# Руководство по интеграции драйвера YADRO CSI с системами управления контейнерами

В руководстве содержатся инструкции по настройке драйвера YADRO CSI (Controller Storage Interface) для интеграции TATLIN.UNIFIED Gen1, TATLIN.UNIFIED.SE, TATLIN.UNIFIED Gen2 (далее — СХД) с системами управления контейнерами.

## 1. Функциональные возможности драйвера YADRO CSI

| **Поддерживаемые ресурсы** | **Имя, тип ресурсов** |
|---|---|
| СХД | TATLIN.UNIFIED Gen1, TATLIN.UNIFIED Gen2, TATLIN.UNIFIED.SE |
| ОС хост-сервера | Ubuntu, РЕД ОС |
| Протокол передачи данных | iSCSI, FC, NFS |
| Тип СХД | Блочный, файловый |

Драйвер YADRO CSI — это интерфейс взаимодействия СХД и систем управления контейнерами.

| **Функция** | **Larissa 24.12** | **Kore 24.10** | **Juliet 24.08** | **Ison** |
|---|---|---|---|---|
| Создание ресурса | да | да | да | да |
| Удаление ресурса | да | да | да | да |
| Монтирование ресурса на узле кластера | да | да | да | да |
| Размонтирование ресурса на узле кластера | да | да | да | да |
| Увеличение объема ресурса | да [1] | да | да | да |

| Функция | Larissa 24.12 | Kore 24.10 | Juliet 24.08 | Ison |
|---|---|---|---|---|
| Уменьшение объема ресурса | нет | нет | нет | нет |
| Получение статистики о ресурсе | да | да | да | да |
| Поддержка групп ресурсов | нет | нет | нет | нет |
| Снапшоты [2] | да | да | да | нет |
| Репликация | нет | нет | нет | нет |
| Поддержка пулов с прямой адресацией | да | да | да | да |
| Поддержка пулов с косвенной адресацией | да | да | да | нет |
| Автоматизация развертывания с помощью Ansible | нет | нет | нет | нет |
| Поддержка проверки состояния драйвера (Healthcheck) | да | да | нет | нет |
| Поддержка режима высокой доступности (High Availability) | да | да | нет | нет |
| Поддержка протокола iSCSI | да | да | да | да |
| Поддержка протокола FC | да | да | нет | нет |
| Поддержка протокола NFS | да [3] [4] | нет | нет | нет |

# 2. Предварительные настройки

## 2.1. Настройка СХД

Перед настройкой ОС и платформ виртуализации на СХД должны быть выполнены следующие настройки:

1. Создан пул.

   > Для использования снапшотов создавайте пулы с косвенным типом адресации. Пулы с прямым типом адресации не поддерживают работу со снапшотами.

2. Заданы параметры портов ввода-вывода СХД, используемых для доступа к ресурсам:
   - статические IP-адреса портов;
   - адрес шлюза;
   - плавающий (виртуальный) IP-адрес СХД.

     > Настройка плавающего (виртуального) адреса обязательна при работе с файловыми ресурсами и опциональна при работе с блочными ресурсами.

## 2.2. Настройка многопутевого ввода-вывода на узлах кластера

Для настройки многопутевого ввода-вывода:

1. Проверьте наличие пакета для работы с `multipath`. Если пакет отсутствует, выполните команду:

   Для РЕД ОС

   ```bash
   yum -y install device-mapper-multipath
   ```

   Для Ubuntu

   ```bash
   apt-get -y install multipath-tools
   ```

2. Создайте файл `/etc/udev/rules.d/70-tatlin.rules` со следующим содержимым:

   ```
   # Set SCSI command timeout for TATLIN devices to 120 seconds
   ACTION=="add|change",
   SUBSYSTEM=="block", ENV{ID_VENDOR}=="YADRO",
       ENV{ID_MODEL}=="TATLIN", ATTR{device/timeout}="120"
   ```

