



## **# Методическое пособие по выполнению домашнего задания по курсу "Администратор Linux. Professional"**

### **# Стенд Vagrant с NFS**

#### **## Цель домашнего задания**

- Научиться самостоятельно развернуть сервис NFS и подключить к нему клиента

#### **## Описание домашнего задания**

Основная часть:

- `vagrant up` должен поднимать 2 настроенных виртуальных машины (сервер NFS и клиента) без дополнительных ручных действий; - на сервере NFS должна быть подготовлена и экспортирована директория;
- в экспортированной директории должна быть поддиректория с именем `__upload__` с правами на запись в неё;
- экспортированная директория должна автоматически монтироваться на клиенте при старте виртуальной машины (systemd, autofs или fstab - любым способом);
- монтирование и работа NFS на клиенте должна быть организована с использованием NFSv3 по протоколу UDP;
- firewall должен быть включен и настроен как на клиенте, так и на сервере.

Для самостоятельной реализации:

- настроить аутентификацию через KERBEROS с использованием NFSv4. **##**

#### **Инструкция по выполнению домашнего задания**

Требуется предварительно установленный и работоспособный [Hashicorp Vagrant](<https://www.vagrantup.com/downloads>) и [Oracle VirtualBox]([https://www.virtualbox.org/wiki/Linux\\_Downloads](https://www.virtualbox.org/wiki/Linux_Downloads)). Также имеет смысл предварительно загрузить образ CentOS 7 2004.01 из Vagrant Cloud командой ``vagrant box add centos/7 --provider virtualbox --box version 2004.01 --clean``, т.к. предполагается, что дальнейшие действия будут производиться на таких образах. Все дальнейшие действия были проверены при использовании CentOS 7.9.2009 в качестве хостовой ОС, Vagrant 2.2.18, VirtualBox v6.1.26 и образа CentOS 7 2004.01 из Vagrant Cloud. Серьёзные отступления от этой конфигурации могут потребовать адаптации с вашей стороны.

#### **### Создаём тестовые виртуальные машины**

Для начала, предлагается использовать этот шаблон для создания виртуальных машин:

```
```ruby
# -*- mode: ruby -*-
# vi: set ft=ruby : vsa
```

```

Vagrant.configure(2) do |config|
  config.vm.box = "centos/7"
  config.vm.box_version = "2004.01"

  config.vm.provider "virtualbox" do |v|
    v.memory = 256
    v.cpus = 1
  end

  config.vm.define "nfss" do |nfss|
    nfss.vm.network "private_network", ip: "192.168.50.10", virtualbox__intnet:
    "net1"
    nfss.vm.hostname = "nfss"
  end

  config.vm.define "nfsc" do |nfsc|
    nfsc.vm.network "private_network", ip: "192.168.50.11", virtualbox__intnet:
    "net1"
    nfsc.vm.hostname = "nfsc"
  end
end

```

Результатом выполнения команды `vagrant up` станут 2 виртуальных машины:  
 \_\_nfss\_\_ для сервера NFS и \_\_nfsc\_\_ для клиента.

### ### Настраиваем сервер NFS

```

- заходим на сервер
```bash
vagrant ssh nfss
```

```

Дальнейшие действия выполняются \_\_от имени пользователя имеющего  
 повышенные привилегии\_\_, разрешающие описанные действия.

