

Лабораторная работа №13

Настройка NFS. Отчет

Мосолов Александр Денисович

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
4	Выводы	15
5	Ответы на контрольные вопросы	16

Список иллюстраций

3.1	Установка nfs-utils на сервере	6
3.2	Создание корневого каталога NFS	6
3.3	Настройка /etc/exports	6
3.4	Настройка контекста SELinux	7
3.5	Запуск службы NFS	7
3.6	Настройка firewall для NFS	7
3.7	Установка nfs-utils на клиенте	8
3.8	Проверка экспорта с клиента	8
3.9	Добавление служб RPC в firewall	8
3.10	Список разрешенных сервисов	9
3.11	Создание точки монтирования	9
3.12	Монтирование ресурса	9
3.13	Проверка монтирования	9
3.14	Статус remote-fs.target	9
3.15	Создание каталога для www	10
3.16	Bind-монтирование www	10
3.17	Обновление exports для www	10
3.18	Запись bind-mount в fstab	10
3.19	Проверка содержимого www на клиенте	11
3.20	Подготовка каталога пользователя	11
3.21	Bind-монтирование каталога пользователя	11
3.22	Обновление exports для пользователя	11
3.23	Применение изменений экспорта	12
3.24	Создание файла на клиенте	12
3.25	Проверка файла на сервере	12
3.26	Подготовка скрипта сервера	12
3.27	Скрипт nfs.sh для сервера	13
3.28	Подготовка скрипта клиента	13
3.29	Скрипт nfs.sh для клиента	14
3.30	Конфигурация Vagrantfile для сервера	14
3.31	Конфигурация Vagrantfile для клиента	14

1 Цель работы

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

2 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.

3 Выполнение лабораторной работы

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. 3.1).

```
[admosolov@server ~]$ sudo dnf -y install nfs-utils
Extra Packages for Enterprise Linux 9 - x86_64                24 kB/s | 38 kB    00:01
Extra Packages for Enterprise Linux 9 - x86_64                3.7 MB/s | 20 MB    00:05
Rocky Linux 9 - BaseOS                                         1.9 kB/s | 4.1 kB    00:02
Rocky Linux 9 - AppStream                                       5.7 kB/s | 4.5 kB    00:00
Rocky Linux 9 - Extras                                         7.3 kB/s | 2.9 kB    00:00
Package nfs-utils-1:2.5.4-34.el9.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
```

Рис. 3.1: Установка `nfs-utils` на сервере

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. 3.2).

```
[admosolov@server ~]$ mkdir -p /srv/nfs
[admosolov@server ~]$ nano
```

Рис. 3.2: Создание корневого каталога NFS

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. 3.3).

```
admosolov@server:~
GNU nano 5.6.1 /etc/exports
/srv/nfs *(ro)
```

Рис. 3.3: Настройка `/etc/exports`

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 (рис. 3.4).

```
[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. 3.4: Настройка контекста SELinux

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. 3.5).

```
[admosolov@server ~]$ sudo restorecon -vR /srv/nfs
[admosolov@server ~]$ systemctl start nfs-server.service
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
Authentication is required to start 'nfs-server.service'.
Multiple identities can be used for authentication:
 1. avlisovskaya
 2. admosolov
Choose identity to authenticate as (1-2): 2
Password:
==== AUTHENTICATION COMPLETE ====
[admosolov@server ~]$ systemctl enable nfs-server.service
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-unit-files ====
Authentication is required to manage system service or unit files.
Multiple identities can be used for authentication:
 1. avlisovskaya
 2. admosolov
Choose identity to authenticate as (1-2): 2
Password: █
```

Рис. 3.5: Запуск службы NFS

Для доступа клиентов к серверу необходимо открыть соответствующие порты в брандмауэре. Я добавляю службу `nfs` в постоянные правила `firewall-cmd` (рис. 3.6).

```
Take care to kill agent is running or run the application as super user
[admosolov@server ~]$ sudo firewall-cmd --add-service=nfs --permanent
```

Рис. 3.6: Настройка firewall для NFS

Перехожу к настройке клиента. На клиентской машине также необходимо

установить пакет `nfs-utils`, чтобы система могла монтировать удаленные ресурсы (рис. 3.7).

