Token: <TOKEN>" \
       --request PUT \
       --data <PAYLOAD> \
       <STARVAULT_ADDR>/v1/sys/policies/acl/<POLICY_NAME>
```

Где `<TOKEN>` - это ваш действительный токен, а `<PAYLOAD>` включает в себя имя политики и строковую политику.

6.2.2.1. StarVault

1. Создайте полезную нагрузку API-запроса, содержащую структурированную политику для `user-tmpl`.

```
$ tee payload_user.json <<EOF
{
  "policy": "# Grant permissions on user specific path\npath \"user-
kv/data/{{identity.entity.name}}/*\" {\n    capabilities = [ \"create\",
\"update\", \"read\", \"delete\", \"list\" ]\n}\n\n# For Web UI usage\npath
\"user-kv/*\" {\n  capabilities = [\"list\"]\n}"
}
EOF
```

2. Создайте политику `user-tmpl`.

```
$ curl --header "X-Vault-Token: $STARVAULT_TOKEN" \
       --request PUT \
       --data @payload_user.json \
       $STARVAULT_ADDR/v1/sys/policies/acl/user-tmpl
```

3. Создайте полезную нагрузку API-запроса, содержащую структурированную политику для `group-tmpl`.

```
$ tee payload_group.json <<EOF
{
   "policy": "# Grant permissions on the group specific path\npath \"group-
kv/data/education/{{identity.groups.names.education.metadata.region}}/*\"
{\n    capabilities = [ \"create\", \"update\", \"read\", \"delete\",
\"list\" ]\n}\n\n# Group member can update the group information\npath
\"identity/group/id/{{identity.groups.names.education.id}}\" {\n
capabilities = [ \"update\", \"read\" ]\n}\n\n# For Web UI usage\npath
\"group-kv/*\" {\n  capabilities = [\"list\"]\n}\n\npath
\"identity/group/id\" {\n  capabilities = [ \"list\" ]\n}"
}
EOF
```

4. Создайте политику `group-tmpl`.

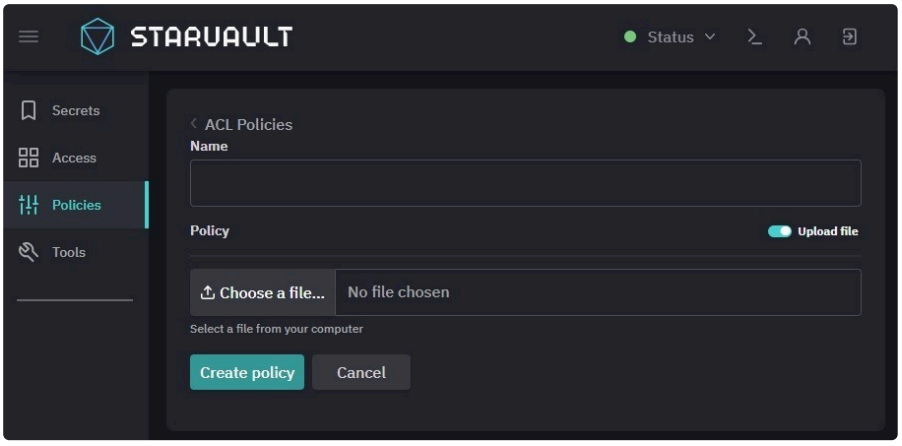
```
$ curl --header "X-Vault-Token: $STARVAULT_TOKEN" \
       --request PUT \
       --data @payload_group.json \
       $STARVAULT_ADDR/v1/sys/policies/acl/group-tmpl
```

BASH

6.2.3. Веб-интерфейс

Откройте веб-браузер и запустите пользовательский интерфейс StarVault (например, `http://127.0.0.1:8200/ui`), затем войдите в систему.

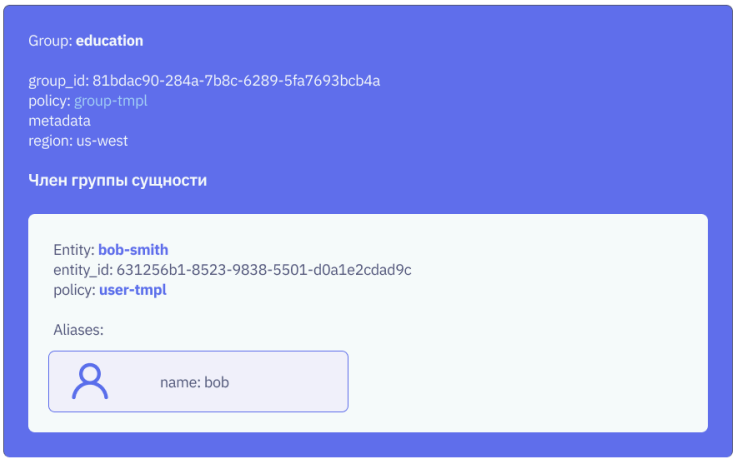
1. Нажмите на вкладку **Policies**, выберите **Create ACL policy**
2. Выберите пункт **Upload file** и нажмите кнопку **Choose a file**, чтобы выбрать файл `user-tmpl.hcl`, который вы записали на шаге 1. Это загрузит политику и установит для Name значение `user-tmpl`.



3. Нажмите на кнопку **Create Policy**.
4. Повторите шаги для создания политики `group-tmpl`.

7. Настройка сущности и группы

Создадим сущность `bob_smith` с пользователем `bob` в качестве псевдонима сущности. Также создадим группу `education` и добавим сущность `bob_smith` в качестве ее члена.



В этом шаге демонстрируются только команды CLI и веб-интерфейса для создания сущностей и групп. Если вам нужна полная информация, обратитесь к учебнику Identity - Entities and Groups.

7.1. CLI команда

Следующая команда использует инструмент `jq` для разбора вывода JSON.

1. Включите метод аутентификации `userpass`.

