Лабораторная работа №13

Презентация

Мосолов А.Д.

29 ноября 2025

Российский университет дружбы народов, Москва, Россия

Докладчик

- Мосолов Александр Денисович
- Студент, НПИбд02-23
- Российский университет дружбы народов
- 1132236128@pfur.ru

Цель

Приобретение навыков настройки сервера NFS (Network File System) для организации удалённого доступа к файловым ресурсам в локальной сети.

Задание

1. Установить и настроить сервер NFSv4.
2. Подмонтировать удалённый ресурс на клиенте.
3. Подключить каталог с контентом веб-сервера к дереву NFS.
4. Подключить домашний каталог пользователя к дереву NFS.
5. Написать скрипты автоматизации настройки (provision) для Vagrant.

Установка nfs-utils на сервере

Первым шагом я устанавливаю необходимый пакет `nfs-utils` на сервере. Этот пакет содержит утилиты для работы как сервера, так и клиента NFS. Как видно из вывода, пакет уже был установлен в системе.

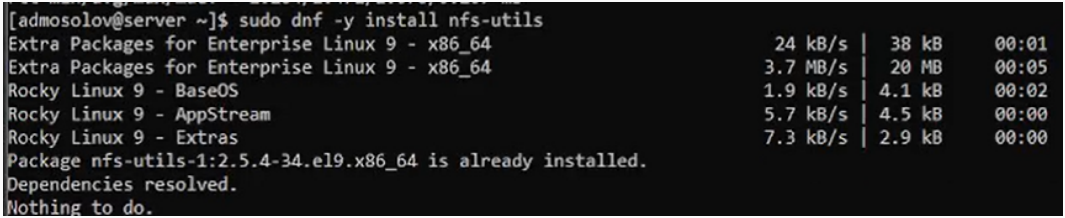


Рис. 1: Установка nfs-utils на сервере

Создание корневого каталога NFS

Далее я создаю основной каталог, который будет служить корнем для экспортируемой файловой системы NFS (`/srv/nfs`). После этого я открываю текстовый редактор для настройки конфигурационного файла экспорта.

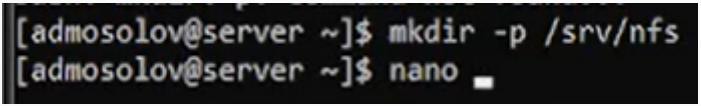


Рис. 2: Создание корневого каталога NFS

Настройка /etc/exports

В файл `/etc/exports` я добавляю запись, разрешающую доступ к каталогу `/srv/nfs` для всех хостов (`*`) в режиме только для чтения (`ro`). Это базовая настройка безопасности.

Рис. 3: Настройка /etc/exports

Настройка контекста SELinux

Для корректной работы NFS с включенным SELinux необходимо настроить контекст безопасности. Я выполняю команду `semanage fcontext` (исправив опечатку с sudo), чтобы назначить тип `nfs_t` для каталога `/srv/nfs` и его содержимого.

```
[admosolov@server ~]$  semanage fcontext -a -t nfs_t "/srv/nfs(/.*)?"
ValueError: SELinux policy is not managed or store cannot be accessed.
[admosolov@server ~]$ sudo semanage fcontext -a -t nfs_t "/srv/nfs(/.*)?"
File context for /srv/nfs(/.*)? already defined, modifying instead
```

Рис. 4: Настройка контекста SELinux

Запуск службы NFS

После определения правил SELinux я применяю их командой `restorecon`. Затем я запускаю службу NFS-сервера и добавляю её в автозагрузку, чтобы сервис поднимался после перезагрузки системы.

