Бенчмарк производительности StarVault

Эффективная работа StarVault для поддержки ваших сценариев использования требует, чтобы вы могли точно измерить его производительность. В идеале для получения реалистичных результатов необходимо проводить сравнительный анализ и измерять производительность в средах, которые напоминают производственные сценарии использования.

1. Задача

Вам нужно измерить производительность StarVault значимым способом и в среде, которая также похожа на ту, что используется в вашем предполагаемом сценарии использования в отношении вычислительных ресурсов.

На тестируемом сервере StarVault должны быть включены те же методы аутентификации и механизмы секретов, а также представлены примеры секретов, аренд и токенов для точного моделирования ваших сценариев использования.

2. Решение

OrionSoft предоставляет утилиту с открытым исходным кодом vault-benchmark, чтобы помочь вам измерить производительность StarVault на гранулированном уровне, используя несколько доступных методов авторизации и механизмов секретов.

Вы можете использовать vault-benchmark как интерфейс командной строки, как образ Docker или как рабочую нагрузку Kubernetes, чтобы соответствовать инфраструктуре, которую вы используете для StarVault.

3. Пользователи

В сквозном сценарии и практической лаборатории, описанных в этом руководстве, задействована один пользователь.

Этот пользователь - оператор хранилища с привилегированными правами на включение и отключение методов аутентификации и секретных механизмов. Все задачи в практическом сценарии выполняются от имени этого пользователя.

4. Предпосылки

Для выполнения практического сценария вам понадобятся следующие ресурсы в зависимости от того, будете ли вы использовать версию StarVault с CLI, Docker или Kubernetes.

4.1. Self-hosted StarVault

Для выполнения практического задания с использованием CLI-версий StarVault и vault-benchmark вам потребуется следующее:

- Бинарный файл StarVault, установленный и находящийся в системном PATH.
- Бинарный файл vault-benchmark, установленный и находящийся в системном PATH.

4.2. Docker

Для выполнения практического задания с использованием Docker-версий StarVault и vault-benchmark вам потребуется следующее:

- Установленный образ StarVault Docker.
- Установленный Docker-образ vault-benchmark.

4.3. Kubernetes

Для выполнения практического задания с использованием Kubernetes вам потребуется следующее:

- Minikube
- StarVault Helm chart

5. Версии, используемые в руководстве

Это руководство было последний раз проверено 21 мая 2025 года на AlmaLinux с использованием следующих версий программного обеспечения.

ПО	Версия

Operating System	BASH
	`$ cat /etc/redhat-release`
	`AlmaLinux release 8.10 (Cerulean Leopard)`

ПО	Версия

StarVault	BASH
	`$ starvault version`
	`StarVault v1.2.0 ('1a46c26ee4e8fd00b4111300a865499c0e522e67+CHANGES'), built 2025-03-14T20:50:17Z`

vault-benchmark	BASH
	`$ vault-benchmark version`
	`vault-benchmark v0.3.0`

Docker server	BASH
	`$ docker version --format '{{.Server.Version}}'`
	`26.1.3`

Curl	BASH
	`$ curl --version | head -n 1 | awk '{print $2}'`
	`7.61.1`

jq	BASH
	`$ jq --version`
	`jq-1.6`

helm	BASH
	`$ helm version --short`
	`v3.17.3+ge4da497`

Minikube	BASH
	`$ minikube version`
	`minikube version: v1.35.0`
	`commit: dd5d320e41b5451cdf3c01891bc4e13d189586ed-dirty`

6. Настройка лаборатории

Настройка практической лаборатории в этом руководстве отличается в зависимости от того, хотите ли вы экспериментировать с vault-benchmark в CLI с сервером dev mode, в CLI с сервером StarVault Dedicated, в Docker или в Kubernetes.

6.1. Создание локальной практической лаборатории

Вы можете создать временный каталог, в котором будет храниться все содержимое, необходимое для этой практической работы, а затем присвоить его путь переменной среды для последующего использования.

1. Откройте терминал и создайте каталог `/tmp/learn-vault-pgp`.

```
$ mkdir /tmp/learn-vault-benchmark
```

2. Экспортируйте путь к каталогу практической лаборатории в качестве значения переменной среды `HC_LEARN_LAB`.

```
$ export HC_LEARN_LAB=/tmp/learn-vault-benchmark
```

Теперь выберите рабочий процесс настройки лаборатории, который соответствует среде, которую вы хотите использовать для этой практической работы.

6.1.1. Self-hosted StarVault

Запустите сервер StarVault dev mode в качестве фонового процесса из терминальной сессии, чтобы проследить за рабочим процессом самостоятельной установки StarVault в этой практике.

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

```
$ starvault server \
    -dev \
    -dev-root-token-id root \
    > "$HC_LEARN_LAB"/starvault-server.log 2>&1 &
```

   По умолчанию сервер StarVault dev работает по адресу `127.0.0.1:8200`. Сервер регистрируется в файле `starvault-server.log` в рабочем каталоге практической лаборатории, автоматически инициализируется и снимает печать.

> **Режим Dev не предназначен для производства**
>
> Не запускайте сервер StarVault dev в производстве. При таком подходе сервер StarVault запускается с базой данных in-memory, и все содержимое будет потеряно, когда процесс сервера StarVault будет остановлен.

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

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

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

```
$ export VAULT_TOKEN=root
```

4. Проверьте состояние StarVault.

```
$ starvault status
```

BASH

Пример вывода:

Key	Value
Seal Type	shamir
Initialized	true
Sealed	false
Total Shares	1
Threshold	1
Version	1.2.0
Build Date	2025-03-14T20:50:17Z
Storage Type	inmem
Cluster Name	vault-cluster-becc609e
Cluster ID	96babf60-d3e2-0ea0-4879-f78763ac595c
HA Enabled	false

5. Сервер StarVault готов к продолжению практической работы.

6.1.2. Docker

Запустите Docker-контейнер StarVault dev mode из терминала, чтобы проследить за рабочим процессом Docker в этой практической работе.

Создайте сеть Docker с именем learn-vault, которую будут использовать все контейнеры в этой практике.

---

1. Создайте сеть.

```
$ docker network create --attachable --subnet 172.30.0.0/24 learn-starvault
```

BASH

2. Запустите контейнер StarVault dev mode.

```bash
$ docker run \
   --ip 172.30.0.2 \
   --name learn-starvault \
   --network learn-starvault \
   --cap-add=IPC_LOCK \
   --env 'VAULT_DEV_ROOT_TOKEN_ID=root' \
   --env 'VAULT_DEV_LISTEN_ADDRESS=0.0.0.0:8200' \
   --publish 8200:8200 \
   --detach \
   --rm \
   hub.orionsoft.ru/public/starvault:v1.2.0 server -dev
```

Если вы никогда раньше не использовали образ StarVault, Docker создаст его, а затем запустит контейнер с этим образом.

Режим Dev не предназначен для производства

Не запускайте сервер StarVault dev в производстве. При таком подходе сервер StarVault запускается с базой данных in-memory, и все содержимое будет потеряно, когда процесс сервера StarVault будет остановлен.

3. Проверьте состояние StarVault.

```bash
$ docker run \
    --cap-add IPC_LOCK \
    --ip 172.30.0.3 \
    --name learn-starvault-client \
    --network learn-starvault \
    --env 'STARVAULT_ADDR=http://172.30.0.2:8200' \
    --rm \
    hub.orionsoft.ru/public/starvault:v1.2.0 status
```

Пример вывода:

Key	Value
`Seal Type`	`shamir`
`Initialized`	`true`
`Sealed`	`false`
`Total Shares`	`1`
`Threshold`	`1`
`Version`	`1.2.0`

Key	Value
`Build Date`	`2025-03-14T20:50:17Z`
`Storage Type`	`inmem`
`Cluster Name`	`vault-cluster-bcfbdbc6`
`Cluster ID`	`a38f6679-03fd-af38-5d7b-a2baad820c35`
`HA Enabled`	`false`

Docker-контейнер сервера StarVault dev mode готов для продолжения практической работы.

6.1.3. Kubernetes

Разверните StarVault в Minikube с помощью диаграммы Helm из терминальной сессии, чтобы проследить за рабочим процессом Kubernetes в этой практической работе.

Погрузитесь глубже

Если вы не знакомы с этим процессом или хотите углубиться в него, просмотрите руководство по установке StarVault на minikube через Helm с интегрированным хранилищем.

1. Запустите Minikube.

```bash
$ minikube start
```

Пример вывода:

```
* minikube v1.35.0 on Almalinux 8.10 (kvm/amd64)
* Automatically selected the kvm2 driver
* Downloading driver docker-machine-driver-kvm2:
    > docker-machine-driver-kvm2-...:  65 B / 65 B [---------] 100.00% ? p/s
0s
    > docker-machine-driver-kvm2-...:  14.14 MiB / 14.14 MiB  100.00% 53.03
MiB
* Downloading VM boot image ...
    > minikube-v1.35.0-amd64.iso....:  65 B / 65 B [---------] 100.00% ? p/s
0s
    > minikube-v1.35.0-amd64.iso:  345.38 MiB / 345.38 MiB  100.00% 33.43
MiB p
* Starting "minikube" primary control-plane node in "minikube" cluster
* Downloading Kubernetes v1.32.0 preload ...
    > preloaded-images-k8s-v18-v1...:  333.57 MiB / 333.57 MiB  100.00%
32.87 M
```

```
* Creating kvm2 VM (CPUs=2, Memory=2200MB, Disk=20000MB) ...
* Preparing Kubernetes v1.32.0 on Docker 27.4.0 ...
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default"
namespace by default
```

2. Проверьте состояние Minikube.

```bash
$ minikube status
```

Пример вывода:

```
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

3. Залогиньтесь в репозиторий StarVault Helm.

