Установка с помощью HELM

StarVault работает с Kubernetes в следующих режимах: **dev**, **standalone**, **ha** и **external**.

1. Helm-чарт

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

Хотя Helm-чарт автоматически настраивает сложные ресурсы и подгоняет конфигурацию под требования, он не поддерживает работу StarVault автоматически. Полезно иметь навыки вести мониторинг, делать резервные копии, устанавливать обновления и тому подобное в кластере StarVault, чтобы поддерживать ту часть работы StarVault, которую не можете автоматизировать.

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.

2. Установка StarVault

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.

С помощью команды helm install установите последний релиз Helm-чарта для StarVault.

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

Команда `helm install` принимает параметры для переопределения значений конфигурации по умолчанию – как встроенных, так и определенных в файле.

Переопределите значение конфигурации `server.dev.enabled`:

```bash
helm install starvault oci://hub.orionsoft.ru/public/starvault --namespace
starvault --create-namespace --set "server.dev.enabled=true"
```

Переопределите все значения конфигурации, которые есть в файле:

```bash
cat override-values.yml
server:
  ha:
    enabled: true
    replicas: 5
##

helm install starvault oci://hub.orionsoft.ru/public/starvault --namespace
starvault --create-namespace --values override-values.yml
```

Скачать файл с переменными:

```bash
helm show values oci://hub.orionsoft.ru/public/starvault > override-values.yaml
```

2.1. Режим разработки (dev-режим)

Helm-чарт может запустить сервер StarVault в режиме разработки (режим `dev`). При этом устанавливается одиночный сервер StarVault с бэкендом хранения в оперативной памяти.

Режим разработки подходит для учебных и демонстрационных сред, но **НЕ** рекомендуется для боевой среды.

Установите последнюю версию Helm-чарта для StarVault в режиме разработки.

```bash
helm install starvault oci://hub.orionsoft.ru/public/starvault --namespace
starvault --create-namespace --set "server.dev.enabled=true"
```

2.2. Автономный режим (standalone)

По умолчанию Helm-чарт запускается в режиме `standalone`. В этом режиме устанавливается одиночный сервер StarVault с бэкендом файлового хранилища.

Установите последнюю версию Helm-чарта для StarVault в автономном режиме.

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

2.3. Режим высокой доступности (HA)

Helm-чарт можно запустить в режиме высокой доступности (HA). При этом устанавливаются три сервера StarVault с существующим бэкендом хранения Consul. Предполагается, что Consul устанавливается с помощью Helm-чарта для Consul.

Установите последнюю версию Helm-чарта для StarVault в режиме высокой доступности.

```bash
helm install starvault oci://hub.orionsoft.ru/public/starvault --namespace
starvault --create-namespace --set "server.ha.enabled=true"
```

3. Пользовательский интерфейс StarVault

В целях безопасности пользовательский интерфейс StarVault включен, но **НЕ** доступен как сервис. Доступ к пользовательскому интерфейсу StarVault можно открыть через переадресацию портов или через значение конфигурации UI.

Откройте доступ к пользовательскому интерфейсу StarVault через переадресацию портов:

```bash
$ kubectl port-forward starvault-0 8200:8200 -n starvault
Forwarding from 127.0.0.1:8200 -> 8200
Forwarding from [::1]:8200 -> 8200
##...
```

4. Инициализация и распечатывание StarVault

После установки Helm-чарта для StarVault в режиме `standalone` или `ha` необходимо инициализировать один из серверов StarVault. При инициализации генерируются учетные данные, необходимые для распечатывания всех серверов StarVault.

4.1. Инициализация и распечатывание с помощью интерфейса командной строки

Просмотрите все поды StarVault в текущем пространстве имен:

```
$ kubectl get pods -l app.kubernetes.io/name=starvault -n starvault

NAME                                    READY   STATUS    RESTARTS   AGE
starvault-0                            0/1     Running   0          1m49s
starvault-1                            0/1     Running   0          1m49s
starvault-2                            0/1     Running   0          1m49s
```

Инициализируйте один сервер StarVault с заданным по умолчанию количеством частей ключа и заданным по умолчанию пороговым количеством частей ключа:

```
$ kubectl -n starvault exec -it starvault-0 -- starvault operator init

Unseal Key 1: MBFSDepD9E6whREc6Dj+k3pMaKJ6cCnCUWcySJQymObb
Unseal Key 2: zQj4v22k9ixegS+94HJwmIaWLBL3nZHe1i+b/wHz25fr
Unseal Key 3: 7dbPPeeGGW3SmeBFFo04peCKkXFuuyKc8b2DuntA4VU5
Unseal Key 4: tLt+ME7Z7hYUATfWnuQdfCEgnKA2L173dptAwfmenCdf
Unseal Key 5: vYt9bxLr0+OzJ8m7c7cNMFj7nvdLljj0xWRbpLezFAI9

Initial Root Token: s.zJNwZlRrqISjyBHFMiEca6GF
##...
```

В выводе отображаются сгенерированные части ключа распечатвания и изначальный корневой ключ.

Распечатывайте сервер StarVault, добавляя части ключа, пока не будет достигнуто пороговое количество частей ключа:

```
kubectl -n starvault exec -it starvault-0 -- starvault operator unseal #Unseal Key 1
kubectl -n starvault exec -it starvault-0 -- starvault operator unseal #Unseal Key 2
kubectl -n starvault exec -it starvault-0 -- starvault operator unseal #Unseal Key 3
```

Повторите процесс распечатывания для всех подов сервера StarVault. Когда все поды сервера StarVault будут распечатаны, они сообщат о готовности: **READY** 1/1.

+

```
$ kubectl get pods -l app.kubernetes.io/name=starvault

NAME                                    READY   STATUS    RESTARTS   AGE
starvault-0                                1/1     Running   0          1m49s
starvault-1                                1/1     Running   0          1m49s
```

```
starvault-2                                    1/1     Running   0
1m49scode
```

5. Пробы

Пробы нужны для обнаружения сбоев, перепланировки и использования подов в Kubernetes. Helm-чарт предлагает настраиваемые пробы готовности и работоспособности, которые можно доработать под разнообразные сценарии использования.

Конечную точку `/sys/health` в StarVault можно доработать так, чтобы модифицировать процесс проверки работоспособности. Например, можно изменить пробу готовности StarVault, чтобы показывать готовые поды StarVault, даже если они еще не инициализированы и запечатаны, с помощью следующей пробы:

YAML

```yaml
server:
  readinessProbe:
    enabled: true
    path: '/v1/sys/health?standbyok=true&sealedcode=204&uninitcode=204'
```

С помощью этой доработанной пробы скрипт `postStart` может запускаться автоматически, как только под будет готов к дополнительной настройке.

6. Обновление StarVault в Kubernetes

Для обновления StarVault в Kubernetes используется обычная процедура обновления StarVault, за исключением того, что можно использовать Helm-чарт для обновления параметра Statef`ulSet сервера StarVault. Прежде чем читать дальше, важно понять, что имеется в виду под обычной процедурой обновления StarVault. Параметр `StatefulSet` в StarVault использует стратегию обновления *OnDelete*. Крайне важно использовать *OnDelete* вместо RollingUpdate, потому что резервные узлы должны быть обновлены раньше основного активного узла. Всегда избегайте отката к более старой версии StarVault в случае отказа.