```

- сервер NFS уже установлен в CentOS 7 как часть дистрибутива, так что нам нужно
лишь доустановить утилиты, которые облегчат отладку ```bash
yum install nfs-utils
```

```

```

- включаем firewall и проверяем, что он работает (доступ к SSH
обычно включен по умолчанию, поэтому здесь мы его не затрагиваем, но имейте это
ввиду, если настраиваете firewall с нуля) ```bash
systemctl enable firewalld --now

```

```

- разрешаем в firewall доступ к сервисам NFS
```bash
firewall-cmd --add-service="nfs3" \
--add-service="rpc-bind" \
--add-service="mountd" \
--permanent
firewall-cmd --reload

```

```

- включаем сервер NFS (для конфигурации NFSv3 over UDP он не требует дополнительной настройки, однако вы можете ознакомиться с умолчаниями в файле `___/etc/nfs.conf___`)

```bash

systemctl enable nfs --now

```

- проверяем наличие слушаемых портов 2049/udp, 2049/tcp, 20048/udp, 20048/tcp, 111/udp, 111/tcp (не все они будут использоваться далее, но их наличие сигнализирует о том, что необходимые сервисы готовы принимать внешние подключения)

```

ss -tnplu

```

- создаём и настраиваем директорию, которая будет экспортирована в будущем

```bash

mkdir -p /srv/share/upload

chown -R nfsnobody:nfsnobody /srv/share

chmod 0777 /srv/share/upload

```

- создаём в файле `___/etc/exports___` структуру, которая позволит экспортировать ранее созданную директорию

```bash

cat << EOF > /etc/exports

/srv/share 192.168.50.11/32(rw,sync,root\_squash)

EOF

```

- экспортируем ранее созданную директорию

```bash

exportfs -r

```

- проверяем экспортированную директорию следующей командой ```bash

exportfs -s

```

Вывод должен быть аналогичен этому:

```

[root@nfss ~]# exportfs -s

/srv/share

192.168.50.11/32(sync,wdelay,hide,no\_subtree\_check,sec=sys,rw,secure

,root\_squash,no\_all\_squash)

```

### ### Настраиваем клиент NFS

- заходим на сервер

```bash

vagrant ssh nfsc

```

Дальнейшие действия выполняются `___от имени пользователя имеющего повышенные привилегии___`, разрешающие описанные действия.

- доустановим вспомогательные утилиты

```bash

```
yum install nfs-utils
```

```
'''
```

- включаем firewall и проверяем, что он работает (доступ к SSH обычно включен по умолчанию, поэтому здесь мы его не затрагиваем, но имейте это ввиду, если настраиваете firewall с нуля) ```bash

```
systemctl enable firewalld --now
```

```
systemctl status firewalld
```

```
'''
```

- добавляем в `__/etc/fstab__` строку\_

```
'''
```

```
echo "192.168.56.10:/srv/share/ /mnt nfs vers=3,proto=udp,noauto,x-systemd.automount 0" >> /etc/fstab
```

```
'''
```

и выполняем

```
```bash
```

```
systemctl daemon-reload
```

```
systemctl restart remote-fs.target
```

```
'''
```

Отметим, что в данном случае происходит автоматическая генерация systemd units в каталоге ``/run/systemd/generator/``, которые производят монтирование при первом обращении к каталогу ``/mnt/`` - заходим в директорию ``/mnt/`` и проверяем успешность монтирования ```bash

```
mount | grep mnt
```

```
'''
```

При успехе вывод должен примерно соответствовать этому ```

```
[root@nfsc mnt]# mount | grep mnt
```

```
systemd-1 on /mnt type autofs
```

```
(rw,relatime,fd=46,pgrp=1,timeout=0,minproto=5,maxproto=5,direct,pip e_ino=26801)
```