```bash
$ helm registry login -u USER -p PASSWORD https://hub.orionsoft.ru/public
```

4. Установите диаграмму Helm сервера StarVault, чтобы запустить сервер в режиме dev.

```bash
$ helm install starvault oci://hub.orionsoft.ru/public/starvault \
    --set "server.dev.enabled=true" \
    --set "server.dev.devRootToken=root"
```

Пример вывода:

```
NAME: starvault
LAST DEPLOYED: Tue Oct 24 15:11:26 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
Thank you for installing StarVault.
...snip...
```

Режим Dev не для производства

Не запускайте сервер StarVault dev в производстве. При таком подходе сервер StarVault запускается с базой данных in-memory, и все содержимое теряется, когда процесс сервера StarVault останавливается.

5. Отобразите все поды в пространстве имен по умолчанию.

```
$ kubectl get pods
```

BASH

Пример вывода:

NAME	READY	STATUS	RESTARTS	AGE
`starvault-0`	`0/1`	`Running`	`0`	`67s`

Сервер StarVault dev mode запущен в поде с именем `starvault-0`. При необходимости проверьте статус и убедитесь, что под имеет статус **Running**, прежде чем продолжить.

6. Проверьте статус сервера StarVault

```
$ kubectl exec starvault-0 -- starvault status
```

BASH

Пример вывода:

Key	Value
---	----
`Seal Type`	`shamir`
`Initialized`	`true`
`Sealed`	`false`
`Total Shares`	`1`
`Threshold`	`1`
`Version`	`1.2.0`
`Build Date`	`2025-03-14T20:50:17Z`
`Storage Type`	`inmem`
`Cluster Name`	`vault-cluster-f0037191`
`Cluster ID`	`1f788714-7dcc-39f9-65a4-adcf8dc90c15`
`HA Enabled`	`false`

Docker-контейнер сервера StarVault dev mode готов для продолжения практической работы.

7. Изучение vault-benchmark

Цель в этом разделе - изучить vault-benchmark с помощью терминала. В конце раздела приведены ресурсы для более глубокого погружения.

7.1. Self-hosted-Vault

1. Проверьте версию вашего vault-benchmark.

```bash
$ vault-benchmark version
```

Пример вывода:

```bash
vault-benchmark v0.3.0
```

2. Вы можете получить помощь, чтобы узнать о доступных командах.

```bash
$ vault-benchmark --help
```

Пример вывода:

```bash
Usage: vault-benchmark <command> [args]

Command list:
   run         Run vault-benchmark test(s)
   review      Review previous test results
```

3. Эта практическая лабораторная работа посвящена использованию команды `run`. Получите справку по команде `run`.

```bash
$ vault-benchmark run --help
```

Пример вывода:

```bash
Usage: vault-benchmark run [options]

This command will run a vault-benchmark test.

Run a vault-benchmark test with a configuration file:

   $ vault-benchmark run -config=/etc/vault-benchmark/test.hcl
```

```
For a full list of examples, please see the documentation.

Command Options:

-annotate=<string>
      Comma-separated name=value pairs include in bench_running prometheus
      metric. Try name 'testname' for dashboard example.

-audit_path=<string>
      Path to file for audit log.

-ca_pem_file=<string>
      Path to PEM encoded CA file to verify external Vault. This can also be
      specified via the VAULT_CACERT environment variable.

-cleanup
      Cleanup benchmark artifacts after run. The default is false.

-cluster_json=<string>
      Path to cluster.json file

-config=<string>
      Path to a vault-benchmark test configuration file.

-debug
      Run vault-benchmark in Debug mode. The default is false.

-disable_http2
      Force HTTP/1.1 The default is false.

-duration=<duration>
      Test Duration. The default is 10s.

-log_level=<string>
      Level to emit logs. Options are: INFO, WARN, DEBUG, TRACE. The default
      is INFO. This can also be specified via the VAULT_BENCHMARK_LOG_LEVEL
      environment variable.

-pprof_interval=<duration>
      Collection interval for vault debug pprof profiling.

-random_mounts
      Use random mount names. The default is true.

-report_mode=<string>
      Reporting Mode. Options are: terse, verbose, json. The default is
terse.

-rps=<int>
      Requests per second. Setting to 0 means as fast as possible.
```

```
-vault_addr=<string>
      Target Vault API Address. The default is http://127.0.0.1:8200. This
can
      also be specified via the VAULT_ADDR environment variable.

-vault_namespace=<string>
      Vault Namespace to create test mounts. This can also be specified via
      the VAULT_NAMESPACE environment variable.

-vault_token=<string>
      Vault Token to be used for test setup. This can also be specified via
      the VAULT_TOKEN environment variable.

-workers=<int>
      Number of workers The default is 10.
```

7.2. Docker

1. Извлеките Docker-образ vault-benchmark.

```
BASH
$ docker pull hashicorp/vault-benchmark
```

Пример вывода:

```
BASH
Using default tag: latest
latest: Pulling from hashicorp/vault-benchmark
4abcf2066143: Already exists
5db785bb5ae1: Pull complete
77cab26cbc7c: Pull complete
900c60e306e5: Pull complete
Digest:
sha256:3f0ebc573b2615db5070bcfa724bd952c4b9f1679f17f7bf3ad3e678bb36a434
Status: Downloaded newer image for hashicorp/vault-benchmark:latest
docker.io/hashicorp/vault-benchmark:latest
```

2. Проверьте версию вашего vault-benchmark.

```
BASH
$ docker run hashicorp/vault-benchmark vault-benchmark version
```

Пример вывода:

```
BASH
vault-benchmark v0.2.0
```

3. Вы можете получить помощь, чтобы узнать о доступных командах.

```bash
$ docker run hashicorp/vault-benchmark vault-benchmark --help
```

Пример вывода:

```
Usage: vault-benchmark <command> [args]

Command list:
   run         Run vault-benchmark test(s)
   review      Review previous test results
```

4. Эта практическая лабораторная работа посвящена использованию команды `run`. Получите справку по команде `run`.

```bash
$ docker run hashicorp/vault-benchmark vault-benchmark run --help
```

Пример вывода:

```
Usage: vault-benchmark run [options]

This command will run a vault-benchmark test.

Run a vault-benchmark test with a configuration file:

   $ vault-benchmark run -config=/etc/vault-benchmark/test.hcl

For a full list of examples, please see the documentation.

Command Options:

-annotate=<string>
      Comma-separated name=value pairs include in bench_running prometheus
      metric. Try name 'testname' for dashboard example.

-audit_path=<string>
      Path to file for audit log.

-ca_pem_file=<string>
      Path to PEM encoded CA file to verify external Vault. This can also be
      specified via the VAULT_CACERT environment variable.

-cleanup
      Cleanup benchmark artifacts after run. The default is false.

-cluster_json=<string>
      Path to cluster.json file

-config=<string>
      Path to a vault-benchmark test configuration file.
```

```
-debug
      Run vault-benchmark in Debug mode. The default is false.

-disable_http2
      Force HTTP/1.1 The default is false.

-duration=<duration>
      Test Duration. The default is 10s.

-log_level=<string>
      Level to emit logs. Options are: INFO, WARN, DEBUG, TRACE. The default
      is INFO. This can also be specified via the VAULT_BENCHMARK_LOG_LEVEL
      environment variable.

-pprof_interval=<duration>
      Collection interval for vault debug pprof profiling.

-random_mounts
      Use random mount names. The default is true.

-report_mode=<string>
      Reporting Mode. Options are: terse, verbose, json. The default is
terse.

-rps=<int>
      Requests per second. Setting to 0 means as fast as possible.

-vault_addr=<string>
      Target Vault API Address. The default is http://127.0.0.1:8200. This
can
      also be specified via the VAULT_ADDR environment variable.

-vault_namespace=<string>
      Vault Namespace to create test mounts. This can also be specified via
      the VAULT_NAMESPACE environment variable.

-vault_token=<string>
      Vault Token to be used for test setup. This can also be specified via
      the VAULT_TOKEN environment variable.

-workers=<int>
      Number of workers The default is 10.
```

7.3. Kubernetes

Вы можете изучить документацию по vault-benchmark, чтобы узнать больше о его настройке.

В документации по Доступным тестам описаны все доступные тесты и их конфигурация. Вы можете использовать эту информацию для настройки задания vault-benchmark Kubernetes.

Вы будете использовать пример конфигурации бенчмарка, включающий метод Kubernetes auth; в Документации по бенчмарку Kubernetes auth описаны все доступные параметры, предостережения и пример использования этого бенчмарка.

Все эталонные тесты документируются аналогичным образом.

8. Настройка бенчмарка

Ваша цель в этом разделе - настроить vault-benchmark для выполнения базового бенчмарка.

Vault Benchmark настраивается с помощью файла HCL. В этом практическом занятии вы можете использовать пример конфигурации, показанный в документации Usage.

Далее пример конфигурационного файла.

vault-benchmark-config.hcl

BASH