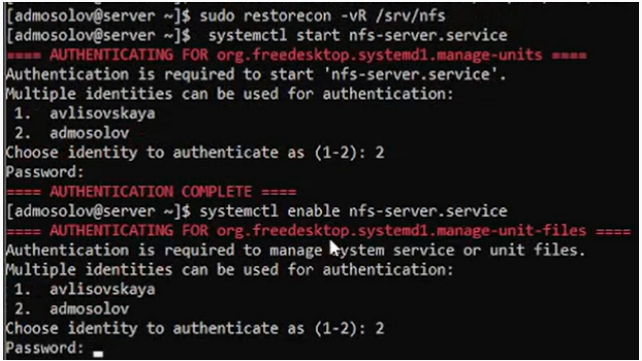


Рис. 5: Запуск службы NFS

Настройка firewall для NFS

Для доступа клиентов к серверу необходимо открыть соответствующие порты в брандмауэре. Я добавляю службу `nfs` в постоянные правила `firewall-cmd`.

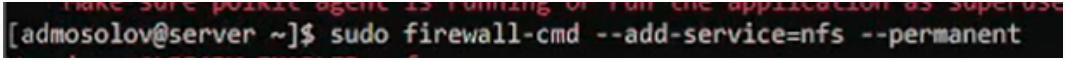


Рис. 6: Настройка firewall для NFS

Установка nfs-utils на клиенте

Перехожу к настройке клиента. На клиентской машине также необходимо установить пакет `nfs-utils`, чтобы система могла монтировать удаленные ресурсы.

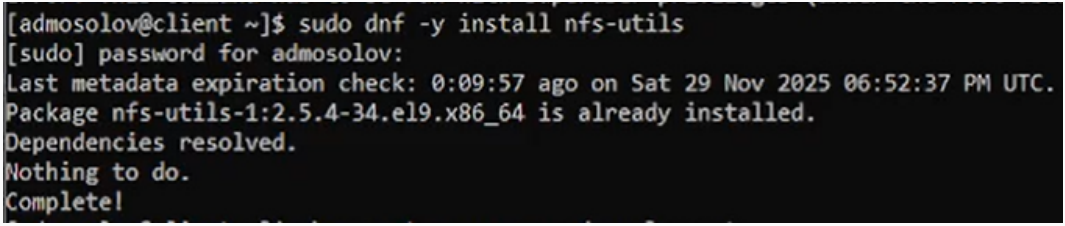


Рис. 7: Установка nfs-utils на клиенте

Проверка экспорта с клиента

С клиентской машины я проверяю доступность экспортируемых ресурсов на сервере с помощью команды `showmount -e`. Вывод показывает, что каталог `/srv/nfs` доступен.

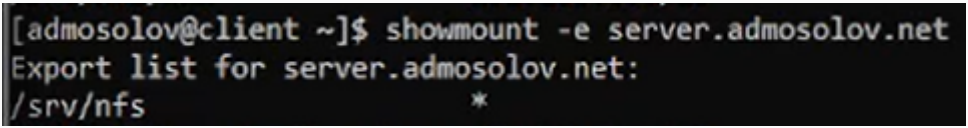


Рис. 8: Проверка экспорта с клиента

Добавление служб RPC в firewall

Возвращаюсь на сервер для донастройки брандмауэра. Поскольку NFS использует RPC, необходимо также разрешить службы `mountd` и `rpc-bind`. Сначала я проверяю открытые порты, а затем добавляю необходимые службы.

```
[admosolov@server ~]$ lsof | grep TCP
lsof: WARNING: can't stat() fuse.gvfsd-fuse file system /run/user/990/gvfs
      Output information may be incomplete.
[admosolov@server ~]$ sudo firewall-cmd --add-service=mountd --add-service=rpc-bind --permanent
Warning: ALREADY_ENABLED: mountd
Warning: ALREADY_ENABLED: rpc-bind
success

[admosolov@server ~]$ sudo firewall-cmd --reload

success
[admosolov@server ~]$
[admosolov@server ~]$ lsof | grep TCP
lsof: WARNING: can't stat() fuse.gvfsd-fuse file system /run/user/990/gvfs
      Output information may be incomplete.
[admosolov@server ~]$
```

Список разрешенных сервисов

Убеждаюсь, что все необходимые службы (nfs, mountd, rpc-bind) добавлены в список разрешенных сервисов брандмауэра.

```
[admosolov@server ~]$ sudo firewall-cmd --list-services
cockpit dhcp dhcpv6-client dns http https imap imaps mountd mysql nfs ntp pop3 pop3s rpc-bind samba smtp
 smtp-submission ssh
[admosolov@server ~]$
```

Рис. 10: Список разрешенных сервисов

Создание точки монтирования

На клиентской машине создаю директорию `/mnt/nfs`, которая будет использоваться как точка монтирования для удаленного ресурса.