Всегда перед обновлением делайте резервную копию ваших данных! StarVault не дает гарантий обратной совместимости для своего хранилища данных. Просто заменив недавно установленный двоичный файл StarVault на предыдущую версию, вы вряд ли сможете полноценно откатиться на старую версию StarVault, поскольку обновления могут вносить изменения в базовую структуру данных, делая их несовместимыми со старой версией. Если нужно откатиться до предыдущей версии StarVault, следует также откатить и ваше хранилище данных.

6.1. Обновление серверов StarVault

По умолчанию Helm установит самую последнюю версию чарта, которую найдет в репозитории. Рекомендуется перед обновлением указать версию чарта.

Пример:

```bash
helm install starvault oci://hub.orionsoft.ru/public/starvault --version 1.2.0 --namespace starvault --create-namespace
```

Чтобы инициировать обновление, задайте для параметра `server.image` значения, соответствующие желаемой версии StarVault; это могут быть значения в YAML-файле или в командной строке.
В целях демонстрации в примере ниже используется `hub.orionsoft.ru/public/starvault:1.2.0`.

```yaml
server:
  image:
    repository: 'hub.orionsoft.ru/public/starvault'
    tag: '1.2.0'
```

Затем выберите нужную версию Helm для установки.

Далее протестируйте обновление с помощью `--dry-run`, чтобы проверить изменения, отправленные в кластер Kubernetes.

```bash
$ helm upgrade starvault oci://hub.orionsoft.ru/public/starvault --version 1.2.0 --namespace starvault \
    --set='server.image.repository=hub.orionsoft.ru/public/starvault' \
    --set='server.image.tag=1.2.0' \
    --dry-run
```

Это не должно привести к каким-либо изменениям (хотя ресурсы обновлены). Если все стабильно, можно запустить `helm upgrade`. Команда `helm upgrade` должна обновить шаблон StatefulSet для серверов StarVault, однако ни один под удален не был. Для обновления поды следует удалять вручную. Удаление подов не приводит к удалению постоянно хранящихся данных.

Если StarVault не развернут в режиме `ha`, одиночный сервер StarVault можно удалить, запустив следующую команду:

```bash
$ kubectl delete pod <имя пода StarVault> -n starvault
```

Если StarVault развернут в режиме `ha`, то сначала следует обновить резервные поды. Благодаря встроенной функции обнаружения сервисов K8s (когда она включена в конфигурации сервера) StarVault будет автоматически менять метки пода, учитывая статус ведущего. По этим меткам можно фильтровать поды.

Например, можно выбрать все резервные поды StarVault:

```
kubectl get pods -l vault-active=false -n starvault
```
BASH

Выберите активный под StarVault:

```
kubectl get pods -l vault-active=true -n starvault
```
BASH

Затем последовательно можно удалить каждый под, который не является активным, и при этом все время сохранять кворум:

```
kubectl delete pod <имя пода StarVault> -n starvault
```
BASH

Если автоматическое распечатывание не используется, вновь запланированные резервные поды StarVault нужно распечатать:

```
kubectl exec -ti <имя пода> -n starvault -- vault operator unseal
```
BASH

Наконец, как только резервные узлы обновлены и распечатаны, можно удалить активный основной узел:

```
kubectl delete pod <имя основного узла StarVault> -n starvault
```
BASH

Как и в случае с резервными узлами, бывший основной тоже нужно распечатать:

```
kubectl exec -ti <имя пода> -n starvault -- vault operator unseal
```
BASH

Затем через пару секунд кластер StarVault должен выбрать новый активный основной узел. Кластер StarVault обновлен.

7. Защита конфиденциальных конфигураций StarVault

Helm для StarVault во время установки формирует файл конфигурации StarVault и хранит его в карте конфигурации Kubernetes. Некоторые конфигурации требуют включения конфиденциальных данных в файл конфигурации и после создания в Kubernetes будут храниться в незашифрованном виде.

В следующем примере показано, как добавить дополнительные файлы конфигурации в Helm для StarVault, чтобы зашифровать хранящиеся конфиденциальные конфигурации с помощью секретов Kubernetes.

Сначала создайте частичную конфигурацию StarVault с конфиденциальными настройками, которые StarVault загружает во время запуска:

```bash
$ cat <<EOF >>config.hcl
storage "mysql" {
username = "user1234"
password = "secret123!"
database = "vault"
}
EOF
```

Затем создайте секрет Kubernetes, содержащий эту частичную конфигурацию:

```bash
$ kubectl create secret generic vault-storage-config \
    --from-file=config.hcl
```

Создайте файл `values.yaml` и добавьте следующий блок в него:

```yaml
server:
  volumes:
    - name: userconfig-vault-storage-config
      secret:
        defaultMode: 420
        secretName: "vault-storage-config"
  volumeMounts:
    - name: "userconfig-vault-storage-config"
      mountPath: "/starvault/userconfig/vault-storage-config"
      readOnly: true
  extraArgs: "-config=/starvault/userconfig/vault-storage-config/config.hcl"
```

Наконец, смонтируйте этот секрет как дополнительный том.

Команда запуска StarVault:

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

8. Архитектура

Рекомендуем запускать StarVault в Kubernetes с той же обычной архитектурой, что используется при запуске в любой другой среде. У Kubernetes есть ряд преимуществ, которые могут упростить эксплуатацию кластера StarVault и будут приведены ниже. Однако стандартное руководство по развертыванию в боевой среде все равно стоит прочитать, даже если StarVault запускается в Kubernetes.

8.1. Чеклист по развертыванию в боевой среде

Сквозное шифрование по протоколу TLS. StarVault в боевой среде всегда следует использовать с шифрованием по протоколу TLS.
Промежуточные балансировщики нагрузки, обратные прокси-серверы должны шифровать проходящий через них трафик по протоколу TLS.
Таким образом, идущий в StarVault трафик всегда шифруется, что минимизирует риски, создаваемые промежуточными участниками процесса передачи.

Один процесс. StarVault должен быть единственным основным процессом, запущенным на машине. Это снижает риск того, что другой процесс, запущенный на той же машине, будет скомпрометирован и сможет взаимодействовать с StarVault. Сделать это можно с помощью настраиваемого поля affinity, которое есть в Helm для StarVault. Пример настройки Helm для StarVault на использование правил совместного существования (affinity) см. в официальной документации.

Функция аудита. Включенная функция аудита StarVault поддерживает несколько бэкендов аудита. Если включить функцию аудита, то сохранится история всех операций, выполненных StarVault, и можно будет получить доказательства в случае расследования злоупотреблений или компрометации. Журналы аудита надежно хэшируют любые конфиденциальные данные, но доступ все равно должен быть ограничен, чтобы предотвратить непреднамеренное раскрытие. Helm для StarVault включает в себя настраиваемую опцию auditStorage, которая предоставляет постоянный том для хранения журналов аудита. Пример настройки Helm для StarVault на использование аудита см. в официальной документации.

Обновление до новых версий без внесения изменений в уже работающие сервера. Для сохранения стабильности работы StarVault использует внешний бэкенд хранения, что позволяет не вносить изменения в сервера, на которых StarVault уже запущен. При обновлении до новых версий запускаются новые сервера с обновленной версией StarVault. Они подключаются к тому же общему бэкенду хранения и распечатываются. Затем старые сервера уничтожаются. В результате снижается необходимость в удаленном доступе и оркестрации процесса обновления, из-за которых могут возникнуть пробелы в безопасности.