3. Убедитесь, что на таргетах настроены уникальные IQN:

   ```bash
   echo "InitiatorName=`/sbin/iscsi-iname`" > /etc/iscsi/initiatorname.iscsi
   ```

4. Создайте файл `/etc/multipath/conf.d/tatlin.conf` со следующим содержимым:

Для РЕД ОС

```
devices {
    device {
        vendor "YADRO"
        product "TATLIN"
        path_grouping_policy "multibus"
        path_selector "service-time 0"
        path_checker directio
        no_path_retry 28
        max_sectors_kb 1024
    }
}
```

Для Ubuntu

```
devices {
    device {
        vendor "YADRO"
        product "TATLIN"
        path_grouping_policy "multibus"
        path_selector "service-time 0"
        path_checker directio
        detect_checker no
        no_path_retry 28
        max_sectors_kb 1024
    }
}
```

Параметр `no_path_retry` задает таймаут изменения состояния многопутевого ввода-вы вода устройства после отказа всех путей. Таймаут определяется как произведение значений `polling_interval` и `no_path_retry` (по умолчанию — 140 секунд, т.к. значение по умолчанию для `polling_interval` — 5 секунд). Таймаут отражает время автоматической реакции (failover) на отказ всех путей (например, при выходе коммутатора из строя). Эти и другие параметры могут быть изменены в соответствии с требованиями инициатора.

5. Создайте файл /etc/multipath.conf со следующим содержимым:

```
defaults {
    user_friendly_names yes
    }
blacklist {
}
```

6. Выполните команду:

```
/sbin/mpathconf --enable
```

7. Перезапустите сервис `multipathd`:

```
systemctl restart multipathd.service
```

# 3. Настройка драйвера YADRO CSI

Файлы с образом драйвера CSI и чартом Helm доступны на сервисном портале YADRO в раз деле TATLIN Satellites для соответствующей версии (дополнительные материалы для интеграции с системами мониторинга и настройки драйвера YADRO CSI).

Для настройки драйвера требуется использовать учетную запись с правами уровня admin.

Чтобы настроить драйвер YADRO CSI для интеграции с СХД:

1. Выполните предварительную настройку СХД.
2. Установите Docker на рабочую станцию согласно официальной документации.
3. Установите менеджер пакетов Helm согласно официальной документации.

Рекомендуемая версия ПО менеджера пакетов Helm — 3.15.4. Рекомендуема версия ПО кластера Kubernetes — 1.29.

## 3.1. Установка драйвера CSI

Чтобы установить драйвер CSI:

1. Разархивируйте файл с образом драйвера CSI:

```
gzip -cd csi-tatlinunified-<VERSION>.tar.gz|tar xf
```

<VERSION> — Версия TATLIN Satellites.

2. Импортируйте образ в локальный демон Docker:

```
docker image load --input csi-tatlinunified-<VERSION>.tar
```

3. Отправьте образ в реестр Docker:

```
export DOCKER_REGISTRY="<DOCKER-HOST-IP>"
docker images
```

```
docker tag <IMAGE-ID> $DOCKER_REGISTRY/csi-tatlinunified:<VERSION>
docker push $DOCKER_REGISTRY/csi-tatlinunified:<VERSION>
```

<IMAGE-ID> — Идентификатор образа.

> Идентификатор присваивается образу после импортирования в локальный демон Docker и отображается в столбце IMAGE ID.

<DOCKER-HOST-IP> — IP-адрес реестра Docker, сервиса для хранения контейнеров Docker.