Рис. 11: Создание точки монтирования

Монтирование ресурса

Выполняю монтирование удаленного каталога `/srv/nfs` с сервера в локальную директорию `/mnt/nfs`, явно указывая использование протокола NFS версии 4.

```
[admosolov@client ~]$ sudo mount -t nfs -o nfsvers=4 server.admosolov.net:/srv/nfs /mnt/nfs
[sudo] password for admosolov:
[admosolov@client ~]$ mount
```

Рис. 12: Монтирование ресурса

Проверка монтирования

Проверяю успешность операции командой `mount`. Видно, что ресурс `server.admosolov.net:/srv/nfs` успешно смонтирован. Параметр `_netdev` указывает, что это сетевое устройство.

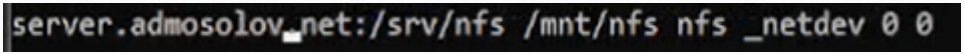


Рис. 13: Проверка монтирования

Статус remote-fs.target

Также проверяю статус системного юнита `remote-fs.target`, который отвечает за работу с удаленными файловыми системами. Он находится в активном состоянии.

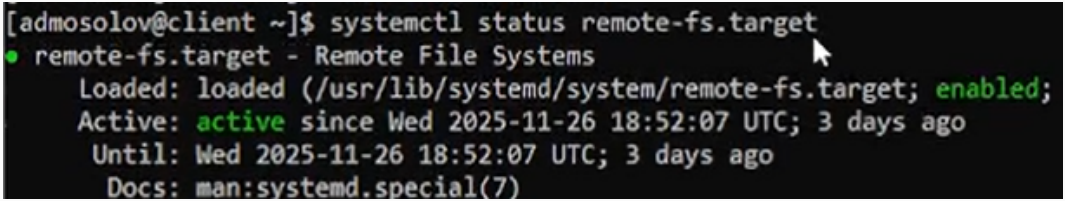


Рис. 14: Статус remote-fs.target

Создание каталога для www

Теперь я перехожу к настройке экспорта каталога веб-сервера. На сервере создаю подкаталог `/srv/nfs/www` внутри дерева NFS.

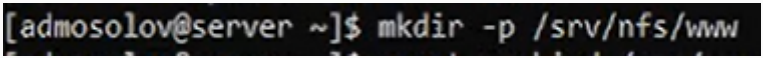


Рис. 15: Создание каталога для www

Bind-монтирование www

Монтирую реальный каталог веб-сервера `/var/www` в созданный подкаталог NFS с использованием опции `--bind`. Это позволяет “пробросить” существующую директорию в дерево экспорта.

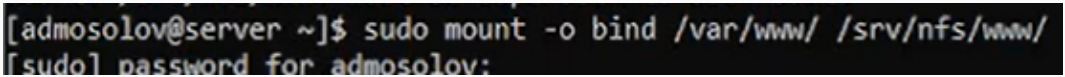


Рис. 16: Bind-монтирование www

Обновление exports для www

Редактирую файл `/etc/exports` для настройки прав доступа к новому ресурсу. Я разрешаю доступ к `/srv/nfs/www` для подсети `192.168.0.0/16` с правами на чтение и запись (`rw`).