```
[admosolov@client ~]$ sudo dnf -y install nfs-utils
[sudo] password for admosolov:
Last metadata expiration check: 0:09:57 ago on Sat 29 Nov 2025 06:52:37 PM UTC.
Package nfs-utils-1:2.5.4-34.el9.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

Рис. 3.7: Установка `nfs-utils` на клиенте

С клиентской машины я проверяю доступность экспортируемых ресурсов на сервере с помощью команды `showmount -e`. Вывод показывает, что каталог `/srv/nfs` доступен (рис. 3.8).

```
[admosolov@client ~]$ showmount -e server.admosolov.net
Export list for server.admosolov.net:
/srv/nfs *
```

Рис. 3.8: Проверка экспорта с клиента

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 (рис. 3.9).

```
[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 ~]$
```

Рис. 3.9: Добавление служб RPC в firewall

Убеждаюсь, что все необходимые службы (`nfs`, `mountd`, `rpc-bind`) добавлены в список разрешенных сервисов брандмауэра (рис. 3.10).


```
[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. 3.10: Список разрешенных сервисов

На клиентской машине создаю директорию `/mnt/nfs`, которая будет использоваться как точка монтирования для удаленного ресурса (рис. 3.11).

```
[admosolov@client ~]$ mkdir -p /mnt/nfs
```

Рис. 3.11: Создание точки монтирования

Выполняю монтирование удаленного каталога `/srv/nfs` с сервера в локальную директорию `/mnt/nfs`, явно указывая использование протокола NFS версии 4 (рис. 3.12).

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

Рис. 3.12: Монтирование ресурса

Проверяю успешность операции командой `mount`. Видно, что ресурс `server.admosolov.net:/srv/nfs` успешно смонтирован. Параметр `_netdev` указывает, что это сетевое устройство (рис. 3.13).

```
server.admosolov_net:/srv/nfs /mnt/nfs nfs _netdev 0 0
```

Рис. 3.13: Проверка монтирования

Также проверяю статус системного юнита `remote-fs.target`, который отвечает за работу с удаленными файловыми системами. Он находится в активном состоянии (рис. 3.14).

```
[admosolov@client ~]$ systemctl status remote-fs.target
● remote-fs.target - Remote File Systems
   Loaded: loaded (/usr/lib/systemd/system/remote-fs.target; enabled;
   Active: active since Wed 2025-11-26 18:52:07 UTC; 3 days ago
   Until: Wed 2025-11-26 18:52:07 UTC; 3 days ago
   Docs: man:systemd.special(7)
```

Рис. 3.14: Статус `remote-fs.target`

Теперь я перехожу к настройке экспорта каталога веб-сервера. На сервере создаю подкаталог `/srv/nfs/www` внутри дерева NFS (рис. 3.15).

```
[admosolov@server ~]$ mkdir -p /srv/nfs/www
```

Рис. 3.15: Создание каталога для 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. 3.16).

```
[admosolov@server ~]$ sudo mount -o bind /var/www/ /srv/nfs/www/  
[sudo] password for admosolov:
```

Рис. 3.16: Bind-монтирование 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. 3.17).

```
admosolov@server:~  
GNU nano 5.6.1 /etc/exports  
/srv/nfs *(ro)  
/srv/nfs/www 192.168.0.0/16(rw)
```

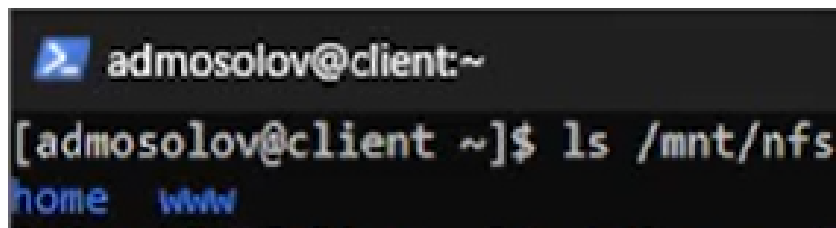
Рис. 3.17: Обновление exports для www

Чтобы bind-монтирование сохранялось после перезагрузки, я добавляю соответствующую запись в файл `/etc/fstab` на сервере (рис. 3.18).