```
vault_addr = "http://127.0.0.1:8200"
vault_token = "root"
vault_namespace="root"
duration = "30s"
cleanup = true

test "approle_auth" "approle_logins" {
  weight = 50
  config {
    role {
      role_name = "benchmark-role"
      token_ttl="2m"
    }
  }
}

test "kvv2_write" "static_secret_writes" {
  weight = 50
  config {
    numkvs = 100
    kvsize = 100
  }
}
```

Строки 1-5 - это глобальные параметры:

- `vault_addr`: полный URL-адрес и порт сервера StarVault для проверки.

- `vault_token` : буквальное значение токена для токена с возможностями включения и управления механизмами секретов и методами аутентификации. В этой практической работе используется начальное значение `root` токена, но вы не должны использовать `root` токен в производстве таким образом.
- `vault_namespace` : имя пространства имен Enterprise, которое будет использоваться для эталона.
- `duration` : количество секунд для выполнения эталона.
- `cleanup` : удалять ли все ресурсы, созданные во время выполнения эталона.

> **Не рекомендуется использовать бенчмарк в производстве**
>
> Бенчмаркинг следует проводить на серверах StarVault, предназначенных для этой цели, но не на рабочих серверах. Бенчмарк может негативно повлиять на производительность сервера и не дает гарантий по очистке артефактов, созданных во время выполнения бенчмарка.

Строки 7-15 и 17-23 представляют собой два теста, составляющих эту конфигурацию бенчмарка. В этой тестовой конфигурации выполняются два разных теста, тест **approle_auth** и тест **kvv2_write**, причем процент запросов распределяется поровну между ними.

Первый тест, начинающийся в строке 7, предназначен для входа в систему методом AppRole auth. Обратите внимание, что первым параметром является **weight**. Его можно представить как процент от всей рабочей нагрузки. Это означает, что бенчмарк выполняет логины AppRole 50 % времени в течение всего прогона.

Вы можете настроить тест в его строфе **config**. В этом тесте **role_name** указывает имя роли **benchmark-role** для метода аутентификации. Каждый токен, который StarVault выдает после входа в AppRole с ролью **benchmark-role**, имеет значение **token_ttl**, равное 2 минутам, чтобы указать время жизни токена (TTL). Подробнее о доступных параметрах можно узнать в документации по эталонному методу AppRole auth.

Второй тест, начинающийся со строки 17, предназначен для записи секретов движка Key/Value версии 2. Этот тест использует оставшиеся 50 % операций бенчмарка с настройкой **weight**. Он также использует numkvs для указания записи 100 секретов ключей/значений и kvsize для ограничения каждого значения 100 байтами. Доступные параметры описаны в документации по бенчмаркам секретов KV v1 и KV v2

Подробнее обо всех тестах вы можете узнать в документации по тестам vault-benchmark.

Теперь, когда вы ознакомились с примером конфигурации, вам нужно немного изменить ее для работы с кластером StarVault, который вы используете.

8.1. Self-hosted StarVault и Docker

1. Вернитесь в терминал и запишите файл конфигурации в `/tmp/learn-vault-benchmark/vault-benchmark-config.hcl`.

```
$ cat > "$HC_LEARN_LAB"/vault-benchmark-config.hcl << EOF
vault_addr = "http://127.0.0.1:8200"
vault_token = "root"
vault_namespace="root"
duration = "30s"
cleanup = true

test "approle_auth" "approle_logins" {
weight = 50
config {
   role {
      role_name = "benchmark-role"
      token_ttl="2m"
   }
}
}

test "kvv2_write" "static_secret_writes" {
weight = 50
config {
   numkvs = 100
   kvsize = 100
}
}
EOF
```

Вы настроили vault-benchmark для теста, и теперь он готов к запуску.

8.2. Docker

Значения переменных окружения `VAULT_ADDR` и `VAULT_TOKEN` переопределяют все, что указано в файле конфигурации vault-benchmark.

Учитывая это, вы можете установить значения vault_addr и vault_token в пустую строку. Их значения будут получены из заданных вами переменных окружения

1. Вернитесь в терминал и запишите файл конфигурации в `/tmp/learn-vault-benchmark/vault-benchmark-config.hcl`.

```
$ cat > "$HC_LEARN_LAB"/vault-benchmark-config.hcl << EOF
vault_addr = ""
vault_token = ""
vault_namespace=""
duration = "30s"
```

```
cleanup = true

test "approle_auth" "approle_logins" {
weight = 50
config {
   role {
      role_name = "benchmark-role"
      token_ttl="2m"
   }
}
}

test "kvv2_write" "static_secret_writes" {
weight = 50
config {
   numkvs = 100
   kvsize = 100
}
}
EOF
```

Вы настроили vault-benchmark для теста, и теперь он готов к запуску.

8.3. Kubernetes

Вы можете настроить vault-benchmark для Kubernetes в виде файла задания. Вот полный пример из репозитория vault-benchmark.

BASH

```
# Copyright (c) HashiCorp, Inc.
# SPDX-License-Identifier: MPL-2.0

---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: vault-benchmark

---
apiVersion: v1
kind: ConfigMap
metadata:
  name: vault-benchmark-configmap
data:
  k8s.hcl: |
    # Basic Benchmark config options
    vault_addr = "http://starvault:8200"
    vault_token = "root"
    duration = "10s"
```

```
    report_mode = "terse"
    random_mounts = true
    cleanup = true

    test "kube_auth" "kube_auth_test1" {
      weight = 100
      config {
        auth {
          kubernetes_host = "https://kubernetes.default.svc"
        }
        role {
          name = "vault-benchmark-role"
          bound_service_account_names = ["vault-benchmark"]
          bound_service_account_namespaces = ["*"]
          token_max_ttl = "24h"
          token_ttl = "1h"
        }
      }
    }

---
apiVersion: batch/v1
kind: Job
metadata:
  name: vault-benchmark
spec:
  backoffLimit: 0
  template:
    metadata:
      name: vault-benchmark
      labels:
        app: vault-benchmark
    spec:
      containers:
      - name: vault-benchmark
        image: hashicorp/vault-benchmark:latest
        imagePullPolicy: IfNotPresent
        command: ["vault-benchmark"]
        args: [
          "run",
          "-config=/config/k8s.hcl",
        ]
        volumeMounts:
        - name: benchmark-config
          mountPath: "/config"
          readOnly: true
      restartPolicy: Never
      serviceAccountName: vault-benchmark
      volumes:
      - name: benchmark-config
```

```
        configMap:
          name: vault-benchmark-configmap
```

В этом бенчмарке есть один тест под названием **kube_auth_test1**, который выполняет вход в систему методом Kubernetes auth.

Вы будете работать с отредактированной версией, в которой изменены следующие настройки.

- Параметр `duration` бенчмарка увеличен до 30 с.
- Добавлен второй тест секретной записи KV v2.
- `weight` каждого теста установлен на 50.

Создайте файл задания `/tmp/learn-vault-benchmark/vault-benchmark-job.yaml`, вставьте в него следующую обновленную конфигурацию и сохраните его.

BASH

```
# Copyright (c) HashiCorp, Inc.
# SPDX-License-Identifier: MPL-2.0

---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: vault-benchmark

---
apiVersion: v1
kind: ConfigMap
metadata:
  name: vault-benchmark-configmap
data:
  k8s.hcl: |
    # Basic Benchmark config options
    vault_addr = "http://starvault:8200"
    vault_token = "root"
    duration = "30s"
    report_mode = "terse"
    random_mounts = true
    cleanup = true

    test "kube_auth" "kube_auth_test1" {
      weight = 50
      config {
        auth {
          kubernetes_host = "https://kubernetes.default.svc"
        }
        role {
          name = "vault-benchmark-role"
```

```
          bound_service_account_names = ["vault-benchmark"]
          bound_service_account_namespaces = ["*"]
          token_max_ttl = "24h"
          token_ttl = "1h"
        }
      }
    }
    test "kvv2_write" "static_secret_writes" {
      weight = 50
      config {
        numkvs = 100
        kvsize = 100
      }
    }

---
apiVersion: batch/v1
kind: Job
metadata:
  name: vault-benchmark
spec:
  backoffLimit: 0
  template:
    metadata:
      name: vault-benchmark
      labels:
        app: vault-benchmark
    spec:
      containers:
      - name: vault-benchmark
        image: hashicorp/vault-benchmark:latest
        imagePullPolicy: IfNotPresent
        command: ["vault-benchmark"]
        args: [
          "run",
          "-config=/config/k8s.hcl",
        ]
        volumeMounts:
        - name: benchmark-config
          mountPath: "/config"
          readOnly: true
      restartPolicy: Never
      serviceAccountName: vault-benchmark
      volumes:
      - name: benchmark-config
        configMap:
          name: vault-benchmark-configmap
```

Теперь вы готовы к запуску бенчмарка.

9. Запуск бенчмарка

Ваша цель в этом разделе - запустить vault-benchmark с конфигурацией, которую вы только что создали.

9.1. Self-hosted StarVault

Запустите бенчмарк.

```bash
$ vault-benchmark run \
    -config="$HC_LEARN_LAB"/vault-benchmark-config.hcl
```

Пример вывода:

```bash
2025-05-21T03:54:47.181-0400 [INFO]  vault-benchmark: setting up targets
2025-05-21T03:54:49.245-0400 [INFO]  vault-benchmark: starting benchmarks:
duration=30s
2025-05-21T03:55:19.246-0400 [INFO]  vault-benchmark: cleaning up targets
2025-05-21T03:55:32.858-0400 [INFO]  vault-benchmark: benchmark complete
Target: http://127.0.0.1:8200
op                    count  rate         throughput   mean        95th%
99th%       successRatio
approle_logins        66525  2217.794922  2217.457572  2.330936ms  7.661117ms
13.64502ms  100.00%
static_secret_writes  66698  2223.240651  2223.186220  2.111459ms  6.553077ms
13.01611ms  100.00%
```

9.2. Docker

Запустите бенчмарк.