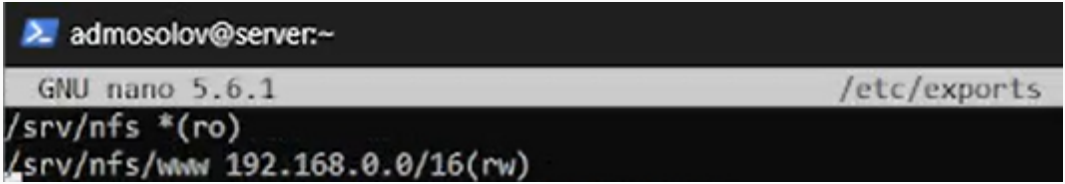


Рис. 17: Обновление exports для www

Запись bind-mount в fstab

Чтобы bind-монтирование сохранялось после перезагрузки, я добавляю соответствующую запись в файл `/etc/fstab` на сервере.

Рис. 18: Запись bind-mount в fstab

Проверка содержимого www на клиенте

На клиенте проверяю содержимое смонтированного ресурса `ls /mnt/nfs`. Теперь там виден каталог `www` (внутри которого `home` и `html`), что подтверждает успешный проброс директории.

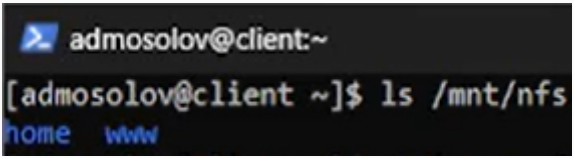


Рис. 19: Проверка содержимого www на клиенте

Подготовка каталога пользователя

Далее я настраиваю экспорт домашнего каталога пользователя. Создаю директорию `~/common`, тестовый файл в ней, а также соответствующую структуру каталогов в `/srv/nfs`, исправляя ошибки доступа при необходимости.

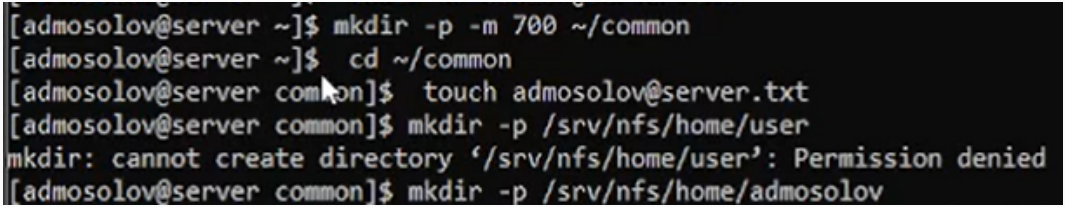


Рис. 20: Подготовка каталога пользователя

Bind-монтирование каталога пользователя

Выполняю bind-монтирование личного каталога пользователя в дерево NFS и открываю файл экспорта для настройки.

```
[admosolov@server common]$ sudo mount -o bind /home/admosolov/common /srv/nfs/home/admosolov
[admosolov@server common]$ sudo nano /etc/exports
```

Рис. 21: Bind-монтирование каталога пользователя

Обновление exports для пользователя

Добавляю в `/etc/exports` запись для `/srv/nfs/home/admosolov`, разрешая доступ на чтение и запись для локальной сети.

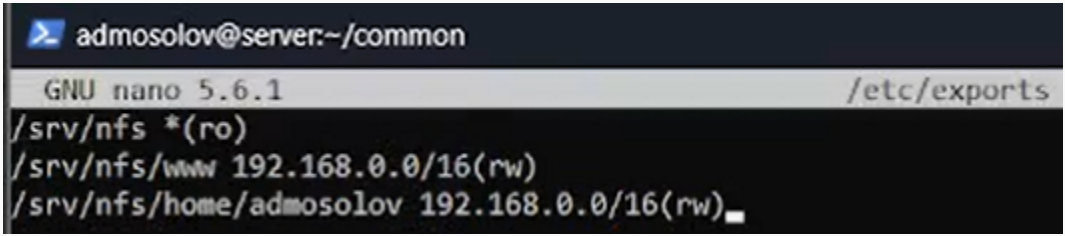


Рис. 22: Обновление exports для пользователя

Применение изменений экспорта

Применяю изменения конфигурации NFS без перезагрузки службы, используя команду `exportfs -r`.