Ограничение доступа к хранилищу. StarVault шифрует все хранящиеся данные, независимо от используемого бэкенда хранения. Хотя данные зашифрованы, злоумышленник с административными правами может изменить или удалить ключи, что приведет к повреждению или потере данных. Доступ к бэкенду хранения должен быть только у StarVault, чтобы избежать несанкционированного доступа или использования.

2025 orionsoft. Все права защищены.

Параметры Helm

Чарт гибко настраивается с помощью значений конфигурации Helm. Каждое значение по умолчанию настроено для обеспечения оптимального запуска StarVault. Перед запуском в боевой среде проверьте приведенные далее параметры.

1. Блок `global`

- `global` - это общие значения, влияющие на многие компоненты чарта.
  - `enabled` (`boolean: true`) — главный конфигурационный параметр, включающий/выключающий установку компонентов. Если параметр установлен в значение `true`, большинство компонентов будет установлено по умолчанию. Если значение `false`, то никакие компоненты не будут установлены по умолчанию, и их придется выбирать вручную, например, установив параметр `server.enabled` в значение `true`.
  - `imagePullSecrets` (`array: []`) — ссылка на секреты, которые используются при извлечении образов из частных реестров. Может быть указана как массив записей с сопоставлением имен или как массив имен:

    ```yaml
    imagePullSecrets:
      - name: image-pull-secret
    # или
    imagePullSecrets:
      - image-pull-secret
    ```

  - `tlsDisable` (`boolean: true`) — установка параметра в значение `true` меняет URL-адреса с https на http (например, переменная окружения `STARVAULT_ADDR=http://127.0.0.1:8200`, заданная на подах StarVault).
  - `externalVaultAddr` (`string: ""`) — адрес внешнего сервера StarVault, который используют Injector и провайдер интерфейса хранилища контейнеров (CSI-провайдер). Эта настройка отключит развертывание сервера StarVault. Сервисная учетная запись с правами на просмотр токенов создается автоматически, если параметр `server.serviceAccount.create=true` задан для использования внешним сервером StarVault.
  - `openshift` (`boolean: false`) — установка параметра в значение true активирует конфигурацию, специально заточенную под OpenShift, например, `NetworkPolicy`, `SecurityContext` и `Route`.
  - `psp` - Значения для настройки Политики безопасности подов.

- `enable` (`boolean: false`) — установка параметра в значение true активирует Политики безопасности подов для StarVault и StarVault Agent Injector.
- `annotations` (`dictionary: {}`) — это значение задает дополнительные аннотации, которые нужно добавить к Политикам безопасности подов. Это может быть либо YAML, либо многострочная шаблонная цепочка символов в формате YAML.

```yaml
annotations:
  seccomp.security.alpha.kubernetes.io/allowedProfileNames:
docker/default,runtime/default
  apparmor.security.beta.kubernetes.io/allowedProfileNames:
runtime/default
  seccomp.security.alpha.kubernetes.io/defaultProfileName:
runtime/default
  apparmor.security.beta.kubernetes.io/defaultProfileName:
runtime/default
# или
annotations:
  seccomp.security.alpha.kubernetes.io/allowedProfileNames:
docker/default,runtime/default
  apparmor.security.beta.kubernetes.io/allowedProfileNames:
runtime/default
  seccomp.security.alpha.kubernetes.io/defaultProfileName:
runtime/default
  apparmor.security.beta.kubernetes.io/defaultProfileName:
runtime/default
```

- `serverTelemetry` — значения для настройки метрик и телеметрии
  - `prometheusOperator` (`boolean: false`) — установка параметра в значение true активирует интеграцию с Prometheus Operator. Убедитесь, что настроен верхнеуровневый раздел `serverTelemetry` для получения дополнительной информации и необходимых значений конфигурации.

2. Блок server

- `server` — значения, которые настраивают запуск сервера StarVault в Kubernetes.
  - `enabled` (`boolean` or `string: "-"`) — установка параметра в значение true создаст сервер StarVault. Установка параметра в значение `"-"` задает значение `global.enabled` как значение по умолчанию.
    - `secretName` (`string: ""`) — имя секрета Kubernetes, в котором содержится корпоративная лицензия. Секрет должен находиться в том же пространстве имен, где установлен StarVault.

    - `secretKey` (`string: "license"`) — Ключ в секрете Kubernetes, в котором содержится корпоративная лицензия.
- `image` — значения, которые настраивают образ StarVault Docker.
    - `repository` (`string: "oci://hub.orionsoft.ru/public/starvault"`) — имя образа Docker для контейнеров, в которых запущен StarVault.
    - `tag` (`string: "1.2.0"`) — Тег образа Docker для контейнеров, в которых запущен StarVault. При работе в боевой среде должен быть привязан к конкретной версии. В противном случае другие изменения, внесенные в чарт, могут непроизвольно привести к обновлению контроллера допуска
    - `pullPolicy` (`string: "IfNotPresent"`) — Политика извлечения для образов контейнеров. По умолчанию политика извлечения `IfNotPresent` — Kubelet пропускает шаг извлечения образа, если тот уже существует.
- `updateStrategyType` (`string: "OnDelete"`) — параметр задает тип стратегии обновления для `StatefulSet`.
- `logLevel` (`string: ""`) — параметр задает уровень детализации для логирования сервера StarVault. Такая настройка переопределит значения, заданные в файле конфигурации StarVault. Поддерживаются следующие уровни журнала: `trace`, `debug`, `info`, `warn`, `error`.
- `logFormat` (`string: ""`) — параметр задает формат логирования сервера StarVault. Такая настройка переопределит значения, заданные в файле конфигурации StarVault. Поддерживаются следующие форматы журнала: standard, json.
- `resources` (`dictionary: {}`) — Запросы и лимиты ресурсов (ЦП, ОЗУ, пр.) для каждого контейнера на сервере. Это должен быть словарь на языке YAML для имеющегося в Kubernetes объекта resource. Если параметр не задан, то поды не будут запрашивать какой-то конкретный объем ресурсов, что ограничивает способность Kubernetes эффективно использовать вычислительные ресурсы. Настоятельно рекомендуем задать эти параметры.

```yaml
resources:
  requests:
    memory: '10Gi'
  limits:
    memory: '10Gi'
```

- `ingress` — значения, которые настраивают сервисы Ingress для StarVault. При развертывании на OpenShift эти настройки `ingress` игнорируются. Используйте конфигурацию `route`, чтобы объявить StarVault на OpenShift.
    - `enabled` (`boolean: false`) — Установка параметра в значение true создаст сервис Ingress.