```bash
$ docker run \
    --volume="$HC_LEARN_LAB":/vault/config \
    --ip 172.30.0.4 \
    --name learn-vault-benchmark \
    --network learn-starvault \
    --env 'VAULT_TOKEN=root' \
    --env 'VAULT_ADDR=http://172.30.0.2:8200' \
    --rm \
    hashicorp/vault-benchmark vault-benchmark run -config=/vault/config/vault-
benchmark-config.hcl
```

Пример вывода:

```bash
2025-05-21T08:29:16.383Z [INFO]  vault-benchmark: setting up targets
2025-05-21T08:29:18.454Z [INFO]  vault-benchmark: starting benchmarks:
```

```
duration=30s
2025-05-21T08:29:48.455Z [INFO]  vault-benchmark: cleaning up targets
2025-05-21T08:30:02.866Z [INFO]  vault-benchmark: benchmark complete
Target: http://172.30.0.2:8200
op                    count  rate         throughput   mean        95th%
99th%        successRatio
approle_logins        68315  2277.136590  2277.074562  2.281326ms  6.829855ms
13.548974ms  100.00%
static_secret_writes  68282  2276.113953  2276.001160  2.057878ms  5.895404ms
12.87487ms   100.00%
```

9.3. Kubernetes

1. Запустите бенчмарк

```
BASH
$ kubectl apply -f "$HC_LEARN_LAB"/vault-benchmark-job.yaml
```

2. Выведите поды.

```
BASH
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
starvault-0	1/1	Running	0	12m
vault-benchmark-ghjlq	1/1	Running	0	7m

Бенчмарк выполняется в поде vault-benchmark-ghjlq.

3. Изучите результаты, просмотрев логи.

```
BASH
$ kubectl logs --follow vault-benchmark-ghjlq
```

```
BASH
2025-05-21T10:33:14.986Z [INFO]  vault-benchmark: setting up targets
2025-05-21T10:33:17.352Z [INFO]  vault-benchmark: starting benchmarks:
duration=30s
2025-05-21T10:34:42.187Z [INFO]  vault-benchmark: cleaning up targets
2025-05-21T10:34:46.496Z [INFO]  vault-benchmark: benchmark complete
Target: http://starvault:8200
op                    count  rate       throughput  mean          95th%
99th%           successRatio
kube_auth_test1       514    14.511228  5.991322    1.170117445s
1.100958326s  1m5.082967137s  98.64%
```

```
static_secret_writes  529    26.371231  6.215953    456.881766ms
266.507158ms  3.1722234s      99.43%
```

10. Просмотр вывода

По умолчанию эталонный результат выводится в табличной форме. Этот пример относится к самостоятельному серверу dev mode, но вывод будет аналогичным для StarVault в любой среде.

BASH

```
2025-05-21T03:54:47.181-0400 [INFO]  vault-benchmark: setting up targets
2025-05-21T03:54:49.245-0400 [INFO]  vault-benchmark: starting benchmarks:
duration=30s
2025-05-21T03:55:19.246-0400 [INFO]  vault-benchmark: cleaning up targets
2025-05-21T03:55:32.858-0400 [INFO]  vault-benchmark: benchmark complete
Target: http://127.0.0.1:8200
op                    count  rate         throughput   mean        95th%
99th%       successRatio
approle_logins        66525  2217.794922  2217.457572  2.330936ms  7.661117ms
13.64502ms  100.00%
static_secret_writes  66698  2223.240651  2223.186220  2.111459ms  6.553077ms
13.01611ms  100.00%
```

Первые 4 строки - это лог-выводы от vault-benchmark, описывающие его текущие действия.

В 5-й строке показан целевой URL-адрес эталона, а в 6-й - заголовки для метрических данных.

Для последних двух строк, представляющих каждый из двух настроенных эталонных тестов, метрики данных выглядят следующим образом.

- **op**: название теста.
- **count**: количество тестов, выполненных за время работы бенчмарка.
- **rate**: истинная скорость выполнения операций в секунду для всех тестов данного типа.
- **throughput**: количество операций в секунду во всех успешных тестах данного типа.
- **mean**: среднее время в миллисекундах на одну операцию.
- **95th%**: время 95-го процентиля в миллисекундах на одну операцию.
- **99th%**: время 99-го процентиля в миллисекундах на одну операцию.
- **successRatio**: процент успешных тестов данного типа. Если тесты не увенчались успехом, следует обратиться к оперативным журналам StarVault и журналам аудита для получения подробной информации о неудачных тестах.

> **Tip**
>
> Вы также можете вывести результаты бенчмарка в формате JSON с помощью флага `-report_mode=json`.

11. Очистка

Выполните действия по очистке среды, которую вы использовали в практической лаборатории.

11.1. Self-hosted StarVault

1. Остановите сервер StarVault dev mode.

```
$ pkill starvault
```

2. Удалите каталог лабораторных работ.

```
$ rm -rf "$HC_LEARN_LAB"
```

3. Снимите настройки переменных окружения.

```
$ unset STARVAULT_ADDR STARVAULT_TOKEN
```

4. Удалите кэшированный токен StarVault.

```
$ rm -f ~/.vault-token
```

11.2. Docker

1. Остановите контейнер StarVault dev mode; Docker автоматически удалит его.

```
$ docker stop learn-starvault
```

2. Удалите сеть Docker.

```
$ docker network rm learn-starvault
```

3. Удалите каталог лабораторных работ.

```
$ rm -rf "$HC_LEARN_LAB"
```

4. Снимите настройки переменных окружения.

```
$ unset STARVAULT_ADDR STARVAULT_TOKEN
```

11.3. Kubernetes

1. Удалите файл задания бенчмарка.

```
$ kubectl delete jobs/vault-benchmark
```

Пример вывода:

```
job.batch "vault-benchmark" deleted
```

2. Удалите диаграмму Helm сервера StarVault, чтобы остановить и удалить поды StarVault.

```
$ helm uninstall starvault
```

Пример вывода:

```
release "starvault" uninstalled
```

3. Удалите экземпляр Minikube, чтобы остановить его.

```
$ minikube delete
* Deleting "minikube" in kvm2 ...
* Removed all traces of the "minikube" cluster.
```

4. Удалите каталог лабораторных работ.

```
$ rm -rf "$HC_LEARN_LAB"
```

5. Снимите настройки переменных окружения.

```
$ unset STARVAULT_ADDR STARVAULT_TOKEN
```

12. Следующие шаги

Вы узнали об основах работы с инструментом Vault Benchmark, включая настройку и запуск эталона. Вы также узнали о выводах эталона по умолчанию и доступных ресурсах документации по Vault Benchmark.

Чтобы глубже изучить производительность StarVault, ознакомьтесь с документацией по настройке производительности и укреплению производства.

Диагностика проблем с сервером

При работе с StarVault вы можете столкнуться с проблемами при запуске сервера, вызванными целым рядом причин, начиная от проблем с конфигурацией сервера и заканчивая ограничениями операционной среды.

1. Задача

Чтобы эффективно устранить неполадки и решить проблемы с StarVault, необходимо изучить и объединить информацию из трех разных источников, чтобы найти первопричины:

- Условия среды операционной системы, например, ограничения на количество пользователей.
- Файл конфигурации сервера StarVault.
- Вывод лога сервера StarVault, как описано в разделе "Логи сервера StarVault" учебника "Устранение неполадок StarVault".

Конфигурация сервера StarVault важна для устранения проблем с запуском, лог может выявить полезные предупреждения или ошибки StarVault, которые могут иметь первопричины, связанные с операционной средой.

Сбор информации из системной среды и логов сервера для выявления первопричины может быть трудным процессом, особенно в случае сбоя в работе.

Это задача, которая идеально подходит для автоматизации, чтобы результаты были последовательными, повторяемыми и поступали по мере необходимости.

Инструмент, помогающий операторам StarVault собирать и интерпретировать эту информацию, снижает нагрузку на поиск неисправностей, сокращает время обнаружения первопричины и значительно уменьшает время простоя во время отключения.

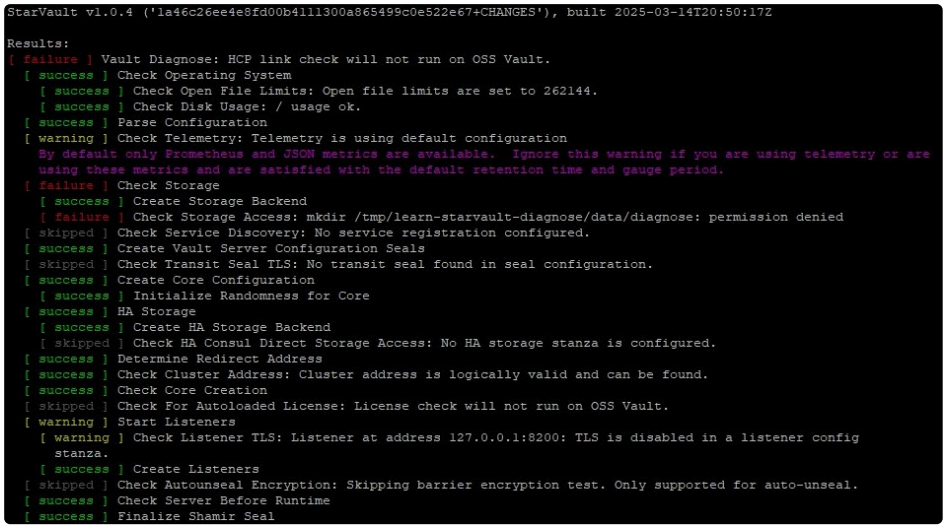
2. Решение

В StarVault есть подкоманда `diagnose` для команды `operator` CLI, которая помогает операторам выявлять основные причины наиболее часто встречающихся проблем с конфигурацией и запуском сервера.

Вы можете использовать команду с фактической конфигурацией сервера, который необходимо продиагностировать. Типичный рабочий процесс заключается в том, чтобы вызвать диагностику конфигурации и данных сервера во время его простоя. Существует также опция, позволяющая выполнять диагностику на работающем сервере, о которой вы узнаете позже.