```
$ starvault auth enable userpass
```

2. Создайте нового пользователя `bob` с паролем "training".

```
$ starvault write auth/userpass/users/bob password="training"
```

3. Получите указатель монтирования `userpass` и сохраните его в файле с именем `accessor.txt`.

```
$ starvault auth list -format=json | jq -r '.["userpass/"].accessor' >
accessor.txt
```

4. Создайте сущность `bob_smith` и сохраните идентификатор сущности в файле `entity_id.txt`.

```
$ starvault write -format=json identity/entity name="bob_smith" \
        policies="user-tmpl" \
```

```
        metadata=team="Processor" \
        | jq -r ".data.id" > entity_id.txt
```

5. Добавьте псевдоним для сущности `bob_smith`.

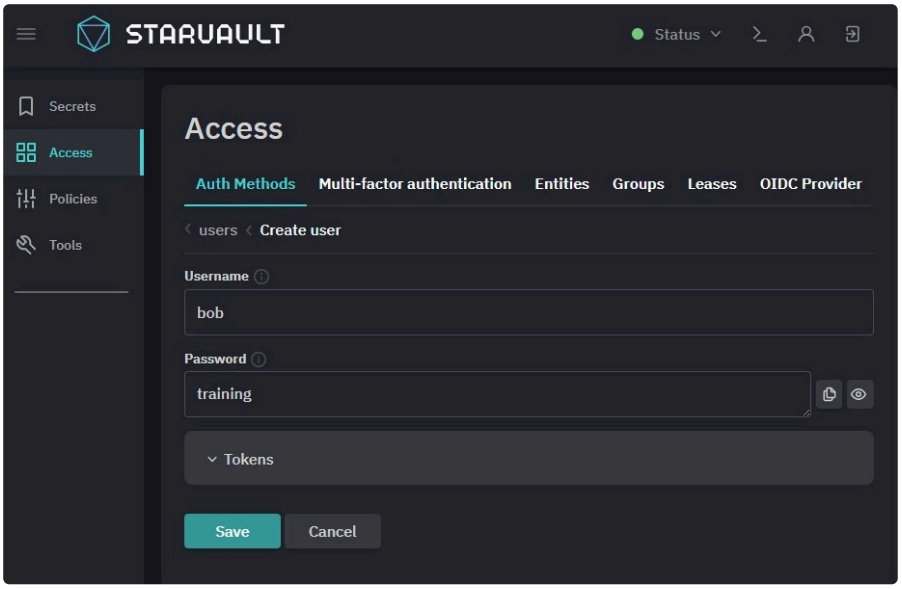
```
$ starvault write identity/entity-alias name="bob" \
      canonical_id=$(cat entity_id.txt) \
      mount_accessor=$(cat accessor.txt)
```

6. Наконец, создайте группу education и добавьте в нее сущность `bob_smith`. Сохраните сгенерированный идентификатор группы в файле `group_id.txt`.