- `labels` (`dictionary: {}`) — Метки для сервиса Ingress.
- `annotations` (`dictionary: {}`) — это значение задает дополнительные аннотации, которые нужно добавить к сервису Ingress. Это может быть либо YAML, либо многострочная шаблонная цепочка символов в формате YAML.

```yaml
annotations:
  kubernetes.io/ingress.class: nginx
  kubernetes.io/tls-acme: "true"
# или
annotations: |
  kubernetes.io/ingress.class: nginx
  kubernetes.io/tls-acme: "true"
```

  - `ingressClassName` (`string: ""`) — укажите IngressClass, который следует использовать при внедрении Ingress
  - `activeService` (`boolean: true`) — когда включен режим высокой доступности и используется регистрация в сервисе K8s, этот параметр настраивает сервис Ingress так, чтобы тот указывал на активный сервис StarVault.
  - `extraPaths` (`array: []`) — параметр задает дополнительные пути, которые добавляются в начало конфигурации хоста. Это полезно при работе с сервисами, в основе которых лежат аннотации.

```yaml
extraPaths:
  - path: /*
    backend:
      service:
        name: ssl-redirect
        port:
          number: use-annotation
```

  - `tls` (`array: []`) — параметр настраивает ту часть в Ingress spec, которая относится к TLS. `hosts` - это список хостов, указанных в поле Common Name сертификата TLS. `secretName` - это имя Секрета, в котором содержатся необходимые файлы TLS, например, сертификаты и ключи.

```yaml
tls:
  - hosts:
      - sslexample.foo.com
      - sslexample.bar.com
    secretName: testsecret-tls
```

  - `hosts` — значения, которые задают правила хоста Ingress.
    - `host` (`string: "chart-example.local"`): Имя хоста, который используется для Ingress.

        - `paths` (`array: []`): Устарело: лучше используйте `server.ingress.extraPaths`. Список путей, которые будут отправлены сервису StarVault. Нужно указать хотя бы один путь.

```yaml
paths:
  - /
  - /starvault
```

- `route` - Значения, которые настраивают сервисы Route для StarVault в OpenShift. Если включен режим `ha`, сервис Route будет указывать на активный сервер StarVault с помощью сервиса `active`.
    - `enabled` (`boolean: false`) — установка параметра в значение `true` создаст маршрут `Route` для StarVault.
    - activeService (`boolean: true`) — когда включен режим высокой доступности и используется регистрация в сервисе K8s, этот параметр настраивает маршрут так, чтобы тот указывал на активный сервис StarVault.
    - `labels` (`dictionary: {}`) — Метки для Route
    - `annotations` (`dictionary: {}`) — Аннотации для добавления к Route. Это может быть либо YAML, либо многострочная шаблонная цепочка символов в формате YAML.
    - `host` (`string: "chart-example.local"`) — параметр задает имя хоста для Route.
    - `tls` (`dictionary: {termination: passthrough}`) — Файл конфигурации TLS будет передан напрямую в файл конфигурации TLS маршрута, который можно использовать для настройки других методов терминирования, с помощью которых TLS терминируется на маршрутизатор.
- `authDelegator` - Значения, которые настраивают объект Cluster Role Binding, привязанный к служебной учетной записи StarVault.
    - `enabled` (`boolean: true`) — установка параметра в значение true привяжет объект Cluster Role Binding к служебной учетной записи StarVault. У объекта Cluster Role Binding есть необходимые права для того, чтобы StarVault мог использовать метод аутентификации Kubernetes.
- `readinessProbe` - Значения, которые настраивают пробу готовности для подов StarVault.
    - `enabled` (`boolean: true`) — установка параметра в значение true применит пробу готовности к подам StarVault.
    - path (`string: ""`) — установка параметра в конкретное значение активирует пробы `HTTP`/`HTTPS` вместо используемой по умолчанию пробы exec. Схема `http`/`https` контролируется значением `tlsDisable`.

- `failureThreshold` (`int: 2`) — установка параметра в конкретное значение определяет, сколько неудачных проб допускает Kubernetes.
- `initialDelaySeconds` (`int: 5`) — установка параметра в конкретное значение определяет количество секунд после запуска контейнера и до инициализации пробы.
- `periodSeconds` (`int: 5`) — установка параметра в конкретное значение определяет, через какие интервалы (в секундах) брать пробу.
- `successThreshold` (`int: 1`) — установка параметра в конкретное значение определяет, через какое минимальное число удачных проб, взятых после неудачной, проба снова считается адекватной.
- `timeoutSeconds` (`int: 3`) — установка параметра в конкретное значение определяет, через сколько секунд проба считается неудачной.
- `port` (`int: 8200`) — установка параметра в конкретное значение переопределяет порт по умолчанию, в используемый для пробы готовности сервера.

```yaml
readinessProbe:
  enabled: true
  path: /v1/sys/health?standbyok=true
  failureThreshold: 2
  initialDelaySeconds: 5
  periodSeconds: 5
  successThreshold: 1
  timeoutSeconds: 3
  port: 8200
```

- `livenessProbe` - Значения, которые настраивают пробу жизнеспособности для подов StarVault.
  - `enabled` (`boolean: false`) — установка параметра в значение true применит пробу жизнеспособности к подам StarVault.
  - `path` (string: "v1/sys/health?standbyok=true") — Установка параметра в конкретное значение активирует пробы `HTTP` / `HTTPS` вместо используемой по умолчанию пробы exec. Схема http/https контролируется значением tlsDisable.
  - `initialDelaySeconds` (`int: 60`) — Задает изначальное время задержки для пробы жизнеспособности при запуске контейнера.
  - `failureThreshold` (`int: 2`) — установка параметра в конкретное значение определяет, сколько неудачных проб допускает Kubernetes.
  - `periodSeconds` (`int: 5`) — установка параметра в конкретное значение определяет, через какие интервалы (в секундах) брать пробу.

- `successThreshold` (`int: 1`) — установка параметра в конкретное значение определяет, через какое минимальное число удачных проб, взятых после неудачной, проба снова считается адекватной.
- `timeoutSeconds` (`int: 3`) — установка параметра в конкретное значение определяет, через сколько секунд проба считается неудачной.
- `port` (`int: 8200`) — установка параметра в конкретное значение переопределяет порт по умолчанию, в используемый для пробы жизнеспособности сервера.

```yaml
livenessProbe:
  enabled: true
  path: /v1/sys/health?standbyok=true
  initialDelaySeconds: 60
  failureThreshold: 2
  periodSeconds: 5
  successThreshold: 1
  timeoutSeconds: 3
  port: 8200
```

- `terminationGracePeriodSeconds` (`int: 10`) — Время в секундах, которое нужно поду для корректного завершения работы.
- `preStopSleepSeconds` (`int: 5`) — Используется, чтобы на шаге preStop задать время, через которое объект переходит в спящий режим.
- `postStart` (`array: []`) — Используется, чтобы определить, какие команды запускать после того, как под будет готов. Это можно использовать для автоматизации процессов, например, методов аутентификации при инициализации и начальном запуске.

```yaml
postStart:
  - /bin/sh
  - -c
  - /starvault/userconfig/myscript/run.sh
```

- `extraInitContainers` (`array: null`) — extraInitContainers представляет собой список контейнеров init. Указывается как список на языке YAML. Это полезно, если нужно запустить скрипт для динамического предоставления сертификатов TLS или выписывания файлов конфигурации.
- `extraContainers` (`array: null`) — Дополнительные контейнеры для применения к подам сервера StarVault.

```yaml
extraContainers:
  - name: mycontainer
```

```
    image: 'app:0.0.0'
    env: ...
```

- `extraEnvironmentVars` (`dictionary: {}`) — Дополнительные переменные окружения для применения к серверу StarVault.

```yaml
extraEnvironmentVars:
  GOOGLE_REGION: global
  GOOGLE_PROJECT: myproject
  GOOGLE_APPLICATION_CREDENTIALS:
/starvault/userconfig/myproject/myproject-creds.json
```