```
/var/www /srv/nfs/www none bind 0 0
```

Рис. 3.18: Запись bind-mount в fstab

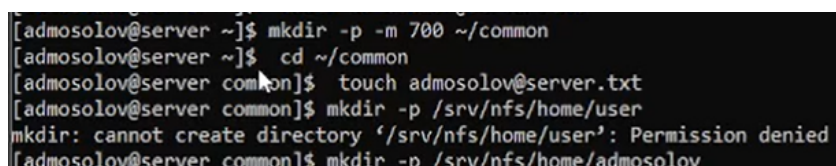
На клиенте проверяю содержимое смонтированного ресурса `ls /mnt/nfs`. Теперь там виден каталог `www` (внутри которого `home` и `html`), что подтверждает успешный проброс директории (рис. 3.19).



```
admosolov@client:~  
[admosolov@client ~]$ ls /mnt/nfs  
home www
```

Рис. 3.19: Проверка содержимого www на клиенте

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. 3.20).



```
[admosolov@server ~]$ mkdir -p -m 700 ~/common  
[admosolov@server ~]$ cd ~/common  
[admosolov@server common]$ touch admosolov@server.txt  
[admosolov@server common]$ mkdir -p /srv/nfs/home/user  
mkdir: cannot create directory '/srv/nfs/home/user': Permission denied  
[admosolov@server common]$ mkdir -p /srv/nfs/home/admosolov
```

Рис. 3.20: Подготовка каталога пользователя

Выполняю bind-монтирование личного каталога пользователя в дерево NFS и открываю файл экспорта для настройки (рис. 3.21).



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

Рис. 3.21: Bind-монтирование каталога пользователя

Добавляю в /etc/exports запись для /srv/nfs/home/admosolov, разрешая доступ на чтение и запись для локальной сети (рис. 3.22).



```
admosolov@server:~/common  
GNU nano 5.6.1 /etc/exports  
/srv/nfs *(ro)  
/srv/nfs/www 192.168.0.0/16(rw)  
/srv/nfs/home/admosolov 192.168.0.0/16(rw)
```

Рис. 3.22: Обновление exports для пользователя

Применяю изменения конфигурации NFS без перезагрузки службы, используя команду `exportfs -r` (рис. 3.23).

```
[admosolov@server common]$ sudo nano /etc/exports
[admosolov@server common]$ sudo exportfs -r
[admosolov@server common]$
```

Рис. 3.23: Применение изменений экспорта

Для проверки прав на запись перехожу на клиенте в личный каталог пользователя и создаю там тестовый файл `admosolov@client.txt` (рис. 3.24).

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

Рис. 3.24: Создание файла на клиенте

Возвращаюсь на сервер и проверяю содержимое исходного каталога `~/common`. Файл, созданный на клиенте, успешно появился, что подтверждает корректную работу `rw`-доступа (рис. 3.25).

```
[admosolov@server common]$ ls
admosolov@client.txt admosolov@server.txt
[admosolov@server common]$
```

Рис. 3.25: Проверка файла на сервере

Начинаю этап автоматизации. Создаю на сервере каталог для хранения конфигурационных файлов `provisioning`-скрипта, копирую туда текущий `/etc/exports` и создаю скрипт `nfs.sh` (рис. 3.26).

```
[admosolov@server common]$ cd /vagrant/provision/server
[admosolov@server server]$ mkdir -p /vagrant/provision/server/nfs/etc
bash: mkdir -p: command not found...
[admosolov@server server]$ mkdir -p /vagrant/provision/server/nfs/etc
[admosolov@server server]$ cp -R /etc/exports /vagrant/provision/server/nfs/etc/
[admosolov@server server]$ cd /vagrant/provision/server
[admosolov@server server]$ touch nfs.sh
[admosolov@server server]$ chmod +x nfs.sh
```

Рис. 3.26: Подготовка скрипта сервера

Наполняю скрипт `nfs.sh` командами, которые я выполнял вручную: установка пакетов, копирование конфигов, настройка `firewall`, `SELinux`, создание каталогов, монтирование и обновление `fstab` (рис. 3.27).