```
$ starvault write -format=json identity/group name="education" \
        policies="group-tmpl" \
        metadata=region="us-west" \
        member_entity_ids=$(cat entity_id.txt)  \
        | jq -r ".data.id" > group_id.txt
```

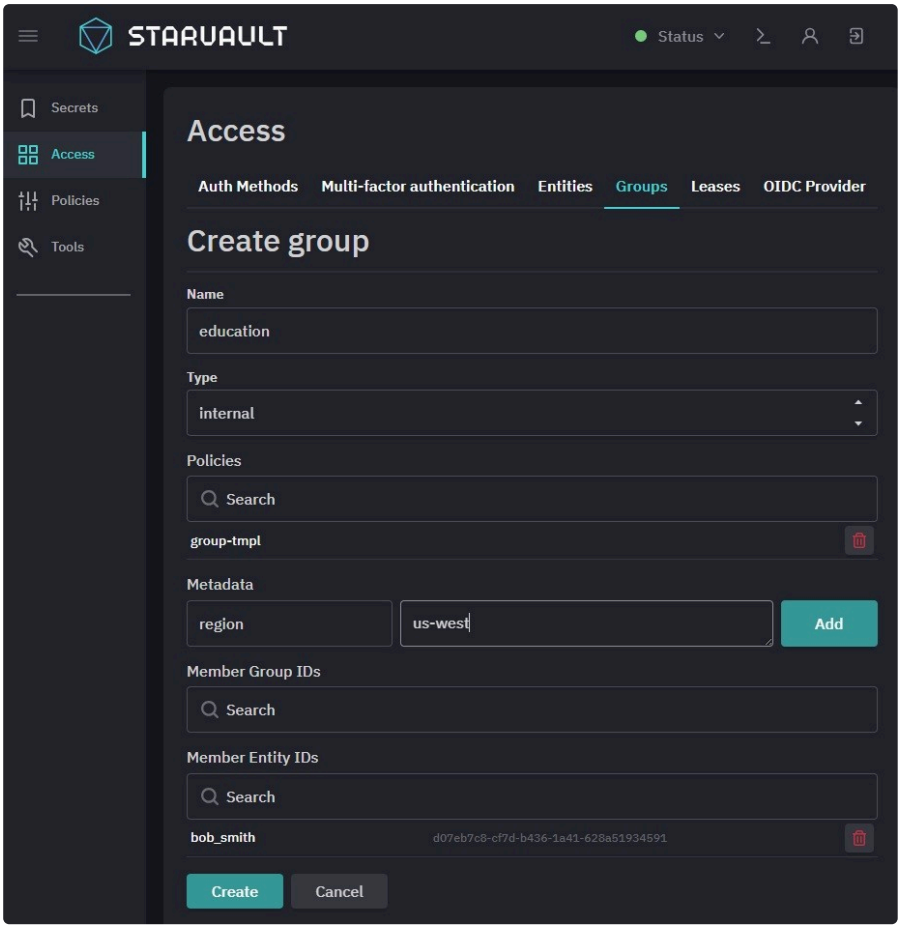
7.2. Веб-интерфейс

1. Перейдите на вкладку **Access** и выберите **Enable new method**.
2. Выберите **Username & Password** в раскрывающемся меню **Type**.
3. Нажмите **Enable Method**.
4. Выберите **< userpass**, чтобы вернуться на страницу авторизации, и нажмите **Create user**.
5. Введите `bob` в поле **Username** и `training` в поле **Password**.



6. Нажмите **Save**.

7. На вкладке **Access** выберите **Entities**, затем **Create entity**.
8. Введите `bob_smith` в поле **Name** и введите `user-tmpl` в поле **Policies**.
9. Нажмите **Create**.
10. Выберите **Create alias**. Введите `bob` в поле `Name` и выберите `userpass/ (userpass)` из раскрывающегося списка **Auth Backend**.
11. Нажмите **Create**.
12. В левой части навигации нажмите **Groups** и выберите **Create group**.
13. Введите `education` в поле **Name** и введите `group-tmpl` в полях **Policies**. В разделе **Metadata** введите `region` в качестве ключа и `us-west` в качестве значения ключа. В поле Member **Entity IDs** введите `bob_smith`.



14. Нажмите **Create**.

8. Тестирование шаблонизации ACL

8.1. CLI команда

1. Включите механизм секретов key/value v2 у `user-kv`.

