Chapter 7:
File Sharing Services

Learning objectives

Upon completing this chapter, the learner should be able to:

- Understand the concepts of NFS file sharing
- Setup and configure basic NFS server
- Setup and configure NFS client
- Understand the concepts of Samba file sharing
- Setup and configure FTP service
- Setup and configure FTP client
- Understand and use the scp tool
- Understand and use the rsync tool

Key terms

Network File System (NFS)

Pseudo root mounts

Exports file

Mounting NFS shares

Automount

Indirect map

Direct map

Samba server

smb.conf

Samba account

Samba-only

File Transfer Protocol (FTP)

FTP server

FTP client

vsftpd.conf

lftp

Secure Copy (scp)

Remote SYNC (rsync)

Table of content

1. Working with NFS 4
   1.1. Configuring the NFSv4 Server 4
   1.2. Setting up a base NFS server 7
   1.3. Setting up an NFS client 8
      1.3.1. Testing client access with showmount 8
      1.3.2. Mounting NFS shares 9
      1.3.3. Making NFS mounts persistent 9
      1.3.4. Configuring automount for NFS 11
2. Working with Samba file services 13
   2.1. Setting up Samba file sharing 13
      2.1.1. Installing Samba 14
      2.1.2. Preparing shared directories on Linux 14
      2.1.3. Configuring /etc/samba/smb.conf 15
      2.1.4. Creating shares 17
      2.1.5. Using Samba users 17
      2.1.6. Mounting Samba shares 18
3. Working with FTP 19
   3.1. Setting up FTP server 19
   3.2. Setting up FTP client 21
4. Other common transferring tools 23
   4.1. The Secure CP (scp) tool 23
   4.2. The Remote SYNC (rsync) tool 25
      4.2.1. Syntax and options 26
      4.2.2. Examples 27

1. Working with NFS

NFS (Network File System) is the classic network file system, its purpose is to make it possible to mount remote file systems into the local file system hierarchy. NFS provides the classical way of sharing files between UNIX and Linux hosts, where an NFS server offers shares, which are also referred to as exports, and the NFS client mounts the share into its local file system.

What makes NFS an efficient protocol is that: shares are mounted in the local file system, which provides tree structure enables access to files, no matter where these files physically are. Because it is such a simple protocol with relatively low overhead, NFS is still commonly used, in two cases in particular:

1. To provide access to home directories for Lightweight Directory Access Protocol (LDAP) users.
2. To easily access shared file systems on other Linux servers, which makes transferring files between servers easier.

1.1. Configuring the NFSv4 Server

On RHEL7, NFS4 is the default version of NFS. Even if NFSv4 offers interesting enhancements as compared to earlier versions of the protocol, backward compatibility is also available. This might be useful to offer support to NFS clients that support previous versions only. If when making an NFS mount the NFS server offers a previous version of NFS, the client falls automatically back to that version.

NFS offers some interesting enhancements as compared to earlier versions of the protocol:

1. **Pseudo root mounts**: A client that has access to several NFS shares on specific server no longer has to mount every individual mount but may perform a root mount. Thus, the client can mount the root directory on the NFS server, which provides access to all shares exported to the client on that server.
2. **Kerberos security**: NFS was designed to use host-based security only. Once the host is authenticated, user-based restrictions are limited. Kerberos security is added to add encryption but also to give access to users only after their Kerberos credentials have been verified.
3. **Simplified firewalling**: NFSv4 offers simplified firewalling, where TCP port 2049 is opened in the firewall to allow access to all NFS processes, and ports 111 as well as 2049 are opened for full client access.

On the NFS server, to create an export, the /etc/exports file has to include:

- The name of the directory that is exported
- The NFS clients to whom it is exported
- The options used to create the export

This export definition can be created in the /etc/exports file, but also as a file with the extension. exports in the directory /etc/exports.d. An export line might look like this:

```
[root@server1:~]# cat /etc/exports
…
/srv/nfsexport vm[1-10].example.com(rw,no_root_squash)
```

To define which hosts (clients) will get access to the share, you can refer to those hosts in the second part of the export definition. In this example, access is granted to the hosts vm1 through vm10. Other approaches used as well, including * for all hosts, specific hostnames, specific IP addresses, and complete network addresses such as 10.50.0.0/16.