Более подробную информацию о диагностике можно получить из Документации по диагностике оператора или вызвав `starvault operator diagnose -help` в терминале.

Приведем пример фактического вывода, чтобы ознакомить вас с типами проверок, выполняемых и сообщаемых диагностикой.



В результате диагностики был получен отказ в отношении хранилища, а также несколько предупреждений об использовании диска, лицензировании и TLS.

Команда предназначена для объяснения результатов на понятном языке, поэтому результаты часто не требуют пояснений. Она также содержит рекомендации, помогающие устранять предупреждения и сбои, такие как, например, рекомендация иметь как минимум 1 ГБ свободного места на каждом разделе.

2.1. Что проверяет диагностика?

На высоком уровне диагностика проверяет и сообщает об следующих распространенных причинах проблем с запуском сервера:

- **Среда**
  - Ограничения пользователя: максимальное количество открытых файлов
  - Емкости для хранения

- **Сборка**
  - Доступ к настроенному хранилищу
  - Доступ к хранилищу HA
  - Создание ограничений
  - Установочное ядро
  - Адрес перенаправления
  - Адрес кластера
  - Прослушиватели
    - Настройка TLS
  - Блокировка

Вы узнаете больше о типах сбоев, предупреждениях и рекомендациях по диагностике в сценарии.

3. Предпосылки

Для выполнения действий, описанных в сценарии, вам понадобятся:

- StarVault.
  - Руководство по установке StarVault (поможет вам справиться с установкой).
- jq для обработки вывода JSON из StarVault CLI.

4. Представление сценария

Запустите локальный сервер StarVault из командной строки в терминале, используя предоставленный файл конфигурации в качестве примера.

Воспользуйтесь функцией диагностики для проверки конфигурации в качестве примера.

Используя информацию из программы диагностики, устраните обнаруженный сбой в среде.

5. Подготовка среды

Создайте временный каталог, в котором будет храниться работа, которую вы будете выполнять в этом сценарии, и присвойте путь к нему переменной окружения LEARN_VAULT.

BASH

```
$ mkdir -pm 0000 /tmp/learn-starvault-diagnose/data && \
  export LEARN_STARVAULT=/tmp/learn-starvault-diagnose
```

6. Запись примера конфигурации

Начните сценарий с примера файла конфигурации, `starvault-server.hcl`.

Запишите его в домашний каталог сценария.

```bash
$ cat > "${LEARN_STARVAULT}"/starvault-server.hcl << EOF
api_addr                = "http://127.0.0.1:8200"
cluster_addr            = "http://127.0.0.1:8201"
cluster_name            = "learn-diagnose-cluster"
default_lease_ttl       = "10h"
disable_mlock           = true
max_lease_ttl           = "10h"
pid_file                = "$LEARN_STARVAULT/pidfile"
ui                      = true

listener "tcp" {
  address       = "127.0.0.1:8200"
  tls_disable   = "true"
  tls_cert_file = "starvault.crt"
  tls_key_file  = "starvault.key"
}

backend "file" {
  path    = "$LEARN_STARVAULT/data"
  node_id = "learn-diagnose-server"
}
EOF
```

7. Выполнение диагностики

Выполните команду диагностики, чтобы проверить начальную конфигурацию примера.

```bash
$ starvault operator diagnose -config $LEARN_STARVAULT/starvault-server.hcl
```

Вывод должен быть похож на этот пример.

```bash
StarVault v1.2.0 ('1a46c26ee4e8fd00b4111300a865499c0e522e67+CHANGES'), built
2025-03-14T20:50:17Z

Results:
[ failure ] Vault Diagnose: HCP link check will not run on OSS Vault.
  [ success ] Check Operating System
    [ success ] Check Open File Limits: Open file limits are set to 262144.
    [ success ] Check Disk Usage: / usage ok.
  [ success ] Parse Configuration
```

```
  [ warning ] Check Telemetry: Telemetry is using default configuration
    By default only Prometheus and JSON metrics are available.  Ignore this
warning if you are using telemetry or are
    using these metrics and are satisfied with the default retention time and
gauge period.
  [ failure ] Check Storage
    [ success ] Create Storage Backend
    [ failure ] Check Storage Access: mkdir /tmp/learn-starvault-
diagnose/data/diagnose: permission denied
  [ skipped ] Check Service Discovery: No service registration configured.
  [ success ] Create Vault Server Configuration Seals
  [ skipped ] Check Transit Seal TLS: No transit seal found in seal
configuration.
  [ success ] Create Core Configuration
    [ success ] Initialize Randomness for Core
  [ success ] HA Storage
    [ success ] Create HA Storage Backend
  [ success ] Determine Redirect Address
  [ success ] Check Cluster Address: Cluster address is logically valid and can
be found.
  [ success ] Check Core Creation
  [ skipped ] Check For Autoloaded License: License check will not run on OSS
Vault.
  [ warning ] Start Listeners
    [ warning ] Check Listener TLS: Listener at address 127.0.0.1:8200: TLS is
disabled in a listener config
      stanza.
    [ success ] Create Listeners
  [ skipped ] Check Autounseal Encryption: Skipping barrier encryption test.
Only supported for auto-unseal.
  [ success ] Check Server Before Runtime
  [ success ] Finalize Shamir Seal
```

Диагностика привела к общему сбою в строке 4, а в строках 12 и 14 есть сообщение о сбое хранилища, а также предупреждения о конфигурации TLS слушателя в строках 27-28 и о конфигурации телеметрии в строке 9.

Сбой в строке 14 **Check Storage Access: mkdir /tmp/learn-starvault-diagnose/data/diagnose: permission denied** указывает на проблему с каталогом данных StarVault. Попробуйте подтвердить режимы в этом каталоге.

BASH

```
$ ls -l /tmp/learn-starvault-diagnose/
total 4
d---------. 2 nova wheel   6 May 23 03:24 data
-rw-r--r--. 1 nova wheel 596 May 23 03:22 starvault-server.hcl
```

Слишком строгие разрешения на каталог данных.

Чтобы понять сообщения журнала StarVault об этой проблеме на данном этапе, попробуйте запустить сервер StarVault с такой конфигурацией.

```bash
$ starvault server -config $LEARN_STARVAULT/starvault-server.hcl

WARNING! Unable to read storage migration status.
2025-05-23T03:39:50.688-0400 [INFO]  proxy environment: http_proxy="" https_proxy="" no_proxy=""
2025-05-23T03:39:50.688-0400 [WARN]  storage migration check error: error="open /tmp/learn-starvault-diagnose/data/core/_migration: permission denied"
```

При попытке получить доступ к ключу миграции основного хранилища сервер StarVault выдает аналогичную ошибку о запрете доступа к пути данных.

Нажмите `Ctrl`+`C` для остановки сервера.

Измените режим на `0700`, чтобы StarVault мог писать в файловое хранилище, сконфигурированное для этого пути.

```bash
$ chmod 0700 /tmp/learn-starvault-diagnose/data
```

Выполните команду диагностики еще раз, чтобы повторно проверить конфигурацию.

```bash
$ starvault operator diagnose -config $LEARN_STARVAULT/starvault-server.hcl
```

Вывод должен быть похож на этот пример.

```bash
StarVault v1.2.0 ('1a46c26ee4e8fd00b4111300a865499c0e522e67+CHANGES'), built 2025-03-14T20:50:17Z

Results:
[ warning ] Vault Diagnose: HCP link check will not run on OSS Vault.
  [ success ] Check Operating System
    [ success ] Check Open File Limits: Open file limits are set to 262144.
    [ success ] Check Disk Usage: / usage ok.
  [ success ] Parse Configuration
  [ warning ] Check Telemetry: Telemetry is using default configuration
    By default only Prometheus and JSON metrics are available.  Ignore this warning if you are using telemetry or are
    using these metrics and are satisfied with the default retention time and gauge period.
  [ success ] Check Storage
    [ success ] Create Storage Backend
    [ success ] Check Storage Access
  [ skipped ] Check Service Discovery: No service registration configured.
  [ success ] Create Vault Server Configuration Seals
  [ skipped ] Check Transit Seal TLS: No transit seal found in seal
```

```
configuration.
  [ success ] Create Core Configuration
    [ success ] Initialize Randomness for Core
  [ success ] HA Storage
    [ success ] Create HA Storage Backend
  [ success ] Determine Redirect Address
  [ success ] Check Cluster Address: Cluster address is logically valid and can
be found.
  [ success ] Check Core Creation
  [ skipped ] Check For Autoloaded License: License check will not run on OSS
Vault.
  [ warning ] Start Listeners
    [ warning ] Check Listener TLS: Listener at address 127.0.0.1:8200: TLS is
disabled in a listener config
      stanza.
    [ success ] Create Listeners
  [ skipped ] Check Autounseal Encryption: Skipping barrier encryption test.
Only supported for auto-unseal.
  [ success ] Check Server Before Runtime
  [ success ] Finalize Shamir Seal
```

Проблема с хранилищем устранена, но в выводе диагностики осталось как минимум два предупреждения.

В зависимости от среды вы можете заметить и другие предупреждения, отсутствующие в примере. Например предупреждения о емкости хранилища, открытых файлах и т.д. Вы можете попытаться устранить их, чтобы получить положительный результат, но для целей данного руководства это излишне.

Одно из предупреждений сообщает, что в слушателе не включен TLS.

Это предупреждение очень важно, но оно не мешает вам работать с StarVault (например, в качестве разработчика или QA).

Лучшие примеры, описанные в Документации по усилению производственной защиты, рекомендуют использовать StarVault со включенным сквозным TLS для производственного использования.

Учитывая отсутствие сбоев, сервер StarVault должен запуститься даже при наличии предупреждений в результатах диагностики.