```
192.168.50.10:/srv/share/ on /mnt type nfs
```

```
(rw,relatime,vers=3,rsize=32768,wsizе=32768,namlen=255,hard,proto=ud
```

```
p,timeo=11,retrans=3,sec=sys,mountaddr=192.168.50.10,mountvers=3,mou
```

```
ntport=20048,mountproto=udp,local_lock=none,addr=192.168.50.10) ```
```

Обратите внимание на ``vers=3`` и ``proto=udp``, что соответствует NFSv3 over UDP, как того требует задание.

### Проверка работоспособности

- заходим на сервер

- заходим в каталог ``/srv/share/upload``

- создаём тестовый файл ``touch check_file``

- заходим на клиент

- заходим в каталог ``/mnt/upload``

- проверяем наличие ранее созданного файла

- создаём тестовый файл ``touch client_file``

- проверяем, что файл успешно создан

Если вышеуказанные проверки прошли успешно, это значит, что проблем с правами нет.

Предварительно проверяем клиент:

- перезагружаем клиент

- заходим на клиент

- заходим в каталог ``/mnt/upload``

- проверяем наличие ранее созданных файлов

Проверяем сервер:

- заходим на сервер в отдельном окне терминала
- перезагружаем сервер
- заходим на сервер
- проверяем наличие файлов в каталоге `/srv/share/upload/` - проверяем статус сервера NFS `systemctl status nfs` - проверяем статус firewall `systemctl status firewalld` - проверяем экспорты `exportfs -s`
- проверяем работу RPC `showmount -a 192.168.50.10`

Проверяем клиент:

- возвращаемся на клиент
- перезагружаем клиент
- заходим на клиент
- проверяем работу RPC `showmount -a 192.168.50.10` - заходим в каталог `/mnt/upload`
- проверяем статус монтирования `mount | grep mnt`
- проверяем наличие ранее созданных файлов
- создаём тестовый файл `touch final_check`
- проверяем, что файл успешно создан

Если вышеуказанные проверки прошли успешно, это значит, что демонстрационный стенд работоспособен и готов к работе.

### ### Создание автоматизированного Vagrantfile

Ранее предложенный Vagrantfile предлагается дополнить до такого: ```ruby

```
# -*- mode: ruby -*-
```

```
# vi: set ft=ruby :
```

```
Vagrant.configure(2) do |config|
  config.vm.box = "centos/7"
  config.vm.box_version = "2004.01"
```

```
  config.vm.provider "virtualbox" do |v|
    v.memory = 256
    v.cpus = 1
  end
```

```
  config.vm.define "nfss" do |nfss|
    nfss.vm.network "private_network", ip: "192.168.50.10", virtualbox____intnet:
    "net1"
    nfss.vm.hostname = "nfss"
    nfss.vm.provision "shell", path: "nfss_script.sh" end
```

```
  config.vm.define "nfsc" do |nfsc|
    nfsc.vm.network "private_network", ip: "192.168.50.11", virtualbox____intnet:
    "net1"
    nfsc.vm.hostname = "nfsc"
    nfsc.vm.provision "shell", path: "nfsc_script.sh" end
end
```

...

и далее создать 2 bash-скрипта, `nfss\_script.sh` - для конфигурирования сервера и `nfsc\_script.sh` - для конфигурирования клиента, в которых описать bash-командами ранее выполненные шаги. \_Альтернатива - воспользоваться Ansible.\_

После того, как вы опишете конфигурацию для автоматизированного развёртывания, уничтожьте тестовый стенд командой `vagrant destroy -f`, создайте его заново и выполните все пункты из \_\_[Проверка работоспособности](#Проверка-работоспособности)\_\_, убедившись, что всё работает как задумывалось и требуется.

### ### Документация

Создайте файл README.md и снабдите его следующей информацией: - название выполняемого задания;

- текст задания;
- описание каталогов и файлов в репозитории;
- особенности проектирования и реализации решения, в т.ч. существенные отличия от того, что написано в выше;
- заметки, если считаете, что имеет смысл их зафиксировать в репозитории.

### ## Критерии оценивания

5 баллов - стенд разворачивается автоматически при выполнении `vagrant up`, работоспособен, а в приложенной документации дана исчерпывающая информация о стенде, реализации решения и особенностях, если таковые имеются.

Проверяющий имеет право снизить итоговый балл на своё усмотрение, если какая-либо часть работы не реализована, к примеру, отсутствует документация.

1 балл - дополнительно реализована аутентификация с использованием KERBEROS (NFSv4)

Максимальная оценка - 6 баллов.