While creating the export, specific export options can be used as well (the third part of the export definition), including the following:

- **The option ro**: The share is exported as read-only, which ensures that no matter which permissions a remote user has on the share, no files can be created or modified.
- **The option rw**: The share is exported as read/write, which enables writes, but only if the user has writing permissions on the NFS Linux server file system of the exported directory as well.
- **The option no_root_squash**: By default, the user root is mapped to the user nfsnobody on the NFS server. This ensures that a user who comes in as root from an NFS client has minimal permissions on the NFS server. If you want to grant full access to the user root from an NFS client, use the no_root_squash option. Realize, though, that this is a very insecure option.

By default, NFS security is limited, where it is allowed or denied based on the hostname that wants to access the share. If the hostname is allowed, the share can be mounted and accessed by users from the NFS client. When a client accesses an NFS share, the NFS server by default maps the UID of the client user to the same UID on the NFS server. This can lead to unexpected results.
If on the NFS server user user1 has UID 505, and on the client user user2 has UID 505, for example, user2 will access the NFS share as UID 505, which is mapped to UID user1 on the server. So, user2 will have access to all files that user1 has access to on the server.

1.2. Setting up a base NFS server

In this section, you set up a basic NFSv4 server and open the firewall for client access to this server, the following illustrates the required steps:

1. Install required packages on server (nfs-utils policycoreutils-python).
2. Define sharing points in the /etc/exports with the desired permissions options.
3. Create the shared directories on the file system according to defined sharing points.
4. Activate all changes to /etc/exports by using the exportfs -r command.
5. If the SELinux is activated on the server, the NFS must be set in the context on the desired shares. The semanage and restorecon commands are used as follows:

```
[root@server1:~]# semanage fcontext -a -t nfs_t "/path/to/shared/directory"?(*./)

[root@server1:~]# restorecon -Rv /path/to/shared/directory
```

6. If it's not yet, launch and enable the NFS server using the systemctl command
7. If the firewall is activated on the server, the nfs, mountd and rpc-bind services must be added to its configuration by using the firewall-cmd --permanent --add-service <service-name> command, and off course reloading the settings

1.3. Setting up an NFS client

We called a client in this case; any Linux machine needs to get access to the shared directories in that sever we'd created using the NFS service. In general, it's a mounting procedure that solves the problem with a pre checking and testing.

1.3.1. Testing client access with showmount

Now that you have created an NFS share on the server, it is time to access it but we need to test whether a mount is available, you can use the showmount command, the command showmount -e hostname shows all mounts that are available on hostname. On RHEL7, you can access the NFS server if ports 2049 and 111 have been opened in the firewall. You can accomplish this using the following commands mentioned in the previous section.

```
[root@client:~]# showmount -e 192.168.114.148
Export list for 192.168.114.148:
/srv/nfs       *
/srv/nfsexport *
```

1.3.2. Mounting NFS shares

To mount an NFS share, you first need to find the names of the shares. This information can be provided by the administrator, but it is also possible to find out yourself. To discover which shares are available, you have multiple options:

- If NFSv4 is used on the server, you can use a root mount. That means that you just mount the root directory of the NFS server, and under the mount point you'll only see the shares that you have access to
- Use netstat -an | grep your.nfs.server.ip:port to verify the availability of the mount
- Use the showmount -e nfsserver command to find out which shares are available

Now, to create the mount, just type a command using the structure: mount nfsserver:/sharename /mountpoint, and the mount will now be integrated in the local file system.