```
$ starvault secrets enable -path=user-kv kv-v2
```

2. Включите механизм секретов key/value v2 в `group-kv`.

```
$ starvault secrets enable -path=group-kv kv-v2
```

3. Снимите переменную окружения `STARVAULT_TOKEN`, чтобы можно было войти в систему под другим пользователем.

```
$ unset STARVAULT_TOKEN
```

4. Войдите в систему под именем `bob`.

```
$ starvault login -method=userpass username="bob" password="training"
```

Key	Value
token	5f2b2594-f0b4-0a7b-6f51-767345091dcc
token_accessor	78b652dd-4320-f18f-b882-0732b7ae9ac9
token_duration	768h
token_renewable	true
token_policies	["default"]
identity_policies	["group-tmpl" "user-tmpl"]
policies	["default" "group-tmpl" "user-tmpl"]
token_meta_username	bob

5. Помните, что `bob` является членом сущности `bob_smith`; поэтому выражение `"user-kv/data/{{identity.entity.name}}/"` **в политике** `user-tmpl` **переводится как** `"user-kv/data/bob_smith/"`. Проверим.

```
$ starvault kv put user-kv/bob_smith/apikey webapp="12344567890"
```

Key	Value
created_time	2025-05-22T07:16:42.406652868Z
custom_metadata	<nil>

Key	Value
deletion_time	n/a
destroyed	false
version	1

Использование циклов управления, таких как циклы for, для динамического шаблонирования путей может увеличить время отклика.

6. Регион был установлен на `us-west` для группы `education`, к которой принадлежит `bob_smith`. Поэтому выражение `"group-kv/data/education/{{identity.groups.names.education.metadata.region}}/"` **в политике** `group-tmpl` **преобразуется в** `"group-kv/data/education/us-west/"`. Проверим.

```
$ starvault kv put group-kv/education/us-west/db_cred
password="ABCDEFGHIJKLMN"
```

Key	Value
---	-----
created_time	2025-05-22T07:18:16.946740638Z
custom_metadata	<nil>
deletion_time	n/a
destroyed	false
version	1

7. Убедитесь, что вы можете обновлять информацию о группе. Политика `group-tmpl` разрешает "update" и "read" по пути `"identity/group/id/{{identity.groups.names.education.id}}"`. В шаге 2 вы сохранили идентификатор группы `education` в файле `group_id.txt`.

```
$ starvault write identity/group/id/$(cat group_id.txt) \
        policies="group-tmpl" \
        metadata=region="us-west" \
        metadata=contact_email="james@example.com"
```

8. Выведите информацию о группе, чтобы убедиться, что данные были обновлены.

```
$ starvault read identity/group/id/$(cat group_id.txt)
```

Key	Value
alias	map[]
creation_time	2025-05-22T07:10:21.762300968Z
id	d6ee454e-915a-4bef-9e43-4ffd7762cd4c
last_update_time	2025-05-22T07:21:27.162861859Z
member_entity_ids	[1a272450-d147-c3fd-63ae-f16b65b5ee02]
member_group_ids	<nil>
metadata	map[contact_email:james@example.com region:us-west]
modify_index	3
name	education
namespace_id	root
parent_group_ids	<nil>
policies	[group-tmpl]
type	internal

8.2. Вызов API с помощью cURL

8.2.1. StarVault

1. Создайте полезную нагрузку запроса API, содержащую тип двигателя секретов.

```bash
$ tee payload.json <<EOF
{
  "type": "kv",
  "options": {
    "version": "2"
  }
}
EOF
```

2. Включите механизм секретов key/value v2 у `user-kv`.

```bash
$ curl --header "X-Vault-Token: $STARVAULT_TOKEN" \
   --request POST \
```

```
   --data @payload.json \
   $STARVAULT_ADDR/v1/sys/mounts/user-kv
```

3. Включите механизм секретов key/value v2 в `group-kv`.

```
$ curl --header "X-Vault-Token: $STARVAULT_TOKEN" \
   --request POST \
   --data @payload.json \
   $STARVAULT_ADDR/v1/sys/mounts/group-kv
```

4. Войдите в систему под именем `bob` и создайте переменную с `client_token`.