```
#!/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
```

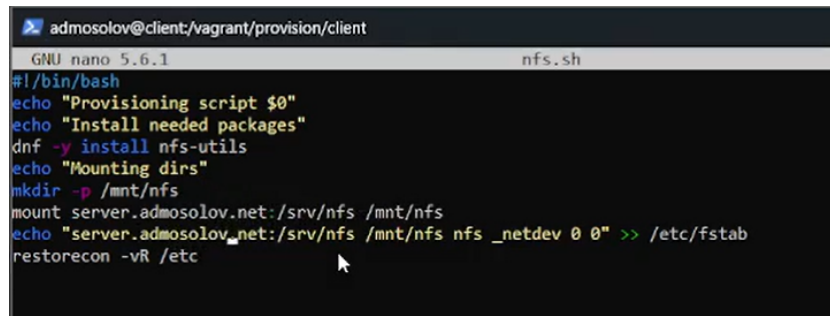
Рис. 3.27: Скрипт nfs.sh для сервера

Аналогичные действия выполняю для клиента: создаю структуру каталогов в /vagrant/provision/client и создаю исполняемый скрипт настройки (рис. 3.28).

```
[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. 3.28: Подготовка скрипта клиента

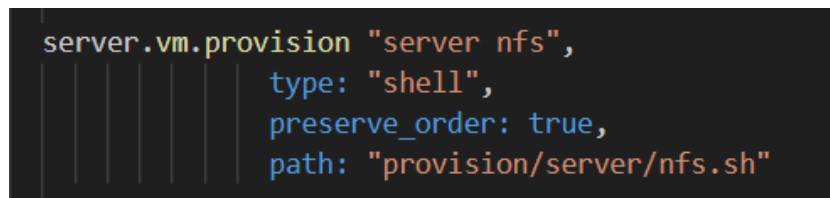
В скрипт клиента добавляю команды установки nfs-utils, создания точки монтирования, монтирования ресурса и добавления записи в /etc/fstab для авто-монтирования (рис. 3.29).

A terminal window titled 'admosolov@client/vagrant/provision/client' showing the contents of a file named 'nfs.sh' in nano editor. The script contains the following commands:

```
#!/bin/bash
echo "Provisioning script $0"
echo "Install needed packages"
dnf -y install nfs-utils
echo "Mounting dirs"
mkdir -p /mnt/nfs
mount server.admosolov.net:/srv/nfs /mnt/nfs
echo "server.admosolov.net:/srv/nfs /mnt/nfs nfs _netdev 0 0" >> /etc/fstab
restorecon -vR /etc
```

Рис. 3.29: Скрипт nfs.sh для клиента

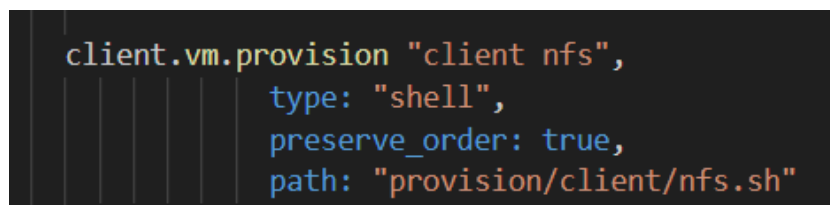
В завершение я настраиваю Vagrantfile. Добавляю блок provisioning для сервера, указывая путь к созданному shell-скрипту (рис. 3.30).

A snippet of Vagrantfile configuration for the server, showing a provisioning block:

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

Рис. 3.30: Конфигурация Vagrantfile для сервера

Также добавляю блок provisioning для клиента, чтобы он автоматически настраивался при разворачивании виртуальной машины (рис. 3.31).

A snippet of Vagrantfile configuration for the client, showing a provisioning block:

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

Рис. 3.31: Конфигурация Vagrantfile для клиента

4 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.

5 Ответы на контрольные вопросы

1. **Как называется файл конфигурации, содержащий общие ресурсы NFS?** Файл конфигурации называется `/etc/exports`. В нем описываются экспортируемые каталоги, разрешенные клиенты и параметры доступа.
2. **Какие порты должны быть открыты в брандмауэре, чтобы обеспечить полный доступ к серверу NFS?** Для работы NFSv4 достаточно открыть TCP порт 2049. Однако для полной функциональности и совместимости (включая RPC, `mountd`) обычно требуется открыть порты для служб: `nfs` (2049), `rpc-bind` (111) и `mountd` (20048).
3. **Какую опцию следует использовать в `/etc/fstab`, чтобы убедиться, что общие ресурсы NFS могут быть установлены автоматически при перезагрузке?** Следует использовать опцию `_netdev`. Она указывает системе, что устройство требует сетевого подключения, и предотвращает попытки монтирования до того, как сеть будет полностью инициализирована.