1.3.3. Making NFS mounts persistent

The /etc/fstab file is used to mount file systems that need to be mounted automatically when a server restart. Only the user root can add mounts to this configuration file, thus providing shares that will be available for all users, this file can be used to mount the NFS file system as well as Samba. To mount an NFS file system through /etc/fstab, make sure that the following line is included:

```
[root@client:~]# cat /etc/fstab
...
server1:/share /nfs/mount/point nfs,x-systemd.automount,sync  0 0
```

When making an NFS mount through fstab, you have a few options to consider:

- In the first column, you need to specify the server and share name. Use a colon after the name of the server to identify the mount as an NFS share.
- The second column has the file system where you want to do the mount; this is not different from any regular mount.
- The third column contains the NFS file system type.
- The fourth column that is used to specify mount options includes the sync option, this ensures that modified files are committed to the remote file system immediately and are not placed in write buffers first (which would increase the risk of data getting lost).
- The fifth column contains a zero, which means that no backup support through the dump utility is requested.
- The sixth column also contains a zero, to indicate that no fsck has to be performed on this file system while booting to check the integrity of the file system. The integrity of the file system would need to be checked on the server, not on the client.

1.3.4. Configuring automount for NFS

As an alternative to using /etc/fstab, you can configure automount to mount the share automatically. Automount can be used for SMB as well as NFS mounts, and the big difference is that mounts through automount are affected on demand and not by default. So, using automount ensures that no file systems are mounted that are not really needed. An important benefit of using automount is that it works completely in user space, and contrary to mounts that are made through the mount command, no root permissions are required.

On previous versions of RHEL, the auto.master file was used to define the file systems to be automounted. On RHEL7, you can work with snippet files in the /etc/ auto. master.d directory as an alternative. You can still use the old method, though.

To configure automount, the autofs package has to be installed. After you install it, a master-map file needs to be created in the directory /etc/ auto. master.d. The name of this file does not matter, but it needs to end in autofs. In the master map file, the directory is specified that should be monitored by the autofs service. From this file, a second file is referred to that contains the setting with which the automount is performed. When using automount, you can use two different kinds of automount maps:

- An indirect map contains a directory that should be created by automount. Indirect mounting allows for alteration without the need to restart the autofs service, which makes it completely accessible from user space.
- A direct file does not involve creation of a directory, it has to exist before automount can mount the (remote) file system. Configuring an automount solution is a complicated multistep procedure.

Exercise: Configuring NFS service

1. On server1, use yum install nfs-utils policycoreutils-python- y to install the required utilities.
2. Open the file /etc/exports with an editor and add the following line: /srv/nfsexport *(rw).
3. Type mkdir /srv/nfsexport to create the shared directory.
4. Type exportfs -r to make all changes to /etc/exports effective.
5. Set the NFS context on the share:
   semanage fcontext -a -t nfs_t "/srv/nfsexport (/.*)?".
   Type restorecon -Rv /srv/nfsexport to apply the setting to the file system.
6. Launch and enable the NFS server by using: systemctl start nfs -server rpcbind; systemctl enable nfs-server rpcbind.
7. Stop the firewall service, or open it for the nfs - server by using: firewall-cmd --permanent-add-service nfs; firewall-cmd --permanent -add -service rpc -bind; firewall-cmd --permanent -add -service mountd; firewall-cmd --reload.
8. On the client, yum -y install nfs-utils, then type showmount -e <server1-ip-address>, and it will display the shared points from server1.
9. On the client, type mkdir /mnt/nfs; mount server1: /srv/nfsexport /mnt/nfs.
10. Type mount to verify that the NFS share has mounted correctly.
11. Still on the client, open the file /etc/fstab and add the following line to make the mount persistent: server1: /srv/nfsexport /mnt/nfs nfs 0 0
12. Type systemctl status remote -fs.target and verify that this systemd unit is enabled.
13. Restart the client and verify that the mount is activated automatically.

2. Working with Samba file services

In many networks, Windows clients are used to access services on servers that are running on Linux more and more often. To allow these clients to use native file access protocols to access file shares, you can configure the Samba service on Linux.

Samba is a common solution that ensures the best possible compatibility for file exchange between different operating systems. On Red Hat Enterprise Linux, you can configure the Samba server to provide access to clients using the Server Message Block (SMB) or Common Internet File System (CIFS) protocols to access these shares.

2.1. Setting up Samba file sharing

Setting up a Samba file server involves a few steps:

1. Install Samba packages.
2. Prepare directories on Linux.
3. Prepare permissions on Linux.
4. Create the share in /etc/samba/smb.conf.
5. Create Samba user accounts.
6. Mounting Samba shares.