Для отправки образа также можно использовать утилиту `skopeo`:

BASH

```
skopeo copy docker-archive:./csi-tatlinunified-<VERSION.>tar
docker://$DOCKER_REGISTRY/csi-tatlinunified:<VERSION>
```

4. Для управления ресурсами CSI разархивируйте чарт Helm:

BASH

```
tar xf csi-tatlinunified-<VERSION>.tar
```

5. Замените реестр Docker на внутренний реестр:

BASH

```
sed -i -e "s|DOCKER_REGISTRY|$DOCKER_REGISTRY|" csi-
tatlinunified/values.yaml
```

6. Для передачи параметров СХД в кластер на управляющем узле кластера создайте конфигурацион ный файл `/values.yaml` со следующими параметрами:

YAML

```
images:
  driver: <PATH>
controller:
  controllerCount: 1
  healthMonitor:
     enabled: false
storageList:
- name: <STORAGE-NAME>
  username: <USERNAME>
  password: <PASSWORD>
  address: <IP-ADDRESS>
  skipCertificateValidation: true
  storageClass:
  - protocolName: <PROTOCOL>
    FsType: <FILESYSTEM>
    poolName: <POOL-NAME>
    isDefault: true
    volumeExportPort: "<PORT>,<PORT>"
```

<PATH> — Путь к артефакту с драйвером CSI.
<STORAGE-NAME> — Имя конфигурации подключения к СХД. Задается пользователем. Должно быть уникально в пределах конфигурации.
<USERNAME> — Имя учетной записи пользователя СХД.
<PASSWORD> — Пароль учетной записи пользователя СХД.
<IP-ADDRESS> — IP-адрес СХД.
<PROTOCOL> — Поддерживаемый протокол передачи данных. Может содержать от одного протокола передачи дан ных.
<FILESYSTEM> — Тип файловой системы.
<POOL-NAME> — Имя пула хранения. Содержит имя, заданное при создании пула на стороне СХД.
<PORT> — Порт подключения к ресурсу. Может содержать от одного порта передачи данных.

7. Если в кластере не установлены компоненты: `crds`, `snapshot controller`, `snapshot validation webhook`, добавьте параметры в конфигурационный файл `values.yaml`:

```yaml
crds:
    enabled: true
snapshotctrl:
    enabled: true
snapshotwebhook:
    enabled: true
```

Установка компонентов crds, snapshot controller и snapshot validation webhook обязательна для корректной работы со снапшотами.

Значения в созданном конфигурационном файле `/values.yaml` являются приоритетными. Значения по умолчанию из аналогичного конфигурационного файла чарта Helm будут за менены на значения из созданного вручную файла.

8. (Опционально) Настройте проверку состояния кластера.
9. (Опционально) Настройте режим High Availability.
10. На управляющем узле кластера запустите чарт Helm:

```
helm upgrade csi-tatlinunified <PATH> -i -f ./values.yaml
```

<PATH> — Путь к артефактам чарта Helm.

## 3.2. Создание StorageClass

Ресурс StorageClass создается автоматически при запуске чарта Helm. StorageClass хранит параметры подключения к системе хранения данных. Для добавления нового ресурса или

настройки параметров создайте файл с необходимой конфигурацией.

Подробнее о ресурсе CSI StorageClass см. в официальной документации.

Чтобы создать ресурс StorageClass:

1. Создайте конфигурационный файл `<STORAGE_CLASS_FILE>.yaml` со следующими параметрами:

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <NAME>
provisioner: <DRIVER-NAME>
parameters:
  storageID: <STORAGE-NAME>
  csi.storage.k8s.io/fstype: <FILE-SYSTEM>
  poolName: <POOL-NAME>
  protocol: <PROTOCOL>
  volumeExportPort: <PORT>
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: WaitForFirstConsumer
```

   <STORAGE_CLASS_FILE> — Файл с конфигурацией StorageClass.
   <NAME> — Уникальное имя StorageClass.
   <DRIVER-NAME> — Имя драйвера CSI.
   <STORAGE-NAME> — Имя конфигурации подключения к СХД. Задается пользователем. Должно быть уникально в пределах конфигурации.
   <FILE-SYSTEM> — Тип файловой системы. Содержит тип файловой системы, которая будет смонтирована на блочном устройстве.
   <PROTOCOL> — Протокол для подключения к ресурсу.

2. Создайте ресурс с новой конфигурацией:

```bash
kubectl apply -f <STORAGE_CLASS_FILE>.yaml
```

## 3.3. Настройка Healthcheck

Чтобы настроить проверку состояния кластера, в конфигурационном файле `/values.yaml` для параметра `healthMonitor` задайте значение `true`:

```yaml
controller:
    healthMonitor:
        enabled: true
```

По умолчанию для параметра `healthMonitor` выставлено значение `false`, проверка состояния кластера отключена.

## 3.4. Настройка High Availability

Чтобы настроить режим High Availability, в конфигурационном файле `values.yaml` для параметра `controllerCount` задайте значение больше `1`:

YAML

```yaml
controller:
    controllerCount: 2
```

1. Недоступно для файловых ресурсов при доступе по протоколу NFS.
2. Поддерживается только для пулов с косвенной адресацией при работе с СХД TATLIN.UNIFIED Gen1 и TATLIN.UNIFIED Gen2 с версией ПО 3.1 и выше.
3. Поддерживается только для пулов с прямой адресацией.
4. Рекомендуется использовать на СХД TATLIN.UNIFIED Gen1 и TATLIN.UNIFIED Gen2 с версией ПО 3.1.1.

# Установка NFS-provisioner

## 1. Предварительные приготовления:

### 1.1. На рабочей станции, с которой будет проводиться установка:

- Установлен helm.
- Настроен доступ к кластеру через `kubectl`.

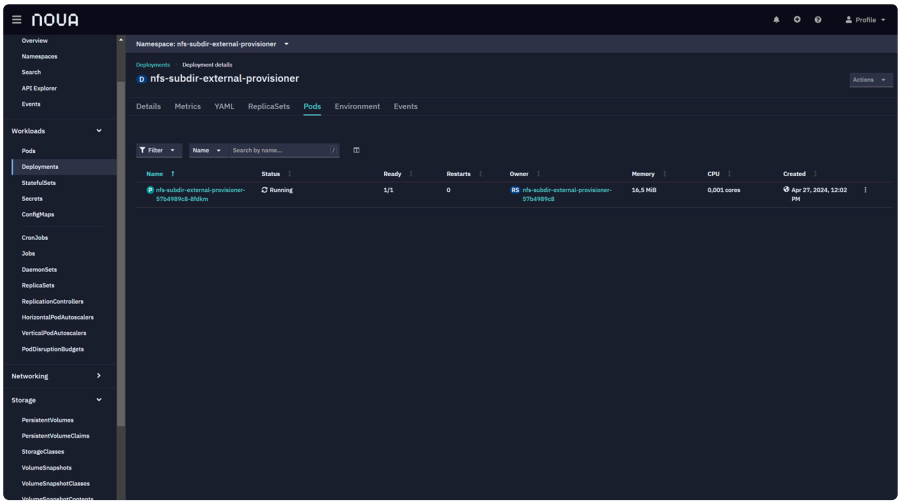
### 1.2. На NFS-сервере:

- Создана директория для экспорта.
- В файле `/etc/exports` настроены корректные права доступа к директории для каждого *worker-узла*. Пример настройки доступа:
  ```
  [root@localhost ~] cat /etc/exports
  /mnt/nfstest 10.249.120.104(rw,sync,no_subtree_check)
  10.249.120.105(rw,sync,no_subtree_check)
  ```

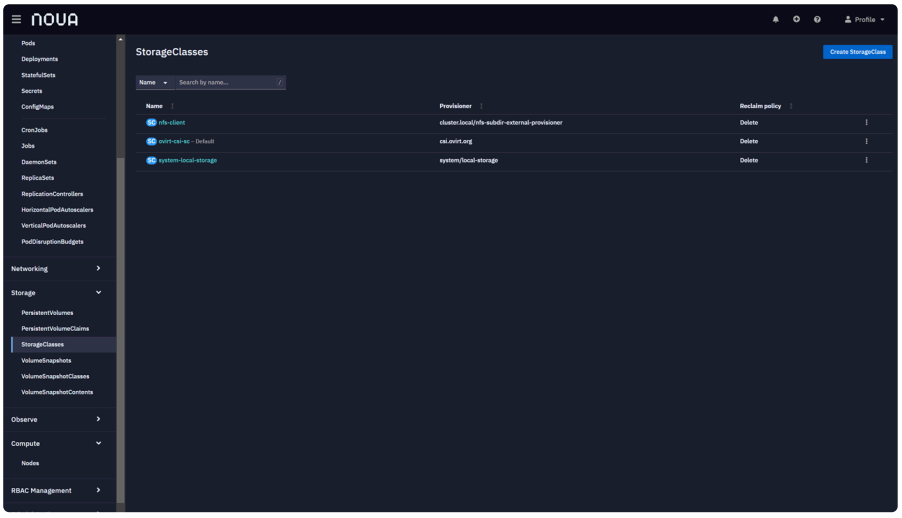
## 2. Процесс установки nfs-provisioner

1. Установите на *worker-узлы* пакеты для работы *nfs-клиента*. Имя пакета зависит от дистрибутива ОС, обычно это *nfs-utils* для RHEL-based и *nfs-common* для Debian-based.