- `shareProcessNamespace` (`boolean: false`) — параметр обеспечивает совместный доступ к пространству имен процесса для StarVault и `extraContainers`. Это полезно, если StarVault, например, получает сигнал на отправку SIGHUP для ротации журналов.
- extraArgs (`string: null`) — Дополнительные аргументы для применения к команде запуска сервера StarVault.
- `extraArgs`: `-config=/path/to/extra/config.hcl -log-format=json`
- `extraPorts` (`array: []`) — дополнительные порты, которые добавятся в контроллер Statefulset сервера

```yaml
extraPorts:
  - containerPort: 8300
    name: http-monitoring
```

- `extraSecretEnvironmentVars` (`array: []`) — Дополнительные переменные окружения заполняющиеся из секрета и которые следует применить к серверу StarVault.
  - `envName` (`string: required`) — имя переменной окружения, которую нужно заполнить в контейнере StarVault.
  - `secretName` (`string: required`) — имя секрета Kubernetes, используемого для заполнения переменной окружения, заданной с помощью envName.
  - `secretKey` (`string: required`) — имя ключа, в котором находится запрашиваемое секретное значение в секрете Kubernetes.

```yaml
extraSecretEnvironmentVars:
  - envName: AWS_SECRET_ACCESS_KEY
    secretName: starvault
    secretKey: AWS_SECRET_ACCESS_KEY
```

- `extraVolumes` (`array: []`) — Устарело: Лучше используйте volumes. Список дополнительных томов для монтирования на серверы StarVault. Это полезно, когда

нужно ввести дополнительные данные (например, сертификаты TLS), на которые другие конфигурации смогут ссылаться по знакомому пути. Значение — список объектов. Каждый объект поддерживает следующие ключи:

- `type` (`string: required`) — тип тома, должен быть либо `configMap`, либо `secret`. С учетом регистра.
- `name` (`string: required`) — имя тома configMap или secret, который будет монтироваться. Параметр также контролирует путь монтирования. По умолчанию том будет монтироваться по пути `/starvault/userconfig/<имя>`, если только не задано значение для параметра `path`.
- `path` (`string: /starvault/userconfigs`) — имя пути, по которому монтируется том `configMap` или `secret`. Если значение параметра не задано, том будет монтироваться по пути `/starvault/userconfig/<имя тома>`.
- `defaultMode` (`string: "420"`) — Режим по умолчанию для смонтированных файлов.

```yaml
extraVolumes:
  - type: 'secret'
    name: 'starvault-certs'
    path: '/etc/pki'
```

- `volumes` (`array: null`) — Список томов, доступных всем контейнерам. Параметр берет стандартные определения томов Kubernetes.

```yaml
volumes:
  - name: plugins
    emptyDir: {}
```

- `volumeMounts` (`array: null`) — Список доступных всем контейнерам точек монтирования томов. Параметр берет стандартные определения томов Kubernetes.

```yaml
volumeMounts:
  - mountPath: /usr/local/libexec/starvault
    name: plugins
    readOnly: true
```

- `affinity` - Это значение задает совместное существование (`affinity`) для подов сервера. Это должна быть либо многострочная цепочка символов, либо YAML, сопоставленный с полем `affinity` в PodSpec. По умолчанию допускается только один под на каждом узле, что минимизирует риск потери работоспособности кластера в случае отказа узла. Если нужно запустить больше подов на каждом узле (например, для тестирования на Minikube), установите этот параметр в значение `null`.

```yaml
affinity: |
  podAntiAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
    - labelSelector:
      matchLabels:
        app.kubernetes.io/name: {{ template "starvault.name" . }}
        app.kubernetes.io/instance: "{{ .Release.Name }}"
        component: server
      topologyKey: kubernetes.io/hostname
```

- `topologySpreadConstraints` (`array: []`) — Настройки топологии для подов сервера. Это может быть либо YAML, либо многострочная шаблонная цепочка символов в формате YAML.
- `tolerations` (`array: []`) — это значение задает массивы `Toleration`, приемлемые при создании графика. Это должна быть либо многострочная цепочка символов, либо YAML, сопоставленный с массивом `Toleration` в `PodSpec`.

```yaml
tolerations: |
  - key: 'node.kubernetes.io/unreachable'
    operator: 'Exists'
    effect: 'NoExecute'
    tolerationSeconds: 6000
```

- `nodeSelector` (`dictionary: {}`) — это значение задает дополнительные критерии отбора узлов, чтобы лучше контролировать место развертывания серверов StarVault. Значение параметра имеет формат либо многострочной цепочки символов, либо сопоставления YAML.

```yaml
nodeSelector: |
  disktype: ssd
```

- `networkPolicy` — значения, которые задают политику StarVault Network Policy.
  - `enabled` (`boolean: false`) — установка параметра в значение true активирует политику Network Policy для кластера StarVault.
  - `egress` (`array: []`) — это значение задает правила сетевой политики `egress`.

```yaml
egress:
  - to:
      - ipBlock:
          cidr: 10.0.0.0/24
    ports:
      - protocol: TCP
        port: 8200
```

- `priorityClassName` (`string: ""`) — класс приоритета для подов сервера
- `extraLabels` (`dictionary: {}`) — это значение задает дополнительные метки для подов сервера.

```yaml
extraLabels:
  'sample/label1': 'foo'
  'sample/label2': 'bar'
```

- `annotations` (`dictionary: {}`) — это значение задает дополнительные аннотации для подов сервера. Это может быть либо YAML, либо многострочная шаблонная цепочка символов в формате YAML.

```yaml
annotations:
  "sample/annotation1": "foo"
  "sample/annotation2": "bar"
# или
annotations: |
  "sample/annotation1": "foo"
  "sample/annotation2": "bar"
```

- `service` — значения для настройки сервиса Kubernetes, созданного для StarVault. Эти параметры также используются для сервисов `active` и `standby`, когда включен режим `ha`.
  - `enabled` (`boolean: true`) — установка параметра в значение true создаст сервис Kubernetes для StarVault.
  - `active` — значения, которые применяются только к сервису `starvault-active`.
    - `enabled` (`boolean: true`) — установка параметра в значение true создаст сервис Kubernetes `starvault-active` для StarVault с выбором подов, которые маркируют себя как ведущие узлы кластера с помощью параметра `starvault-active: true`.
  - `standby` — значения, которые применяются только к сервису `starvault-standby`.
    - `enabled` (`boolean: true`) — установка параметра в значение true создаст сервис Kubernetes `starvault-standby` для StarVault с выбором подов, которые маркируют себя как ведомые узлы кластера с помощью параметра `starvault-active: false`.
  - `clusterIP` (`string`) — `clusterIP` контролирует, назначен ли IP-адрес (кластера) сервису StarVault внутри Kubernetes. По умолчанию IP-адрес кластера для сервиса StarVault будет установлен в значение None и, следовательно, будет отключен. Если IP-адрес отключен, Kubernetes создаст

headless-сервис. Headless-сервисы можно использовать, чтобы связываться с подами напрямую через DNS, не задействуя балансировщик нагрузки.