These steps are discussed in detail in the following subsections.

2.1.1. Installing Samba

To install the Samba server packages and the client tools, make sure to install the following RPMs:

- **samba**: Contains the Samba daemons and related configuration files
- **Cifs-utils**: Contains the Samba client packages, including the command you need to mount remote SMB shares
- **Samba-client**: Contains some utilities that are required to set up Samba shares

You can do this by executing: `yum install samba cifs-utils samba-client`.

2.1.2. Preparing shared directories on Linux

To configure a server to share directories with the Samba server, you first need to set up the Linux part of the configuration. When a user authenticates to the Samba server, a Samba user account is used, but the Samba user account is mapped to a Linux user account, and that user account needs access permissions.

Therefore, you have to create a directory and set appropriate permissions on that directory. You can do that in several ways: the easy way, where you just set permission mode 777 on the shared directory, or the more sophisticated way, where you set up directories with group owners and the permissions that members of that group need to access the directory. You can also work with access control lists (ACLs) for more sophisticated access control to the shared directories.

2.1.3. Configuring /etc/samba/smb.conf

The main part of the Samba configuration itself is set in the file /etc/samba/smb.conf. This file contains two parts:

- The first part is the [global] section, where generic properties of the Samba service are defined
- The second part contains the share definitions, where share specific settings are defined, two special shares may be enabled as well:
  - [homes] contains default values for accessing home directories that are shared through Samba.
  - [printers] is used to provide access to printers that are shared using the CUPS printing system.

Listing shows how these shares are defined in a default smb.conf file.

```
[root@server1:~]# cat /etc/samba/smb.conf
# See smb.conf.example for a more detailed config file or
# read the smb.conf manpage.
# Run 'testparm' to verify the config is correct after
# you modified it.

[global]
        workgroup = SAMBA
        security = user

        passdb backend = tdbsam
        printing = cups
        printcap name = cups
        load printers = yes
        cups options = raw
[homes]
        comment = Home Directories
        valid users = %S, %D%w%S
        browseable = No
        read only = No
        inherit acls = Yes

[printers]
        comment = All Printers
        path = /var/tmp
        printable = Yes
        create mask = 0600
        browseable = No

[print$]
        comment = Printer Drivers
        path = /var/lib/samba/drivers
        write list = @printadmin root
        force group = @printadmin
        create mask = 0664
        directory mask = 0775
```

2.1.4. Creating shares

To create a directory share, you need to add a section near the end of the smb.conf file. The share name is placed in brackets and is followed by the directives that further define the share.

Before testing the share settings, use the testparm command. This command tells you whether any syntax errors exist in the smb.conf file. It will not test for any logical errors, though, so it does not complain if you are trying to provide access to a share for users that do not exist.

2.1.5. Using Samba users

When the security =user setting is used, you need to create two accounts to enable access to shared files and directories:

- A Linux account that has the appropriate Linux permissions on the share
- A Samba account that has a name that matches the Linux account

Samba-only users are user accounts that are used by Windows users who are connecting to a Samba share but that do not require login to a Linux terminal as well. For these Samba-only users, you do not have to set a Linux password. Even better, while creating the user, set the login shell to /sbin/nologin, which prevents the user from ever logging in to a terminal on your server. To create Samba users, complete the following steps:

1. Create the Linux user (if it does not exist already) using useradd -s /sbin/nologin user1.
2. Add the samba user account, using smbpasswd -a user1, enter the password twice when prompted.

Notice that if the smbpasswd command is used without the -a option, the command tries to change the password for existing Samba users. So, make sure to create new users that use the -a option.

2.1.6. Mounting Samba shares

Before connecting to a Samba share, it is a good idea to just list the Samba shares that are available on the Samba server. To do this, you can use smbclient -L command, followed by the name of the host that is offering Samba services. Also, you need to add to the client the sambaclient service to the firewall configuration on the client by using firewall-cmd --add-service samba-client --permanent; firewall-cmd --reload.

After installing this, you can use the smbclient -L command to discover available SMB shares. The most common way to access Samba shares is by mounting the share into the local Linux file system.