2. На рабочей станции, с которой будет производиться установка, добавьте HELM репозиторий для `nfs-provisioner`:
   ```
   helm repo add nfs-subdir-external-provisioner https://kubernetes-sigs.github.io/nfs-subdir-external-provisioner/
   ```
3. Выполните установку чарта, заменив в команде адрес nfs-сервера (*nfs.server*) и путь к директории (*nfs.path*). Вы можете изменить дополнительные параметры установки (имя для StorageClass, reclaimPolicy и т.д.), подробнее https://github.com/kubernetes-sigs/nfs-subdir-external-provisioner/tree/master/charts/nfs-subdir-external-provisioner:
   ```
   helm install nfs-subdir-external-provisioner nfs-subdir-external-provisioner/nfs-subdir-external-provisioner --set nfs.server=10.249.120.100 --set nfs.path=/mnt/nfstest --set nodeSelector."node-role\.kubernetes\.io/worker"="" --namespace nfs-subdir-external-provisioner --create-namespace
   ```

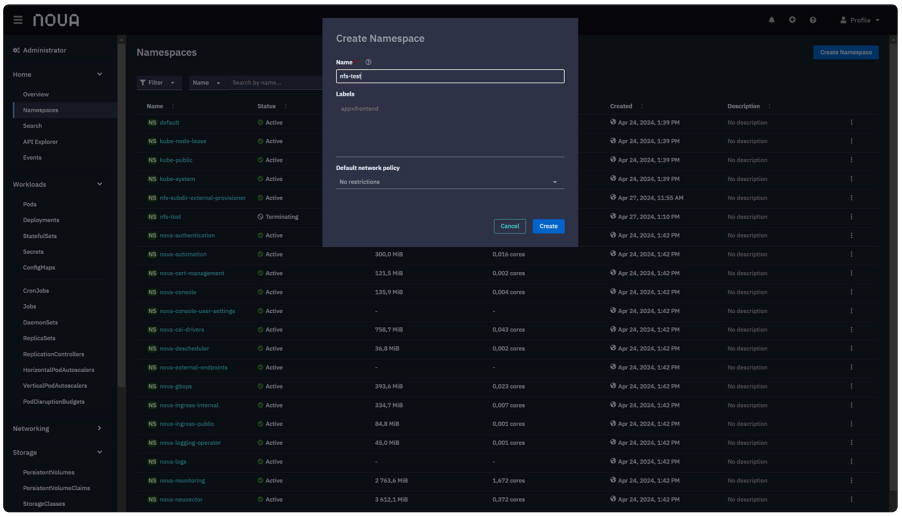
4. После завершения установки, откройте в браузере веб-интерфейс Nova Console и перейдите в раздел *Workloads* → *Deployments* и выберите `nfs-subdir-external-provisioner`.
5. Перейдите на вкладку *Pods* и убедитесь, что поды находятся в статусе *Running*:



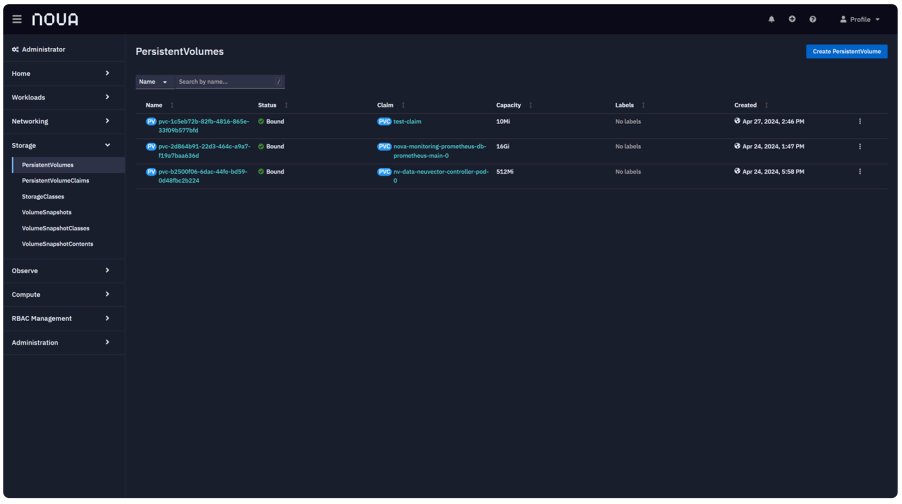

6. Перейдите на вкладку *Storage* → *StorageClasses* и убедитесь, что появился новый класс *nfs-client*:




7. Теперь вы можете протестировать работу данного *provisioner*. Для этого перейдите в раздел *Home* → *Namespaces* и нажмите *«Create Namespace»*.



8. В открывшемся окне введите имя для пространства имен, например `nfs-test` и нажмите *«Create»*:




9. Нажмите в правом-верхнем углу на кнопку *“+”* и вставьте следующий манифест для создания тестового *PVC*, после чего нажмите *«Create»*:

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: test-claim
  namespace: nfs-test
spec:
  storageClassName: nfs-client
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Mi
```