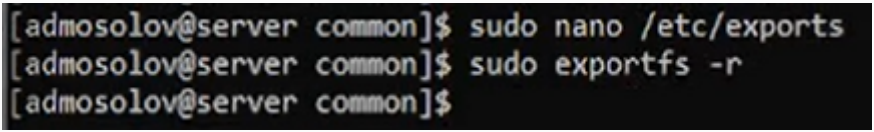


Рис. 23: Применение изменений экспорта

Создание файла на клиенте

Для проверки прав на запись перехожу на клиенте в личный каталог пользователя и создаю там тестовый файл `admosolov@client.txt`.

```
[admosolov@client common]$ cd /mnt/nfs/home/admosolov
[admosolov@client admosolov]$  touch admosolov@client.txt
[admosolov@client admosolov]$
```

Рис. 24: Создание файла на клиенте

Проверка файла на сервере

Возвращаюсь на сервер и проверяю содержимое исходного каталога `~/common`. Файл, созданный на клиенте, успешно появился, что подтверждает корректную работу rw-доступа.

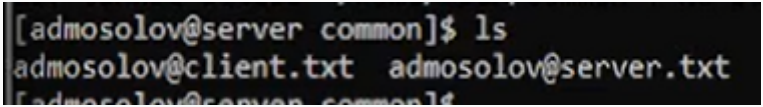


Рис. 25: Проверка файла на сервере

Подготовка скрипта сервера

Начинаю этап автоматизации. Создаю на сервере каталог для хранения конфигурационных файлов provisioning-скрипта, копирую туда текущий `/etc/exports` и создаю скрипт `nfs.sh`.

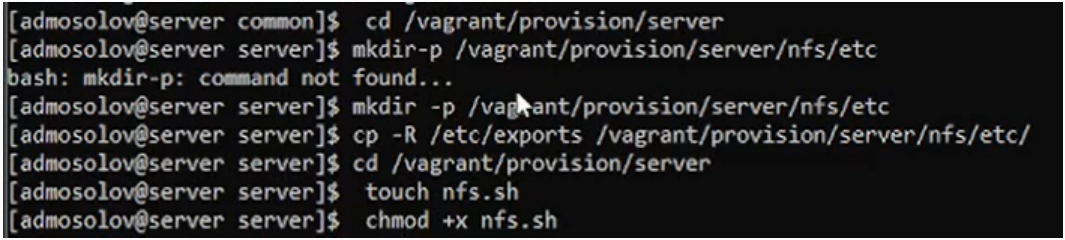
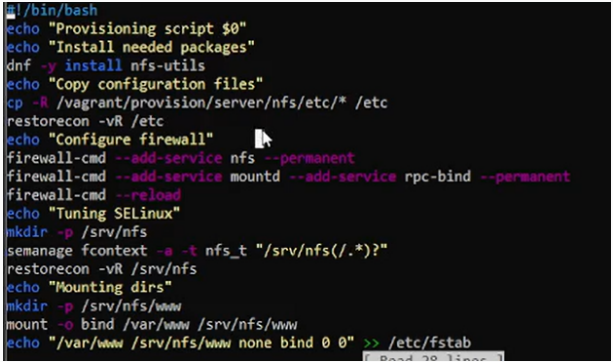