- `type` (`string: "ClusterIP"`) — параметр задает тип создаваемого сервиса, например, `NodePort`.
- `externalTrafficPolicy` (`string: "Cluster"`) — параметр `externalTrafficPolicy` можно установить в значение `Cluster` или `Local` и он действителен только для типов сервисов `LoadBalancer` и `NodePort`.
- `port` (`int: 8200`) — порт, который прослушивается сервером StarVault внутри пода.
- `targetPort` (`int: 8200`) — порт, который прослушивается сервисом.
- `nodePort` (`int:`) — когда тип установлен в значение NodePort, привязанный к узлу порт можно настроить, используя это значение. Если значение не выбрано, будет назначен произвольный порт.
- `activeNodePort` (`int:`) — (при включенном режиме высокой доступности) если тип установлен в значение `NodePort`, то для сервиса `active` можно задать конкретное значение `nodePort`, а если не задан, то будет выбрано произвольное значение.
- standbyNodePort (`int:`) — (при включенном режиме высокой доступности) если тип установлен в значение `NodePort`, то для сервиса standby можно задать конкретное значение `nodePort`, а если не задан, то будет выбрано произвольное значение.
- `publishNotReadyAddresses` (`boolean: true`) — если значение true, то не ждите, пока поды будут готовы, а сразу добавляйте их в таргеты сервиса. Не применяется для headless-сервиса, который используется для взаимодействия внутри кластера.
- `instanceSelector`
  - `enabled` (`boolean: true`) — при установке параметра в значение false селектор сервисов, используемый для сервисов `starvault`, `starvault-active` и `starvault-standby`, не будет фильтровать по `app.kubernetes.io/instance`. Это значит, что сервисы могут выбрать поды за пределами данной системы, развернутой из Helm-чарта. Это не влияет на headless-сервис `starvault-internal` с ClusterIP: None.
- `annotations` (`dictionary: {}`) — это значение задает дополнительные аннотации для сервиса. Это может быть либо YAML, либо многострочная шаблонная цепочка символов в формате YAML.

```yaml
annotations:
  "sample/annotation1": "foo"
  "sample/annotation2": "bar"
```

```
# или
annotations: |
  "sample/annotation1": "foo"
  "sample/annotation2": "bar"
```

- `serviceAccount` — значения для настройки служебной учетной записи Kubernetes, созданной для StarVault.
  - `create` (`boolean: true`): если значение `true`, создается служебная учетная запись, которую использует StarVault.
  - `name` (`string: ""`): имя служебной учетной записи, которая будет использоваться. Если значение этого параметра не задано, а параметр «create» установлен в значение `true`, то имя генерируется с использованием имени установки (по умолчанию это `starvault`).
  - `annotations` (`dictionary: {}`) — это значение задает дополнительные аннотации для служебной учетной записи. Это может быть либо YAML, либо многострочная шаблонная цепочка символов в формате YAML.

    YAML

    ```
    annotations:
      "sample/annotation1": "foo"
      "sample/annotation2": "bar"
    # или
    annotations: |
      "sample/annotation1": "foo"
      "sample/annotation2": "bar"
    ```

  - `extraLabels` (`dictionary: {}`) — это значение задает дополнительные метки для служебной учетной записи сервера StarVault.

    YAML

    ```
    extraLabels:
      'sample/label1': 'foo'
      'sample/label2': 'bar'
    ```

  - `serviceDiscovery` — значения задают разрешения, которые нужны серверу StarVault, чтобы автоматически обнаруживать кластер StarVault и присоединяться к нему с помощью метаданных пода.
    - `enabled` (`boolean: true`) — параметр включает или отключает привязку роли служебной учетной записи к разрешениям, которые нужны StarVault для настройки регистрации сервиса Kubernetes.
- `dataStorage` — параметр задает том, где будут храниться данные StarVault, когда не используется внешнее хранилище, например Consul.
  - `enabled` (`boolean: true`) — параметр разрешает создание постоянного тома для хранения данных StarVault в ситуациях, когда не используется внешний сервис хранения.

- `size` (`string: 10Gi`) — размер тома, создаваемого для хранения данных StarVault в ситуациях, когда не используется внешний сервис хранения.
- `storageClass` (`string: null`) — имя класса хранилища, которое будет использоваться при создании тома хранения данных.
- `mountPath` (`string: /starvault/data`) — параметр задает путь в поде StarVault, где будет монтироваться хранилище данных.
- `accessMode` (`string: ReadWriteOnce`) — тип доступа к устройству хранения.
- `annotations` (`dictionary: {}`) — это значение задает дополнительные аннотации, которые нужно добавить к запросу на постоянный том (PVC) для хранения данных. Это может быть либо YAML, либо многострочная шаблонная цепочка символов в формате YAML.

```yaml
annotations:
  kubernetes.io/my-pvc: foobar
# или
annotations: |
  kubernetes.io/my-pvc: foobar
```

- `auditStorage` — параметр задает том, где будут храниться журналы аудита StarVault.
  - `enabled` (`boolean: false`) — параметр разрешает создание постоянного тома для хранения журналов аудита StarVault.
  - `size` (`string: 10Gi`) — размер тома, создаваемый для журналов аудита StarVault.
  - `storageClass` (`string: null`) — имя класса хранилища, которое будет использоваться при создании тома хранения журналов аудита.
  - `mountPath` (`string: /starvault/audit`) — параметр задает путь в поде StarVault, где будет монтироваться хранилище журналов аудита.
  - `accessMode` (`string: ReadWriteOnce`) — тип доступа к устройству хранения.
  - `annotations` (`dictionary: {}`) — это значение задает дополнительные аннотации, которые нужно добавить к запросу на постоянный том (PVC) для хранения журналов аудита. Это может быть либо YAML, либо многострочная шаблонная цепочка символов в формате YAML.

```yaml
annotations:
  kubernetes.io/my-pvc: foobar
# или
annotations: |
  kubernetes.io/my-pvc: foobar
```

- `dev` — параметр задает режим dev для кластера StarVault.

- `enabled` (`boolean: false`) — параметр включает режим `dev` для кластера StarVault. Этот режим полезен, чтобы поэкспериментировать с StarVault без необходимости распечатывания.
- `devRootToken` (`string: "root"`) — параметр задает корневой токен для сервера врежиме разработки StarVault.

  > Никогда и ни при каких обстоятельствах не запускайте сервер из режима `dev` в боевой среде. Это небезопасно, и каждый перезапуск будет приводить к потере данных (так как они хранятся в оперативной памяти). Этот режим предназначен только для разработки и экспериментирования.

- `standalone` — параметр задает режим standalone для сервера StarVault.
  - `enabled` (`boolean: true`) — параметр включает режим standalone для сервера StarVault. Этот режим использует бэкенд хранения file, и ему нужен том для постоянного хранения (dataStorage).
  - `config` (`string or object: "{}"`) — неформатированная строка для дополнительного конфигурирования HCL или JSON для серверов StarVault. Строка будет сохранена «как есть» в `ConfigMap`, читаемый серверами StarVault. Так можно добавлять дополнительные настройки, которые чарт не выдал напрямую. Объект будет записан как JSON.

    ```yaml
    config: |
      api_addr = "http://POD_IP:8200"

      listener "tcp" {
        tls_disable = 1
        address     = "0.0.0.0:8200"
      }

      storage "file" {
        path = "/starvault/data"
      }
    ```

    Это также можно задать с помощью имеющегося в Helm флага `--set` (starvault-helm версии 0.1.0 и выше), используя следующий синтаксис:

    ```bash
    --set server.standalone.config='{ listener "tcp" { address = "0.0.0.0:8200" }'
    ```

- `ha` — параметр настраивает режим `ha` для сервера StarVault.
  - `enabled` (`boolean: false`) — параметр включает режим `ha` для сервера StarVault. В этом режиме для хранения данных StarVault используется бэкенд хранения высокой доступности (например, Consul). По умолчанию настраивается так, чтобы использовать Consul Helm.