10. Перейдите на страницу *Storage* → *PersistentVolumes* и убедитесь, что был создан *PV* для *PVC* из предыдущего пункта.

---

# Обзор

Longhorn — это распределенная система хранения данных, разработанная для использования в контейнерных средах, таких как Kubernetes. Она предоставляет блоковое хранилище, которое легко интегрируется в кластеры Kubernetes и обладает рядом функций:

1. **Простота установки и управления:** Longhorn легко устанавливается в кластер Kubernetes и управляется с помощью стандартных инструментов оркестрации.
2. **Резервное копирование и восстановление:** Система поддерживает создание снапшотов и резервных копий данных, что упрощает управление данными и позволяет быстро восстанавливать их в случае сбоя.
3. **Масштабируемость:** Longhorn позволяет масштабировать хранилище в зависимости от потребностей, добавляя новые узлы.
4. **Высокодоступное хранилище:** Longhorn обеспечивает репликацию данных и устойчивость к сбоям, что позволяет сохранять данные в случае выхода узлов из строя.
5. **Интеграция с Kubernetes:** Longhorn тесно интегрирован с Kubernetes, предоставляя возможность использовать его в качестве встроенного решения для хранения данных.

Longhorn является особенно полезным в сценариях, требующих надежного и гибкого хранилища данных для контейнеризованных приложений.

> В данный момент Longhorn находится на стадии Tech Preview.

## 1. Содержание раздела