```
$ BOB_TOKEN=$(curl --request POST \
   --data '{"password": "training"}' \
   $STARVAULT_ADDR/v1/auth/userpass/login/bob | jq -r ".auth |
.client_token")
```

5. Помните, что `bob` является членом сущности `bob_smith`; поэтому выражение `"user-kv/data/{{identity.entity.name}}/"` **в политике user-tmpl переводится как** `"user-kv/data/bob_smith/"`. Проверим.

```
$ curl --header "X-Vault-Token: $BOB_TOKEN" \
   --request POST \
   --data '{ "data": {"webapp": "12344567890"} }' \
   $STARVAULT_ADDR/v1/user-kv/data/bob_smith/apikey | jq
```

Пример вывода:

```
{
  "request_id": "86f8eaa6-e651-b015-8a3a-c76ce6f23397",
  "lease_id": "",
  "renewable": false,
  "lease_duration": 0,
  "data": {
    "created_time": "2025-05-22T08:43:30.908523274Z",
    "custom_metadata": null,
    "deletion_time": "",
    "destroyed": false,
    "version": 1
  },
  "wrap_info": null,
  "warnings": null,
  "auth": null
}
```

6. Регион был установлен на `us-west` для группы образования, к которой принадлежит `bob_smith`. Поэтому выражение `"group-kv/data/education/{{identity.groups.names.education.metadata.region}}/"` **в**

политике `group-tmpl` преобразуется в `"group-kv/data/education/us-west/ "`. Проверим.

```
$ curl --header "X-Vault-Token: $BOB_TOKEN" \
   --request POST \
   --data '{ "data": {"password": "ABCDEFGHIJKLMN"} }' \
   $STARVAULT_ADDR/v1/group-kv/data/education/us-west/db_cred | jq
```

Пример вывода:

BASH

```
{
  "request_id": "0b19479c-9bf4-4386-45a9-9bd09b1b9424",
  "lease_id": "",
  "renewable": false,
  "lease_duration": 0,
  "data": {
    "created_time": "2025-05-22T08:44:05.722218515Z",
    "custom_metadata": null,
    "deletion_time": "",
    "destroyed": false,
    "version": 1
  },
  "wrap_info": null,
  "warnings": null,
  "auth": null
}
```

7. Убедитесь, что вы можете обновлять информацию о группе. Политика `group-tmpl` разрешает "update" и "read" по пути `"identity/group/id/{{identity.groups.names.education.id}}"`.

   Сначала создайте полезную нагрузку запроса API, содержащую данные, которые вы хотите записать.

BASH

```
$ tee group_info.json <<EOF
{
  "metadata": {
    "region": "us-west",
    "contact_email": "james@example.com"
  },
  "policies": "group-tmpl"
}
EOF
```

8. Проверьте, что вы можете обновлять информацию о группе.

BASH

```
$ curl --header "X-Vault-Token: $BOB_TOKEN" \
   --request POST \
```

```
   --data @group_info.json \
   $STARVAULT_ADDR/v1/identity/group/id/$(cat group_id.txt)
```

9. Прочитайте информацию о группе, чтобы убедиться, что данные были обновлены.

```bash
$ curl --header "X-Vault-Token: $BOB_TOKEN" \
   $STARVAULT_ADDR/v1/identity/group/id/$(cat group_id.txt) | jq
```

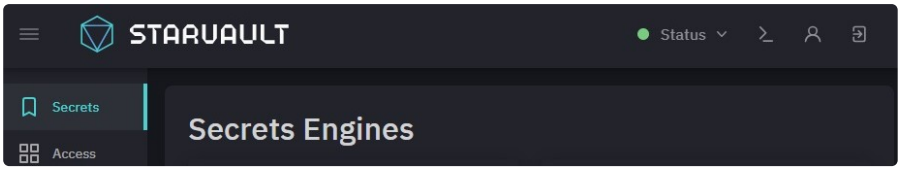
Пример вывода:


```bash
{
  "request_id": "cdbdc0d1-02e9-9e10-97ca-1dddbb66dc7a",
  "lease_id": "",
  "renewable": false,
  "lease_duration": 0,
  "data": {
    "alias": {},
    "creation_time": "2025-05-22T07:10:21.762300968Z",
    "id": "3bdea0a2-d5e9-a4c9-279d-f34f33def9b8",
    "last_update_time": "2025-05-22T08:46:04.891238707Z",
    "member_entity_ids": [
      "d32795fe-08bd-6792-84cb-a63492048ff7"
    ],
    "member_group_ids": null,
    "metadata": {
      "contact_email": "james@example.com",
      "region": "us-west"
    },
    "modify_index": 2,
    "name": "education",
    "namespace_id": "w6LcK",
    "parent_group_ids": null,
    "policies": [
      "group-tmpl"
    ],
    "type": "internal"
  },
  "wrap_info": null,
  "warnings": null,
  "auth": null
}
```


8.3. Веб-интерфейс

1. Во вкладке **Secrets** выберите **Enable new engine.**

2. Нажмите на кнопку **KV** и затем выберите **Next**.
3. Введите `user-kv` в поле path, а затем выберите 2 для версии KV.
4. Нажмите **Enable Engine**.
5. Вернитесь в раздел **Secrets** и снова выберите **Enable new engine**.
6. Нажмите на кнопку **KV** и затем выберите **Next**.
7. Введите `group-kv` в поле path, а затем выберите 2 для версии KV.
8. Нажмите **Enable Engine**.
9. Теперь выйдите из системы, чтобы вы могли войти в систему под именем `bob`.



---

1. На странице входа в StarVault выберите **Username**, затем введите `bob` в поле **Username** и `training` в поле **Password**.
2. Нажмите **Sign in.**
3. Помните, что `bob` является членом сущности `bob_smith`; поэтому выражение `"user-kv/data/{{identity.entity.name}}/"` **в политике** `user-tmpl` **преобразуется в** `"user-kv/data/bob_smith/"`. Выберите механизм секретов `user-kv`, а затем выберите **Create secret**.
4. Введите `bob_smith/apikey` в поле **PATH FOR THIS SECRET**, `webapp` в поле **key** и `12344567890` в поле **value**.
5. Нажмите **Save**. Вы должны успешно выполнить это действие.
6. Регион был установлен на `us-west` для группы `education`, к которой принадлежит `bob_smith`. Поэтому выражение `"group-kv/data/education/{{identity.groups.names.education.metadata.region}}/"` **в политике** `group-tmpl` **преобразуется в** `"group-kv/data/education/us-west/"`. На вкладке **Secrets** выберите механизм секретов `group-kv`, а затем выберите **Create secret**.
7. Введите `education/us-west/db_cred` в поле **PATH FOR THIS SECRET**. Введите `password` в поле **key** и `ABCDEFGHIJKLMN` в поле **value**.
8. Нажмите **Save**. Вы должны успешно выполнить это действие.
9. Чтобы убедиться, что вы можете обновить информацию о группе, которая разрешена выражением `"identity/group/id/{{identity.groups.names.education.id}}"` в политике `group-tmpl`, выберите вкладку **Access**.
10. Выберите **Groups**, а затем `education`.

11. Выберите **Edit group**. Добавьте новые метаданные, где ключ - `contact_email`, а значение - `james@example.com`.
12. Нажмите **Save**. Вы должны успешно выполнить это действие.

Общие сведения об установке с помощью HELM

Для установки и настройки StarVault в Kubernetes рекомендуется использовать Helm-чарт StarVault.

Helm Charts не совместимы с Helm 2. Пожалуйста, используйте Helm 3.6+ для работы с Helm Charts.

Возможности Helm-чартов в StarVault:

- Запуск StarVault.
- Helm-чарт основной метод установки
- Настройки StarVault для интеграции с другими сервисами.
- Развертывания StarVault в режиме высокой доступности (HA).

Для ознакомления с этим разделом необходимы общие знания о Helm и о том, как его использовать. Для установки StarVault с помощью Helm нужно, чтобы Helm был правильно установлен и настроен в кластере Kubernetes.

1. Поддерживаемые версии Kubernetes

В настоящее время поддерживаются следующие промежуточные релизы Kubernetes, перечисленные ниже. Последняя версия StarVault тестируется на совместимость с каждой из нижеперечисленных версий Kubernetes. StarVault может работать с другими версиями Kubernetes, но они не поддерживаются.

- 1.29

2. Использование Helm-чарта

Helm должен быть установлен и настроен.

Для использования Helm-чарта введите логин и пароль для входа в репозиторий и убедитесь, что есть доступ к чарту:

BASH