- `apiAddr` : (`string: "{}"`) — параметр задает конфигурацию адресов API для кластера StarVault. Если задана пустая строка, то используется IP-адрес пода.
- `clusterAddr` (`string: null`) — задает конфигурацию `cluster_addr` для StarVault HA. Если значение равно `0` , то по умолчанию `https://$(HOSTNAME).{{ template "starvault.fullname" . }}-internal:8201` .
- `raft` — параметр настраивает режим встроенного хранилища raft для сервера StarVault.
  - `enabled` (`boolean: false`) — параметр включает режим встроенного хранилища raft для сервера StarVault. Этот режим использует постоянные тома для хранения.
  - `setNodeId` (`boolean: false`) — параметр присваивает имени пода значение, взятое из Node Raft ID.
    - `config` (`string or object: "{}"`) — Неформатированная строка для дополнительного конфигурирования HCL или JSON для серверов StarVault. Строка будет сохранена «как есть» в `ConfigMap` , читаемый серверами StarVault. Так можно добавлять дополнительные настройки, которые чарт не выдал напрямую. Объект будет записан как JSON.
- `replicas` (`int: 3`) — количество подов, которое надо развернуть для создания высокодоступного кластера из серверов StarVault.
- `updatePartition` (`int: 0`) — если задан параметр updatePartition, то все поды с порядковым номером, который больше или равен разделу (`partition`), будут обновлены, как только в контроллере StatefulSet обновится параметр `.spec.template` . Установка параметра в значение `0` отключает все обновления разделов.
- `config` (`string or object: "{}"`) — неформатированная строка для дополнительного конфигурирования HCL или JSON для серверов StarVault. Строка будет сохранена «как есть» в том `ConfigMap` , читаемый серверами StarVault. Так можно добавлять дополнительные настройки, которые чарт не выдал напрямую. Объект будет записан как JSON.

YAML

```yaml
config: |
  ui = true
  api_addr = "http://POD_IP:8200"
  listener "tcp" {
      tls_disable = 1
      address     = "0.0.0.0:8200"
  }

  storage "consul" {
      path = "starvault/"
```

```
      address = "HOST_IP:8500"
  }
```

Также можно задать с помощью имеющегося в Helm флага `--set` (starvault-helm версии 0.1.0 и выше), используя следующий синтаксис:

```bash
--set server.ha.config='{ listener "tcp" { address = "0.0.0.0:8200"
}'
```

- `disruptionBudget` — значения, которые задают политику в отношении допустимого количества неработающих подов (disruption budget policy).
  - `enabled` (`boolean: true`) — активирует политику в отношении допустимого количества неработающих подов, чтобы ограничить количество подов, которые могут быть целенаправленно одновременно отключены.
  - `maxUnavailable` (`int: null`) — максимальное количество недоступных подов. По умолчанию это количество будет автоматически подсчитано, исходя из того, что значение параметра server.replicas равно `(n/2)-1`. Если нужно установить этот параметр в `0`, в команду установки helm-чарта следует добавить флаг `--set server.disruptionBudget.maxUnavailable=0`, что вызвано ограничением в языке Helm-шаблонизации.
- statefulSet — параметр задает настройки для StarVault Statefulset.
  - `annotations` (`dictionary: {}`) — это значение задает дополнительные аннотации, которые нужно добавить к StarVault statefulset. Это может быть либо YAML, либо многострочная шаблонная цепочка символов в формате YAML.

```yaml
annotations:
  kubernetes.io/my-statefulset: foobar
# или
annotations: |
  kubernetes.io/my-statefulset: foobar
```

- `securityContext` — параметр задает контексты безопасности для пода и контейнера
  - `pod` (`dictionary: {}`) — параметр задает `securityContext` для подов сервера на языке YAML либо в виде многострочной шаблонной цепочки символов в формате YAML.
    Если не задан и `global.openshift=false`, то по умолчанию берутся следующие значения:

```yaml
runAsNonRoot: true
runAsGroup: {{ .Values.server.gid | default 1000 }}
```

```
runAsUser: {{ .Values.server.uid | default 100 }}
fsGroup: {{ .Values.server.gid | default 1000 }}
```

Если не задан и `global.openshift=true`, то по умолчанию берутся пустые значения.

- `container` (`dictionary: {}`) — параметр задает `securityContext` для контейнеров сервера на языке YAML либо в виде многострочной шаблонной цепочки символов в формате YAML.
  Если не задан и `global.openshift=false`, то по умолчанию берется значение:

```yaml
allowPrivilegeEscalation: false
```

Если не задан и `global.openshift=true`, то по умолчанию берутся пустые значения.

3. Блок `ui`

- `ui` - Значения, которые настраивают пользовательский интерфейс StarVault.
  - `enabled` (`boolean: false`) — если значение = true, то пользовательский интерфейс будет включен. Пользовательский интерфейс будет включен только на серверах StarVault. Если параметр `server.enabled` установлен в значение `false`, то эта настройка ни на что не влияет. Чтобы так или иначе представить пользовательский интерфейс, следует задать параметр `ui.service`.
  - `serviceType` (`string: ClusterIP`) — тип регистрируемого сервиса. По умолчанию это ClusterIP. Доступные типы сервисов приведены на веб-сайте Kubernetes.
  - `publishNotReadyAddresses` (`boolean: true`) — установка параметра в значение true маршрутизирует трафик на поды StarVault, которые еще не готовы (например, запечатаны или не инициализированы).
  - `activeVaultPodOnly` (`boolean: false`) — если параметр установлен в значение true, сервис пользовательского интерфейса будет маршрутизировать трафик только на активный под в кластере StarVault HA.
  - `serviceNodePort` (`int: null`) — задает значение порта узла Node Port при использовании serviceType: NodePort на сервисе StarVault UI.
  - `externalPort` (`int: 8200`) — параметр задает внешний порт сервиса.
  - `targetPort` (`int: 8200`) — параметр задает целевой порт сервиса.
  - `externalTrafficPolicy` (`string: "Cluster"`) — параметр `externalTrafficPolicy` устанавливается в значение `Cluster` или `Local`, и он будет действителен только для типов сервисов `LoadBalancer` и `NodePort`.

- `loadBalancerSourceRanges` (`array`) — это значение задает дополнительные CIDR-методы источника при использовании `serviceType: LoadBalancer`.

```yaml
loadBalancerSourceRanges:
  - 10.0.0.0/16
  - 120.78.23.3/32
```

- `loadBalancerIP` (`string`) — это значение задает IP-адрес балансировщика нагрузки при использовании параметра `serviceType: LoadBalancer`.
- `annotations` (`dictionary: {}`) — это значение задает дополнительные аннотации для сервиса `UI`. Это может быть либо YAML, либо многострочная шаблонная цепочка символов в формате YAML.

```yaml
annotations:
  "sample/annotation1": "foo"
  "sample/annotation2": "bar"
# или
annotations: |
  "sample/annotation1": "foo"
  "sample/annotation2": "bar"
```

4. Блок `serverTelemetry`

- `serverTelemetry` — значения для настройки метрик и телеметрии. Для включения этих функций задается строка конфигурации `telemetry {}` в конфигурации StarVault. В настоящий момент этот чарт не поддерживает аутентификацию на конечной точке метрик StarVault, поэтому в строку `listener "tcp" {}` в конфигурации StarVault нужно добавить следующий блок `telemetry {}`:

```yaml
listener "tcp" {
  tls_disable = 1
  address     = "0.0.0.0:8200"

  telemetry {
    unauthenticated_metrics_access = "true"
  }
}
```