Запустите сервер снова.

BASH

```
$ starvault server -config $LEARN_STARVAULT/starvault-server.hcl
==> StarVault server configuration:

Administrative Namespace:
             Api Address: http://127.0.0.1:8200
```

```
                     Cgo: disabled
          Cluster Address: https://127.0.0.1:8201
   Environment Variables: BASH_FUNC_which%%, DBUS_SESSION_BUS_ADDRESS, GODEBUG,
GOTRACEBACK, HISTSIZE, HOME, HOSTNAME, LANG, LEARN_STARVAULT, LESSOPEN, LOGNAME,
LS_COLORS, MAIL, PATH, PWD, SHELL, SHLVL, STARVAULT_ADDR, STARVAULT_DETAILED,
STARVAULT_EXPERIMENTS, STARVAULT_FORMAT, STARVAULT_LICENSE,
STARVAULT_LICENSE_PATH, SUDO_COMMAND, SUDO_GID, SUDO_UID, SUDO_USER, TERM, USER,
_, which_declare
              Go Version: go1.22.1
              Listener 1: tcp (addr: "127.0.0.1:8200", cluster address:
"127.0.0.1:8201", max_request_duration: "1m30s", max_request_size: "33554432",
tls: "disabled")
               Log Level:
                   Mlock: supported: true, enabled: false
           Recovery Mode: false
                 Storage: file
                 Version: StarVault v1.2.0, built 2025-03-14T20:50:17Z
             Version Sha: 1a46c26ee4e8fd00b4111300a865499c0e522e67+CHANGES

==> StarVault server started! Log data will stream in below:

2025-05-23T03:42:45.028-0400 [INFO]  proxy environment: http_proxy=""
https_proxy="" no_proxy=""
```

Сервер StarVault запустился, подтвердив решение проблемы с разрешением пути хранения.

8. Очистка

- В терминале, где запущен сервер StarVault, нажмите `Ctrl`+`C`, чтобы остановить сервер.
- Удалите временный каталог, содержащий конфигурацию и данные сервера StarVault.

BASH

```
$ rm -rf "$LEARN_STARVAULT"
```

9. Резюме

Вы узнали о подкоманде диагностики и о том, как использовать ее в конфигурации StarVault для проверки общих проблем.

Ограничение доступа к API и хранилищам во время ликвидации последствий инцидента

При работе с любым программным обеспечением могут возникнуть чрезвычайные ситуации или непредвиденные действия, и StarVault не является исключением.

В статье показано, как использовать критически важные функции StarVault с вашими процессами и процедурами для быстрого реагирования на инциденты.

1. Задача

Операторам и специалистам по реагированию требуются функции и возможности, которые эффективно поддерживают резервные механизмы восстановления доступа, используемые при реагировании на инциденты.

При возникновении таких инцидентов, как утечка учетных данных, вторжение или атаки типа "отказ в обслуживании" (DOS), своевременное устранение последствий имеет огромное значение для поддержки операционных целей бизнеса.

2. Решение

Узнайте и используйте эти две важные функции StarVault, которые призваны помочь операторам эффективно реагировать на инциденты.

- **Печать:** При обычной работе StarVault работает в *незапечатанном* состоянии. Однако оператор с достаточными привилегиями может в любой момент запечатать работающий StarVault. При опечатывании сервер StarVault выбрасывает из памяти ключ для разблокировки данных и поэтому физически не может отвечать на все запросы, за исключением проверок состояния и снятия печати.

3. Предварительные условия

Для выполнения всех шагов, описанных в этом практическом сценарии, вам потребуется:

- StarVault двоичный файл, установленный в вашем системном пути.
- jq используется для вывода JSON в красивом формате.

3.1. Требования к политике

В рамках данного руководства для работы с StarVault можно использовать `root` токен. Однако рекомендуется использовать `root` токены только для первоначальной настройки или в чрезвычайных ситуациях. Лучше всего использовать токены с соответствующим набором политик в зависимости от вашей роли в организации. Если вы не запускаете StarVault локально в режиме разработки (- `dev`), рекомендуется пройти аутентификацию в StarVault и получить токен с соответствующим набором политик в соответствии с вашей ролью в организации.

Для выполнения всех задач, продемонстрированных в этом руководстве, ваша политика должна включать следующие разрешения:

Требования к политике

Если вы подключаетесь к серверу StarVault, не находящемуся в режиме разработки, ваш токен должен иметь политику со следующими разрешениями:

```
# Seal StarVault
path "sys/seal" {
  capabilities = [ "create", "update", "sudo" ]
}

# Unseal StarVault
path "sys/unseal" {
  capabilities = [ "create", "update", "sudo" ]
}
```

Если вы не знакомы с политиками, пройдите учебное пособие по политикам.

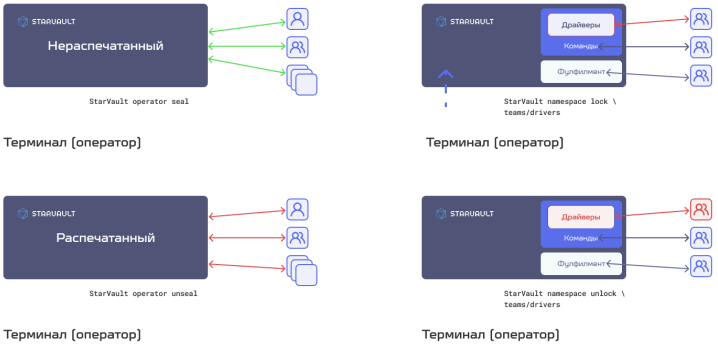
4. Роли

В сквозном сценарии, описанном в этом руководстве, участвуют две персоны:

- `operator` с привилегированными возможностями для запечатывания и разпечатывания StarVault, а также блокировки и разблокировки конечных точек API.
- `driver` использует метод аутентификации по имени пользователя и паролю, включенный в пространство имен драйверов, для аутентификации в StarVault.

5. Введение в сценарий

Как показано слева, когда StarVault запечатывается, это затрагивает всех пользователей, и они не могут его использовать.



Вы будете использовать терминальные сессии для работы с одним сервером StarVault с Integrated Storage (Raft) и пройдете ряд практических сценариев, чтобы узнать, как запечатывать и снимать печать с StarVault с помощью StarVault CLI, HTTP API или веб-интерфейса.

По окончании обучения вы также можете очистить среду сценария с помощью предоставленных примеров команд.

5.1. Создание сценарной среды

Создайте временный каталог для хранения файлов, созданных для этого сценария, и присвойте путь к нему переменной окружения LEARN_VAULT.

```
$ mkdir -p /tmp/learn-starvault/data && export LEARN_STARVAULT=/tmp/learn-
starvault
```

5.2. Создание конфигурации сервера StarVault

Создайте минимальную конфигурацию для одного сервера StarVault, использующего хранилище Raft.

```
$ cat > $LEARN_STARVAULT/starvault-server.hcl << EOF
api_addr                = "http://127.0.0.1:8200"
cluster_addr            = "http://127.0.0.1:8201"
cluster_name            = "learn-starvault"
default_lease_ttl       = "10h"
disable_mlock           = true
max_lease_ttl           = "10h"
ui                      = true

listener "tcp" {
  address       = "127.0.0.1:8200"
  tls_disable   = "true"
}
```

```
backend "raft" {
  path    = "/tmp/learn-starvault/data"
  node_id = "learn-starvault-1"
}
EOF
```

5.3. Запустите один сервер StarVault

Откройте сеанс терминала и запустите сервер StarVault, используя конфигурацию, которую вы только что создали.

BASH

```
$ starvault server -config $LEARN_STARVAULT/starvault-server.hcl
```

Сервер запускается неинициализированным и запечатанным.

5.4. Инициализация, снятие печати и аутентификация

В **другом терминальном сеансе** экспортируйте переменную среды STARVAULT_ADDR для адресации сервера StarVault и экспортируйте переменную среды LEARN_VAULT для определения каталога сценария.

BASH

```
$ export STARVAULT_ADDR=http://127.0.0.1:8200 LEARN_STARVAULT=/tmp/learn-
starvault
```

Для простоты в этом руководстве инициализируйте StarVault с 1 ключевым ресурсом и ключевым порогом 1 и запишите результаты в файл `.starvault-init` в каталоге проекта.

BASH

```
$ starvault operator init \
    -key-shares=1 \
    -key-threshold=1 \
    | head -n3 \
    | cat > $LEARN_STARVAULT/.starvault-init
```

В выводе ничего не будет при успешном выполнении команды.

Экспортируйте значение ключа разблокировки в переменную окружения UNSEAL_KEY.

BASH

```
$ export UNSEAL_KEY=$(grep 'Unseal Key 1'  $LEARN_STARVAULT/.starvault-init |
awk '{print $NF}')
```

Снимите печать с StarVault, используя значение **Unseal Key 1** из файла `.starvault-init`.

BASH

```
$ starvault operator unseal $UNSEAL_KEY
```

Успешный выход из распечатывания StarVault должен напоминать этот пример:

Key	Value
Seal Type	shamir
Initialized	true
Sealed	false
Total Shares	1
Threshold	1
Version	1.2.0
Build Date	2025-03-14T20:50:17Z
Storage Type	raft
Cluster Name	learn-starvault
Cluster ID	fb4c6314-f2df-2cbc-aa21-9e794f44003a
HA Enabled	true
HA Cluster	n/a
HA Mode	standby
Active Node Address	<none>
Raft Committed Index	55
Raft Applied Index	55

Экспортируйте начальное значение `root` токена в переменную окружения ROOT_TOKEN.

```bash
$ export ROOT_TOKEN=$(grep 'Initial Root Token' $LEARN_STARVAULT/.starvault-init
| awk '{print $NF}')
```

войдите в систему с помощью `starvault login`, передав значение **Initial Root Token** из файла `.starvault-init`.

```bash
$ starvault login -no-print $ROOT_TOKEN
```

Эта команда не должна выводить никаких результатов в случае успеха. Если вы хотите подтвердить, что вход был успешным, попробуйте выполнить поиск токена и убедитесь, что

ваши политики токенов содержат `root`.