To mount an SMB file system, use the mount command followed by the name of the Samba user whose credentials you want to use. For instance, to mount a share as user user1, use: mount -o username=user1 //server1/sambashare/mnt. You then need to enter the password for the user specified and you'll be authenticated.

On the Linux server which is going to be used as the Samba client, the cifs-utils package needs to be installed. While mounting the share, you need to specify a username and password, this username and password are going to be used as the mount credentials and apply to everyone who is using the mount.

You can use the -t cifs option to specify that the mount is to an SMB share, but without this option it will also work because the mount command is smart enough to discover by itself that it is an SMB share you want to connect to.

3. Working with FTP

FTP has always been a common service in Linux environments, every system administrator should know at least how to configure an FTP server.

3.1. Setting up FTP server

The following steps show how to briefly install and activate the FTP server on RHEL7:

1. Install the required packages (vsftpd), it's the same as the service name.
2. Start the service and set it to launch when the system boots with the systemctl command.
3. If the firewall is activated, enable the ftp service on it with the port 21: firewall-cmd --permanent --add-port=21/tcp; firewall-cmd --permanent --add-service=ftp; firewall-cmd –reload.
4. Configure the FTP service by setting appropriate options in the file /etc/vsftpd/vsftpd.conf.

The following are some of common options:

Option	Explanation
anonymous_enable=NO	disable anonymous login
local_enable=YES	permit local logins
write_enable=YES	enable FTP commands which change the filesystem
local_umask=022	value of umask for file creation for local users
xferlog_enable=YES	a log file will be maintained detailing uploads and downloads
connect_from_port_20=YES	use port 20 (ftp-data) on the server machine for PORT style connections
userlist_enable=YES	load a list of usernames, from the filename given by userlist_file
userlist_file=/etc/vsftpd. userlist	stores usernames
userlist_deny=NO	value of umask for file creation for local users
chroot_local_user=YES	local users will be placed in a chroot jail, their home directory after login by default settings
allow_writeable_chroot=YES	allow the chroot jail directory to be writable

5. Prepare the file system document root,
   - The FTP server uses the directory /var/ftp as the default document root, in it you can create a subdirectory and set the suitable permissions as desired.
   - When enabling the FTP server to work with local users files, you can put the documents root in users home directories.

3.2. Setting up FTP client

Now, on any RHEL7 you can make it as FTP client to the server we made above by simply applying the following procedure:

- First of all, an FTP client must be installed and generally it's lftp command-line tool.
- By assuming that a connection is opened between these two machines, use the command lftp -u <username> <server-name | server-ip> to reach the remote machine.
- You can now list the available directories on the remote server.
- Now, send files through the FTP connection by using the command put <files>.

Exercise: Configuring FTP service

1. On server1, as a root, use yum install vsftpd.
2. Start the service and set it to launch when the system boots with the following: systemctl start vsftpd, systemctl enable vsftpd.
3. Next, create a rule for your firewall to allow FTP traffic on Port 21: firewall-cmd --permanent –add –port=21/tcp; firewall-cmd –permanent –add –service=ftp; firewall-cmd –reload.
4. create a copy of the default configuration file: sudo cp /etc/vsftpd/vsftpd.conf /etc/vsftpd/vsftpd.conf.default.
5. Next, edit the configuration file and set the following options as displayed:

   Anonymous_enable=NO

   Local_enable=YES

   Write_enable=YES

   Chroot_local_user=YES

   Allow_writeable_chroot=YES

   Userlist_enable=YES

   Userlist_file=/etc/vsftpd/user_list

   Userlist_deny=NO

6. Save your changes, and restart the vsftpd service to apply changes.
7. Create a new user testuser, and type: mkdir –p /home/testuser/ftp/upload; chmod 550 /home/testuser/ftp; chmod 750 /home/testuser/ftp/upload; chown –R testuser: /home/testuser/ftp.
8. On client machine, install the lftp package.
9. Connect to the server1 using the command: lftp –u testuser <server1 –ip –address>.
10. List the file system tree there by using ls command.
11. Type in: put /etc/passwd and notice the message that ensures transferring data.
12. Checking the existence of the copied files back on the server1 testuser home-directory.