```
helm registry login -u USER -p PASSWORD https://hub.orionsoft.ru/public
helm show chart oci://hub.orionsoft.ru/public/starvault
```

- `USER` — логин от учетной записи в репозитории.
- `PASSWORD` — пароль от учетной записи в репозитории.

Helm-чарт – это новое решение, которое находится в стадии интенсивной разработки. Поэтому перед установкой или обновлением всегда запускайте Helm командой `--dry-run`, чтобы проверить наличие изменений.

2.1. Пример использования Helm-чарта

Установка последнего релиза Helm-чарта StarVault с подами с префиксом `starvault`.

```bash
helm install starvault oci://hub.orionsoft.ru/public/starvault --namespace starvault --create-namespace
```

По умолчанию чарт работает в автономном режиме, в котором используется один сервер StarVault с бэкендом файлового хранилища. Эта конфигурация не так надежна и безопасна, а потому НЕ подходит для боевой среды. Настоятельно рекомендуется использовать должным образом защищенный кластер Kubernetes.

Так как репозиторий StarVault защищён — используйте опцию `imagePullSecrets` в переменных Helm-чарта или `StatefulSets` (после деплоя).

2.2. Пример создания секрета с учетными данными для подключения к репозиторию

```bash
kubectl create secret docker-registry myregsecret --docker-server=myregistry.example.com --docker-username=myusername --docker-password=mypassword --docker-email=myemail@example.com
```

- `docker-server` — адрес репозитория.
- `docker-username` — имя пользователя от учетной записи в репозитории.
- `docker-password` — пароль пользователя от учетной записи в репозитории.
- `docker-email` — почта пользователя от учетной записи в репозитории.

Примеры конфигураций HELM

Здесь собраны примеры распространенных конфигураций для StarVault с использованием Helm Charts.

Helm Charts не совместимы с Helm 2. Пожалуйста, используйте Helm 3.6+ для работы с Helm Charts.

Ниже приведены примеры конфигураций для поддержки различных моделей развертывания. Примеры можете просмотреть в списке слева.

1. Настройка сервера StarVault для разработки

Приведенный ниже файл `values.yaml` можно использовать для настройки одного сервера разработки StarVault.

YAML

```yaml
server:
  dev:
    enabled: true
```

2. Настройка сервера StarVault с использованием Raft

YAML

```yaml
global:
  tlsDisable: true
server:
  image:
    repository: "hub.orionsoft.ru/public/starvault"
    tag: "1.2.0"
  ingress:
    enabled: true
    ingressClassName: nginx-public
    activeService: true
    hosts:
      - host: starvault-example.local
        paths: []
    extraPaths: []
    tls: []
  dataStorage:
    enabled: true
    size: 10Gi
```

```
    mountPath: "/starvault/data"
    storageClass: "ovirt-csi-sc"
  auditStorage:
    enabled: true
    size: 10Gi
    mountPath: "/starvault/audit"
    storageClass: "ovirt-csi-sc"
    accessMode: ReadWriteOnce
    annotations: {}
    config: |
      ui = true      listener "tcp" {
        tls_disable = 1
        address = "[::]:8200"
        cluster_address = "[::]:8201"
      }
      storage "file" {
        path = "/starvault/data"
      }
  ha:
    enabled: true
    replicas: 1
    apiAddr: null
    clusterAddr: null
    raft:
      enabled: true
      setNodeId: false
      config: |
        ui = true        listener "tcp" {
          tls_disable = 1
          address = "[::]:8200"
          cluster_address = "[::]:8201"
        }        storage "raft" {
          path = "/starvault/data"
        }        service_registration "kubernetes" {}
    config: |
      ui = true      listener "tcp" {
        tls_disable = 1
        address = "[::]:8200"
        cluster_address = "[::]:8201"
      }
      storage "consul" {
        path = "vault"
        address = "HOST_IP:8500"
      }      service_registration "kubernetes" {}
ui:
  enabled: false
  publishNotReadyAddresses: true
  activeVaultPodOnly: false
  serviceType: "ClusterIP"
  serviceNodePort: null
```

```
  externalPort: 8200
  targetPort: 8200
```