Рис. 26: Подготовка скрипта сервера

Скрипт nfs.sh для сервера

Наполняю скрипт `nfs.sh` командами, которые я выполнял вручную: установка пакетов, копирование конфигов, настройка firewall, SELinux, создание каталогов, монтирование и обновление fstab.



```
#!/bin/bash
echo "Provisioning script $0"
echo "Install needed packages"
dnf -y install nfs-utils
echo "Copy configuration files"
cp -R /vagrant/provision/server/nfs/etc/* /etc
restorecon -vR /etc
echo "Configure firewall"
firewall-cmd --add-service nfs --permanent
firewall-cmd --add-service mountd --add-service rpc-bind --permanent
firewall-cmd --reload
echo "Tuning SELinux"
mkdir -p /srv/nfs
semanage fcontext -a -t nfs_t "/srv/nfs(/.*)?"
restorecon -vR /srv/nfs
echo "Mounting dirs"
mkdir -p /srv/nfs/www
mount -o bind /var/www /srv/nfs/www
echo "/var/www /srv/nfs/www none bind 0 0" >> /etc/fstab
```

Рис. 27: Скрипт nfs.sh для сервера

Подготовка скрипта клиента

Аналогичные действия выполняю для клиента: создаю структуру каталогов в `/vagrant/provision/client` и создаю исполняемый скрипт настройки.

```
[admosolov@client common]$ cd /mnt/nfs/home/admosolov
[admosolov@client admosolov]$  touch admosolov@client.txt
[admosolov@client admosolov]$ cd /vagrant/provision/client
[admosolov@client client]$  cd /vagrant/provision/client
[admosolov@client client]$ cd /vagrant/provision/client
[admosolov@client client]$  touch nfs.sh
[admosolov@client client]$  chmod +x nfs.sh
[admosolov@client client]$
```

Рис. 28: Подготовка скрипта клиента

Скрипт nfs.sh для клиента

В скрипт клиента добавляю команды установки `nfs-utils`, создания точки монтирования, монтирования ресурса и добавления записи в `/etc/fstab` для авто-монтирования.

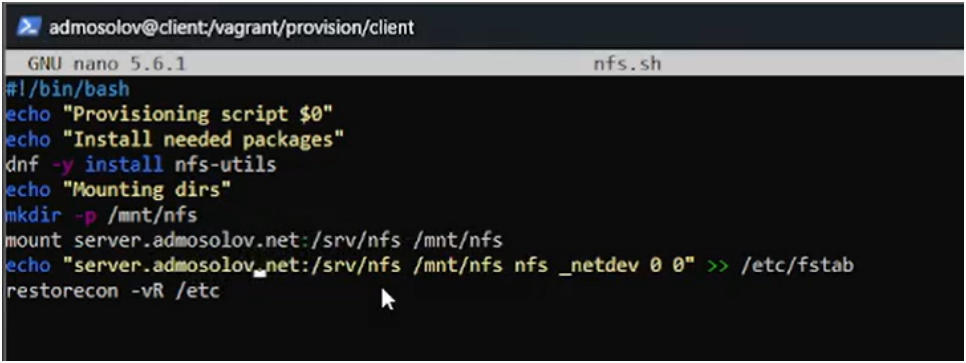


Рис. 29: Скрипт nfs.sh для клиента

Конфигурация Vagrantfile для сервера

В завершение я настраиваю `Vagrantfile`. Добавляю блок provisioning для сервера, указывая путь к созданному shell-скрипту.

```
server.vm.provision "server nfs",
                    type: "shell",
                    preserve_order: true,
                    path: "provision/server/nfs.sh"
```

Рис. 30: Конфигурация Vagrantfile для сервера

Конфигурация Vagrantfile для клиента

Также добавляю блок provisioning для клиента, чтобы он автоматически настраивался при развертывании виртуальной машины.

```
client.vm.provision "client nfs",
        type: "shell",
        preserve_order: true,
        path: "provision/client/nfs.sh"
```

Рис. 31: Конфигурация Vagrantfile для клиента

Выводы

Во время выполнения данной лабораторной работы я приобрел практические навыки настройки сетевой файловой системы NFS. Я научился конфигурировать сервер NFSv4, управлять экспортом каталогов, настраивать брандмауэр и SELinux для корректной работы сервиса. Также я освоил методы монтирования удаленных ресурсов на клиенте, использование bind-mount для проброса директорий и автоматизацию процесса настройки с помощью shell-скриптов в среде Vagrant.