Для простоты в этом сценарии вы будете использовать `root` токен. Однако в реальных производственных средах `root` токены следует тщательно охранять и использовать только в строго контролируемых целях. Для получения более подробной информации ознакомьтесь с документацией по корневым токенам.

```bash
$ starvault token lookup | grep policies
```

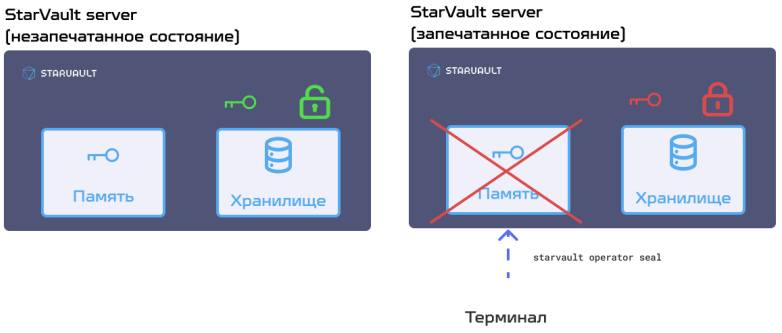
Успешный вывод должен содержать следующее.

```
policies                                [root]
```

Вы готовы к первому сценарию.

5.5. Запечатывание StarVault

StarVault, работающий в незапечатанном состоянии, использует главный ключ шифрования. Он нужен для расшифровки ключа шифрования, хранящегося в хранилище StarVault, который сам используется для шифрования и расшифровки секретов в хранилище. В этом примере StarVault запечатывается с помощью CLI из терминальной сессии; главный ключ шифрования удаляется из памяти, и дальнейшее шифрование или расшифровка из хранилища невозможны до тех пор, пока StarVault снова не будет запечатан.



Если произошел такой инцидент, как вторжение или утечка учетных данных, запечатывание StarVault в ответ - это самый тяжелый инструмент, который вы можете использовать. Ключ, использовавшийся для расшифровки данных, будет немедленно удален из памяти, и StarVault больше не сможет расшифровать ничего в своем хранилище, пока снова не будет снята печать.

Запечатанный StarVault фактически больше не доступен для стандартного использования. Об этом следует помнить, особенно при использовании корпоративной репликации, поскольку запечатанный кластер больше не будет участвовать в репликации. Запечатанный первичный кластер не позволит всем вторичным кластерам реплицироваться с ним, а запечатанный вторичный кластер будет рассинхронизирован с первичным.

Оператор, использующий токен с достаточными возможностями, как описано в разделе Требования к политике, может запечатать StarVault с помощью CLI `starvault`, HTTP API /sys/seal или с помощью веб-интерфейса.

5.5.1. CLI команда

(**Пользователь:** operator)

Запечатайте ваш StarVault сервер.

BASH

```
$ starvault operator seal
Success! StarVault is sealed.
```

Вы можете проверить статус StarVault для подтверждения.

BASH

```
$ starvault status
```

Key	Value
Seal Type	shamir
Initialized	true
Sealed	true
Total Shares	1
Threshold	1
Unseal Progress	0/1
Unseal Nonce	n/a
Version	1.2.0
Build Date	2025-03-14T20:50:17Z
Storage Type	inmem_transactional_ha
HA Enabled	true

Вывод должен показать, что значение `Sealed` равно `true`.

Вы также можете перейти на терминал, с которого был запущен сервер StarVault, чтобы посмотреть на оперативное логирование. Конец лога должен быть похож на этот пример.

```
2025-05-20T04:21:00.396-0400 [INFO]  core.cluster-listener: rpc listeners
successfully shut down
2025-05-20T04:21:00.396-0400 [INFO]  core: cluster listeners successfully shut
down
2025-05-20T04:21:00.396-0400 [INFO]  core: starvault is sealed
```

Обратите внимание на последнюю строку, в которой конкретно указано ядро: `starvault is sealed`; это сигнал из лога для вас как оператора, что StarVault теперь запечатан.

Если вы попытаетесь войти в StarVault сейчас, вы обнаружите ошибку, подобную той, что показана в этом примере:

```
$ starvault login -no-print $ROOT_TOKEN
Error authenticating: error looking up token: Error making API request.

URL: GET http://127.0.0.1:8200/v1/auth/token/lookup-self
Code: 503. Errors:

* StarVault is sealed
```

5.5.2. Вызов API с использование cURL

(**Пользователь:** operator)

Запечатайте ваш StarVault сервер.

```
$ curl \
    --header "X-Vault-Token: $ROOT_TOKEN" \
    --request PUT \
    $STARVAULT_ADDR/v1/sys/seal
```

Вы можете проверить статус StarVault для подтверждения.

```
$ curl --silent $STARVAULT_ADDR/v1/sys/seal-status | jq
```

Вы также можете перейти на терминал, с которого был запущен сервер StarVault, чтобы посмотреть на оперативный лог. Конец лога должен быть похож на этот пример.

```
2025-05-20T04:23:31.966-0400 [INFO]  core.cluster-listener: rpc listeners
successfully shut down
2025-05-20T04:23:31.966-0400 [INFO]  core: cluster listeners successfully shut
```

```
down
2025-05-20T04:23:31.967-0400 [INFO]  core: starvault is sealed
```

Обратите внимание на последнюю строку, в которой конкретно указано ядро: `starvault is sealed`; это сигнал из лога для вас как оператора, что StarVault теперь запечатан.

5.5.3. Веб-интерфейс

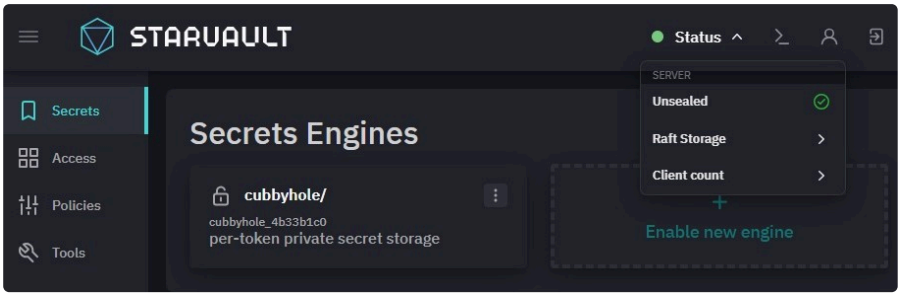
(**Пользователь:** operator)

Откройте браузер и перейдите в пользовательский интерфейс StarVault по адресу http://127.0.0.1:8200.

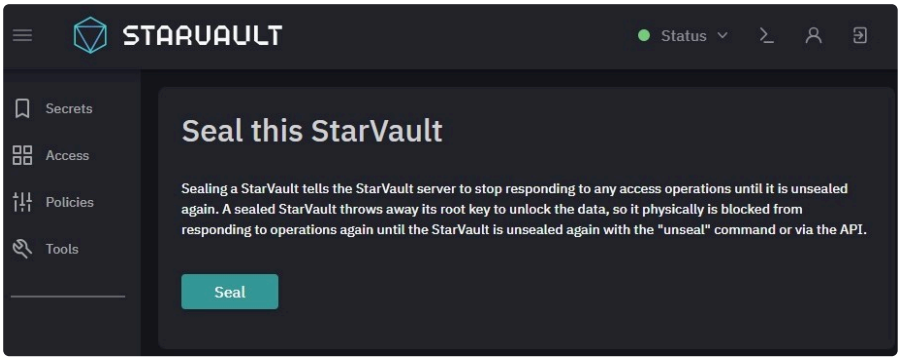
Войдите в систему со значением токена `root` и запечатайте свой сервер StarVault.

---

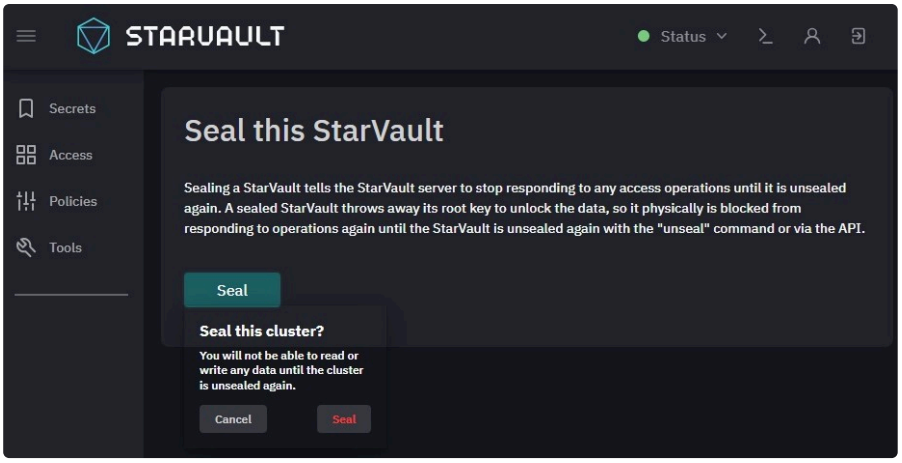
1. Нажмите **Status** и выберите **Unsealed** в меню.