4. Other common transferring tools

If a host is running the sshd service, that service can also be used to securely transfer files between systems, to do that a couple of tools are used. In this section we will present the most common ones: the scp and the rsync tools.

4.1. The Secure CP (scp) tool

This command is very similar to the cp command, which is used to copy local files, but it does include an option that allows it to work with remote hosts. You can use scp to copy files and subdirectories to remote hosts, and subdirectories as well. The syntax is so simple, use scp followed by user@remote-server:/path/to/remote/files in all cases where you need to bring from remote server or you need to put there.

For instance, to copy the /etc/hosts file on the client machine (the source) to the /tmp directory on server1 machine (the destination) using the root account, use the following command:

```
[root@client:~]# scp /etc/hosts root@192.168.114.148:/tmp
hosts                    100%  158     75.0KB/s   00:00
```

On the remote destination you can check the existence of the transferred file:

```
[root@server1:~]# ll /tmp
total 4
-rw-r--r--. 1 root root 158 Dec 14 03:01 hosts
...
```

You can also use scp to copy an entire subdirectory structure, to do so, use the -r switch, as in the following command:

```
[root@client:~]# scp -r /etc/ root@192.168.114.148:/tmp
fstab
100%  524   216.3KB/s   00:00crypttab
100%    0     0.0KB/s   00:00mtab
100%    0     0.0KB/s   00:00resolv.conf
```

Notice that the scp command can be configured to connect to a non-default SSH port also. It is a bit confusing, but to do this with the scp command, you need the -P option followed by the port number you want to connect to. Notice that ssh uses -p (lowercase) to specify the port it needs to connect to; the scp command uses an uppercase -P.

4.2. The Remote SYNC (rsync) tool

As an alternative to using scp for copying files between servers, you might be interested in the rsync command. The basic use of rsync is similar to the use of scp, and it also uses the sshd service to securely copy files. Rsync, however, does offer advanced options that enable you to synchronize files and directories based on a delta sync. That means that only differences are synchronized, which makes this a very efficient solution for keeping files and directories the same between hosts.

It offers a large number of options that control every aspect of its behavior and permit very flexible specification of the set of files to be copied. Some of the additional features of rsync are:

- Support for copying links, devices, owners, groups, and permissions
- Exclude and exclude-from options similar to GNU tar
- Does not require super-user privileges
- Support for anonymous or authenticated rsync daemons (ideal for mirroring)

4.2.1. Syntax and options

Local Sync: rsync [OPTION...] SRC... [DEST]

Remote Sync pull: rsync [OPTION...] [USER@]HOST:SRC... [DEST]

Remote Sync Push: rsync [OPTION...] SRC... [USER@]HOST:DEST

Some of the commonly used options in rsync command are listed below:

Option	Description
-v, --verbose	Increase verbosity
-a, -e-archive	Archive mode, equals-rlptgoD
-b, --backup	Make backups
-u, --update	Skip files that are newer on the receiver
-l, --links	Copy symlinks as symlinks
-p, --perms	Preserve permissions
-o, --owner	Preserve owner (super-user only)
-g, --group	Preserve group
-D	Same as --devices-specials --devices: preserve device files (super-user only) --specials: preserve special files
-t, --times	Preserve modification times
--existing	Skip creating new files on receiver
--delete	Delete extraneous files from dest dirs
--partial	Keep partially transferred files
--size-only	Skip files that match in size
-z, --compress	Compress file data during the transfer
--exclude=PATTERN	Exclude files matching PATTERN
-h, --human-readable	Output numbers in a human-readable format
--progress	Show progress during transfer
--log-file=FILE	Log what we're doing to the specified FILE

4.2.2. Examples

The following listing shows some good and common examples of using the rsync command locally and remotely:

Option	Description
rsync -zavh	Sync files and directories recursively locally.
rsync -zarvh /local/path user@remote:/path	Sync files and directories from local to remote system.
rsync -zarvh user@remote:/path /local/path	Sync files and directories from remote machine to local system.
rsync -avh --progress	Display Synchronization progress in rsync command output.
rsync -av -f"+ */" -f"- *"	Copy the directory structure without copying files.
rsync -P --rsh=ssh user@remote:/path /local/path	Resume large file transfer after getting failed in scp.
rsync -avz --delete	Delete files at destination if it is not present in source.
rsync -avz --max-size='500K'	Put limit on file transfer size.
rsync -avzu	Do not sync/copy the modified file at destination.
rsync --remove-source-files -zvh	Remove files from source after synchronization.
rsync -avz -e ssh --include '*.pdf *.rpm' --exclude '*.png'	Include and Exclude files during synchronization with rsync.
rsync -avz --progress --bwlimit=600	Put restriction on data transfer speed in rsync.
rsync -avzi	View the difference in files and directories between source and destination.

Quiz

Chapter review questions

1. Which command enables you to show available NFS mounts on server1?
   a. showmount
   b. mount – a nfs
   c. mkfs – t nfs
2. Which command enables you to mount an NFS share that is available on server1:/share?
   a. mount – a nfs
   b. mount server1: /share /mnt/nfs
   c. mount /mnt/nfs server1: /share
3. Which command can you use to discover SMB mounts on a specific server?
   a. showmount
   b. smbclient
   c. mount – a smb
4. Which package must be installed on an SMB client before you can make an SMB mount?
   a. smb
   b. nfs – utils
   c. cifs – utils
5. What is the name of the configuration file that contains the vsftpd configuration?
   a. vsftpd. conf
   b. ftp. conf
   c. /etc/conf/vsftpd

6. You type the command showmount – e to display available mounts on an NFS server, but you do not get any result. Which of the following is the most likely explanation?
   a. The NFS client software is not running.
   b. You are using a UID that does not exist on the server.
   c. SELinux is configured properly.
   d. The firewall does not allow showmount traffic.
7. You want to log in to an SMB share. Which of the following commands shows correct syntax for doing so?
   a. mount – o username = sambauser1 //server/share /somewhere
   b. mount – o uname = sambauser1 //server/share /somewhere
   c. mount sambauser1@//server/share /somewhere
   d. mount – o username = sambauser1@//server/share /somewhere
8. The classical way of sharing files between UNIX and Linux hosts is:
   a. samba
   b. NFS
   c. anonymous
   d. Auto. master
9. Which of the following statements is not true about NFS?
   a. NFS 4 is the default version on RHEL 7
   b. Any shared folder should be added to the /etc/exports file
   c. Used for sharing files between Windows and Linux
   d. NFS4 offers some interesting enhancements
10. A client that has access to several NFS shares on specific server no longer has to mount every individual mount but may perform a root mount by using:
   a. Pseudo root mounts
   b. Kerberos security
   c. Simplified firewalling
   d. complicated firewalling
11. NFS was designed to use host – based security only.
   a. True
   b. False

12. To create an export, the /etc/exports file has to include:
    a. NFS clients – file name – export options
    b. directory permissions – NFS clients – export options
    c. directory name – NFS clients – export options
    d. directory name – Samba clients – export options
13. If you want to grant full access to the user root from an NFS client, use the:
    a. nfsexport
    b. ro
    c. rw
    d. no _ root _ squash
14. Required packages on NFS server:
    a. nfs-utils policycoreutils-python
    b. exports
    c. nfs policycoreutils
    d. utils python
15. This option ensures that modified files are committed to the remote file system immediately.
    a. ro
    b. sync
    c. rw
    d. nfs-utils
16. Two different kinds of automount maps:
    a. simple and complicated
    b. direct and indirect
    c. mount and automount
    d. sync and async
17. Samba client packages are:
    a. Cifs – samba
    b. Client – samba
    c. Cifs – utils
    d. samba

18. On an FTP server configuration, if anonymous _ enable is set to NO anonymous login will be enabled.

a. True

b. False

19. The scp command uses the protocal:

a. SMB

b. NFS

c. SSH

d. None

20. You can use the rsync utility for remote machines only?

a. True

b. False

Answers to chapter Review Questions:

1. a
2. b
3. b
4. c
5. a
6. d
7. a
8. b
9. c
10. a
11. a
12. c
13. d
14. a
15. b
16. b
17. c
18. b
19. c
20. b