Кроме того, верхнеуровневая строка конфигурации `telemetry {}` также должна быть включена в конфигурацию StarVault следующим образом:

```yaml
telemetry {
  prometheus_retention_time = "30s",
```

```
  disable_hostname = true
}
```

- `serviceMonitor` — значения, которые настраивают мониторинг сервера StarVault.
  - `enabled` (`boolean: false`) — установка параметра в значение true активирует развертывание ServiceMonitor CustomResource на сервере StarVault. Оператор Prometheus должен быть установлен до того, как эта функция будет включена. В противном случае не получится установить чарт из-за отсутствия ресурса `CustomResourceDefinitions`, который предоставляет оператор.
  - `selectors` (`dictionary: {}`) — метки селектора для добавления к ServiceMonitor.
  - `interval` (`string: "30s"`) — интервал, через который Prometheus собирает метрики.
  - `scrapeTimeout` (`string: "10s"`) — таймаут для процедур сбора с помощью Prometheus.
- `prometheusRules` — значения, которые задают правила Prometheus.
  - `enabled` (`boolean: false`) — параметр развертывает пользовательский ресурс `PrometheusRule` для оповещений через `AlertManager`. Требуется надлежащим образом развернутый `AlertManager`.
  - `selectors` (`dictionary: {}`) — метки селектора для добавления к правилам Prometheus.
  - `rules` (`array: []`) — правила Prometheus, которые надо создать.

    Например:

    YAML

    ```yaml
    rules:
      - alert: starvault-HighResponseTime
        annotations:
          message: The response time of StarVault is over 500ms on
    average over the last 5 minutes.
        expr: starvault_core_handle_request{quantile="0.5",
    namespace="mynamespace"} > 500
        for: 5m
        labels:
          severity: warning
      - alert: starvault-HighResponseTime
        annotations:
          message: The response time of StarVault is over 1s on average
    over the last 5 minutes.
        expr: starvault_core_handle_request{quantile="0.5",
    namespace="mynamespace"} > 1000
        for: 5m
    ```

```
    labels:
      severity: critical
```

2025 orionsoft. Все права защищены.

Общие сведения о платформе StarVault

StarVault - это инструмент для управления конфиденциальной информацией, обеспечивающий централизованное хранение и эффективное управление доступом к таким секретам, как токены, пароли, сертификаты и ключи шифрования. Этот инструмент не только гарантирует безопасное хранение чувствительных данных, но и предоставляет возможность их строгого контроля, а также автоматизации связанных процессов, повышая тем самым уровень защиты данных в целом.

Секреты представляют собой конфиденциальные данные, такие как токены, API-ключи, пароли, ключи шифрования и сертификаты, доступ к которым должен быть строго ограничен. StarVault предлагает унифицированный интерфейс для управления этими секретами, гарантируя при этом тщательный контроль за доступом и ведение детализированного журнала аудита всех операций с ними.

1. Общий принцип работы StarVault

StarVault применяет токенную модель аутентификации, при которой каждый токен связан с политикой, назначенной клиенту. Политики формируются на основе доступа к определенным путям в системе, ограничивая или разрешая клиенту выполнять конкретные действия в рамках этих путей. В StarVault существует возможность как ручного создания токенов и их последующего распределения среди клиентов, так и автоматического получения токенов клиентами в процессе самостоятельной аутентификации в системе.

Рабочий процесс StarVault состоит из четырех этапов:

- **Аутентификация**. Процесс аутентификации в StarVault представляет собой проверку подлинности клиента, в ходе которой клиент предъявляет данные, исходя из которых StarVault определяет соответствие заявленной идентичности. По завершении аутентификации с использованием выбранного метода, система генерирует токен доступа, ассоциированный с определенной политикой.
- **Проверка**. StarVault проверяет клиент на соответствие сторонним доверенным источникам, таким как LDAP, AppRole и другим.
- **Авторизация**. В StarVault каждый клиент ассоциируется с определенной политикой безопасности, представляющей собой комплекс правил, которые регламентируют доступ клиента к API-эндпоинтам через токен StarVault. Политики обеспечивают декларативный механизм для управления доступом, позволяя явно указывать разрешенные и запрещенные действия и пути внутри системы StarVault.

- **Доступ**. StarVault предоставляет доступ к секретам, ключам и возможностям шифрования, выдавая токен на основе политик, связанных с идентификатором клиента. Затем клиент может использовать свой токен StarVault для будущих операций.

2. Преимущества использования StarVault

На современных предприятиях часто встречается практика, при которой учетные данные, такие как пароли, API-ключи и доступы, распределены по всей организации. Они могут храниться в нешифрованном виде в исходном коде приложений, файлах конфигурации и других незащищенных местах. Такой разрозненный подход к хранению учетных данных создает сложности в управлении доступом, затрудняя отслеживание полномочий и доступа сотрудников. Кроме того, наличие учетных данных в открытом виде значительно увеличивает риск нарушения безопасности, открывая возможности для злоумышленников как внутри, так и за пределами организации.

StarVault разработан с учетом указанных проблем. Платформа централизует учетные данные, обеспечивая их единое место хранения, что в свою очередь минимизирует вероятность несанкционированного раскрытия информации. Кроме базовых функций, StarVault гарантирует аутентификацию пользователей, приложений и систем, а также явное разрешение на доступ к ресурсам, дополнительно предоставляя журнал аудита, который систематически регистрирует и хранит записи о действиях клиентов.

Ключевыми особенностями StarVault являются:

- **Безопасное хранение секретов**. Все секреты хранятся в зашифрованном виде. StarVault предоставляет различные бэкенды для хранения данных, обеспечивая гибкость в выборе подходящего хранилища.
- **Динамическое управление секретами**. StarVault способен генерировать временные секреты по запросу для различных сервисов, таких как базы данных или облачные платформы, что уменьшает риски, связанные с постоянными учетными данными.
- **Управление доступом**. StarVault использует политики для детального контроля над тем, кто и как может получать доступ к секретам. Это обеспечивает принцип наименьших привилегий и упрощает аудит доступа.
- **Шифрование данных**. StarVault может шифровать и дешифровать данные без необходимости хранения их внутри самого StarVault, позволяя клиентам использовать его как сервис шифрования.
- **Автоматизация возобновления сертификатов**. StarVault может автоматизировать процесс создания и возобновления сертификатов, что упрощает управление сертификатами и их жизненным циклом.
- **Отзыв секретов**. В StarVault встроена поддержка отзыва секретов. StarVault может отзывать не только отдельные секреты, но и дерево секретов, например, все секреты,

прочитанные определенным пользователем, или все секреты определенного типа. Отзыв помогает при перевыпуске ключей, а также при блокировке систем в случае вторжения.

3. Содержание документации

Переходите в нужный раздел по быстрым ссылкам:

- Агент
  - Кэширование
    - Постоянный кэш
      - Kubernetes
  - Генерация конфигураций
  - Режим супервизора
  - Шаблоны
  - Сервис Windows
  - Совместимость версий
- Прокси
  - Прокси для API
  - Кэширование
    - Постоянное кэширование
    - Kubernetes
  - Совместимость версий

2025 orionsoft. Все права защищены.