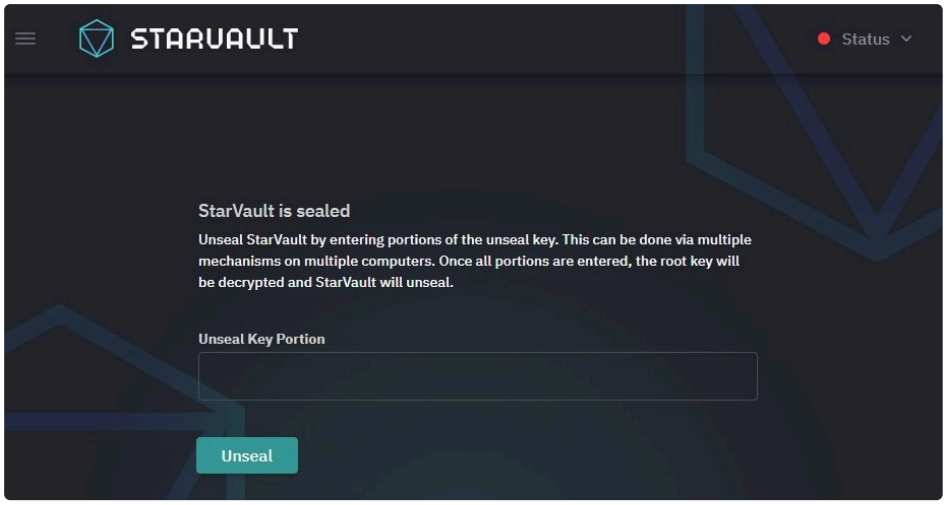
2. Кликните на **Seal**



3. Подтвердите намерение запечатать StarVault, нажав кнопку **Seal** в диалоговом окне подтверждения.



Хранилище запечатано. **Status - индикатор** состояния, который раньше был зеленым, теперь стал красным.



Вы можете снова разблокировать StarVault, указав ключ разблокировки в текстовом поле **Unseal Key Portion** и нажав **Unseal**.

Вы можете перейти на терминал, с которого был запущен сервер StarVault, чтобы посмотреть оперативный лог. Конец лога должен быть похож на этот пример.

```
2025-05-20T05:30:45.121-0400 [INFO]  core.cluster-listener: rpc listeners
successfully shut down
2025-05-20T05:30:45.121-0400 [INFO]  core: cluster listeners successfully shut
down
2025-05-20T05:30:45.121-0400 [INFO]  core: starvault is sealed
```

BASH

Обратите внимание на последнюю строку, где конкретно указано ядро: хранилище запечатано; это сигнал из лога для вас, как оператора, что хранилище теперь запечатано.

Если StarVault запечатан, оператор должен разблокировать его, чтобы восстановить нормальную работу.

В примерах этого руководства используется печать на основе Shamir's Secret Sharing с ключом для снятия печати, но в обычных производственных установках используется автоматическая печать на основе облака. Помните, что если сервер StarVault, использующий автоматическую разблокировку, запечатан, он автоматически разблокируется при перезапуске. Будьте осторожны при перезапуске таких серверов.

Используйте значение разблокирующего ключа, записанное ранее, чтобы снять печать с StarVault в рамках подготовки к следующему сценарию.

5.5.4. CLI команда

(**Пользователь:** operator)

Разблокируйте StarVault.

```bash
$ starvault operator unseal $UNSEAL_KEY
```

Перейдите в терминал, с которого вы запустили сервер StarVault, и посмотрите на вывод лога. Конец лога должен быть похож на этот пример.

```bash
2025-05-20T05:32:25.008-0400 [INFO]  core: vault is unsealed
2025-05-20T05:32:25.039-0400 [INFO]  core: post-unseal setup starting
2025-05-20T05:32:25.048-0400 [INFO]  core: post-unseal setup complete
```

Обратите внимание на ядро линии: `core: vault is unsealed`.

Теперь хранилище не запечатано и готово к нормальной работе.

5.5.5. Вызов API с помощью cURL

(**Пользователь:** operator)

Разблокируйте StarVault.

```bash
$ curl \
    --data "{\"key\": \"$UNSEAL_KEY\"}" \
    --header "X-Vault-Token: root" \
    --request PUT \
    --silent \
    $STARVAULT_ADDR/v1/sys/unseal \
    | jq
```

Вывод команды должен быть похож на этот пример.

```bash
{
  "type": "shamir",
  "initialized": true,
```

```
  "sealed": false,
  "t": 1,
  "n": 1,
  "progress": 0,
  "nonce": "",
  "version": "1.2.0",
  "build_date": "2025-03-14T20:50:17Z",
  "migration": false,
  "cluster_name": "learn-starvault",
  "cluster_id": "fb4c6314-f2df-2cbc-aa21-9e794f44003a",
  "recovery_seal": false,
  "storage_type": "raft"
}
```


Обратите внимание, что значение `sealed` равно `false`.

Перейдите на терминал, с которого вы запустили сервер StarVault, и посмотрите на лог. Конец лога должен быть похож на этот пример.

BASH

```
2025-05-20T05:39:35.379-0400 [INFO]  core: vault is unsealed
2025-05-20T05:39:35.412-0400 [INFO]  core: post-unseal setup starting
2025-05-20T05:39:35.420-0400 [INFO]  core: post-unseal setup complete
```

Обратите внимание на ядро линии: `core: vault is unsealed`.

Теперь хранилище не запечатано и готово к нормальной работе.

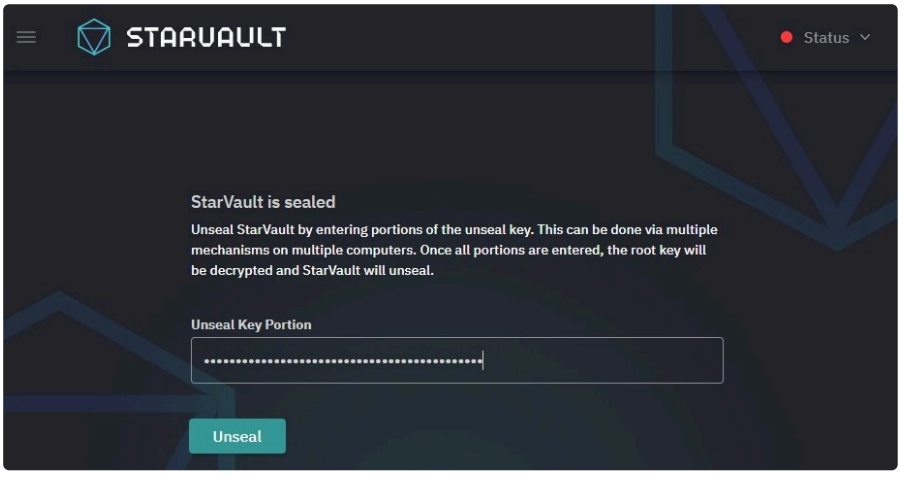
5.5.6. Веб-интерфейс

(**Пользователь:** operator)

Снимите блокировку с StarVault

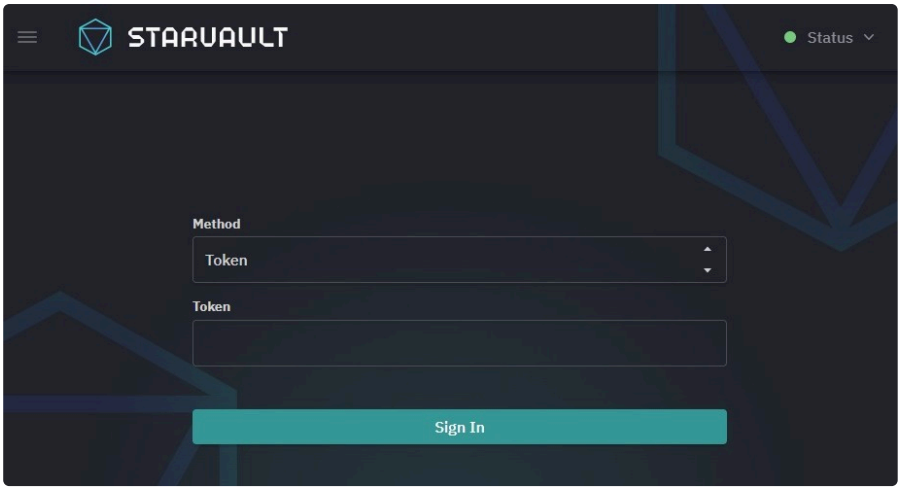
---

1. В окне **Unseal** введите значение ключа разблокировки в текстовое поле **Unseal Key Portion**.



2. Нажмите **Unseal**

3. С StarVault снимается блокировка и он предлагает войти в систему



Вы можете перейти на терминал, с которого был запущен сервер StarVault, и посмотреть на оперативный лог.

Конец логирования должен быть похож на этот пример.

```
2025-05-20T05:43:49.698-0400 [INFO]  core: vault is unsealed
2025-05-20T05:43:49.726-0400 [INFO]  core: post-unseal setup starting
2025-05-20T05:43:49.733-0400 [INFO]  core: post-unseal setup complete
```

Обратите внимание на ядро линии: `core: vault is unsealed`.

Теперь хранилище не запечатано и готово к нормальной работе.

6. Очистка

Если вы хотите очистить свое окружение после завершения обучения, выполните действия, описанные в этом разделе.

Снимите настройки переменных окружения.

```
$ unset ROOT_TOKEN UNSEAL_KEY STARVAULT_ADDR LEARN_STARVAULT
```

Вы можете остановить сервер StarVault, нажав `Ctrl`+`C`, когда сервер запущен. Или выполните следующую команду.

```
$ pgrep -f starvault | xargs kill
```

Удалите каталог окружения сценария.

```bash
$ rm -rf /tmp/learn-starvault
```

7. Резюме

Вы узнали о важной функции реагирования на инциденты, доступной в StarVault.