PACKAGE MANAGEMENT

RPM (Red Hat Package Management)

rpm is a powerful Package Manager for Red Hat, Suse and Fedora Linux. It can be used to build, install, query, verify, update, and remove/erase individual software packages. A Package consists of an archive of files, and package information, including name, version, and description:

|  |  |  |
| --- | --- | --- |
| **Syntax** | **Description** | **Example(s)** |
| rpm -ivh {rpm-file} | Install the package | rpm -ivh mozilla-mail-1.7.5-17.i586.rpm rpm -ivh --test mozilla-mail-1.7.5-17.i586.rpm |
| rpm -Uvh {rpm-file} | Upgrade package | rpm -Uvh mozilla-mail-1.7.6-12.i586.rpm rpm -Uvh --test mozilla-mail-1.7.6-12.i586.rpm |
| rpm -ev {package} | Erase/remove/ an installed package | rpm -ev mozilla-mail |
| rpm -ev --nodeps {package} | Erase/remove/ an installed package without checking for dependencies | rpm -ev --nodeps mozilla-mail |
| rpm -qa | Display list all installed packages | rpm -qa rpm -qa | less |
| rpm -qi {package} | Display installed information along with package version and short description | rpm -qi mozilla-mail |
| rpm -qf {/path/to/file} | Find out what package a file belongs to i.e. find what package owns the file | rpm -qf /etc/passwd rpm -qf /bin/bash |
| rpm -qc {pacakge-name} | Display list of configuration file(s) for a package | rpm -qc httpd |
| rpm -qcf {/path/to/file} | Display list of configuration files for a command | rpm -qcf /usr/X11R6/bin/xeyes |
| rpm -qa --last | Display list of all recently installed RPMs | rpm -qa --last rpm -qa --last | less |
| rpm -qpR {.rpm-file} rpm -qR {package} | Find out what dependencies a rpm file has | rpm -qpR mediawiki-1.4rc1-4.i586.rpm rpm -qR bash |

YUM

**How To Create Yum Server**

**What Is Yum Server ?**  
 The Yellowdog Updater, Modified (yum) is an open-source command-line package-management utility for Linux operating systems using the RPM Package Manager. Though yum has a command-line interface, several other tools provide graphical user interfaces to yum functionality. Yum allows automatic updates, package and dependency management, on RPM-based distributions. Like the Advanced Packaging Tool (APT) from Debian, yum works with software repositories (collections of packages), which can be accessed locally or over a network connection.

**Advantages of Yum Server-**

* Automatic resolution of software dependencies. If a package installation or upgrade request is made and requires the installation or upgrade of additional packages, YUM can list these dependencies and prompt the user to install or upgrade them.
* Command-line and graphical versions. The command-line version can be run on a system with a minimal number of software packages. The graphical versions offer ease-of-use and a user-friendly graphical interface to software management.
* Multiple software locations at one time. YUM can be configured to look for software packages in more than one location at a time.

**Configure the Yum Server-Server Side:**  
 **1.** insert DVD in to DVD-ROM   
**2.** mount the DVD-ROM   
 #mount /dev/cdrom /mnt/   
  
**3.**go to /mnt/Packages/  
  #cd /mnt/Packages/  
  #ll                           {List all the package}  
  
**4.**now install the yum server rpm   
#rpm -ivh vsftpd-2.2.2-11.el6.i686.rpm   
#rpm -ivh createrepo-0.44...........noarch.rpm  
  
**5.**Create a folder under the Directory  
#mkdir /var/ftp/pub/yum                    {Note-yum is a Directory name}  
  
**6.** Change The Directory Permission  
#chmod -R 777 /var/ftp/pub/yum   
  
**7.**Copy all /mnt Data in this location /var/ftp/pub/yum  
#cp -rvf . /mnt/\* /var/ftp/pub/yum/  
  
**8.**Go to Repository file  
#cd /etc/yum.repos.d/  
#ll  
  
**9.**now create repository file  
#vi server.repo  or vi /etc/yum.repos.d/server.repo  
  
Then Entry-  
  
[server]  
name=Rhel6  
baseurl=file:///var/ftp/pub/yum/  
gpgcheck=0  
enabled=1  
  
Now Save And Exit.  
  
**10.**Createrepo -(is a program that creates a repomd (xml-based rpm metadata)repository from a set of rpms.)  
        
or   
yumrepolist

#createrepo /var/ftp/pub/yum/

**11.**Umount Dvd-ROM and /mnt

#umount /mnt

#umount /dev/cdrom

**12.**Clean all cache memory  
#yum clean all  
  
**13**.restart vdftpd services  
#/etc/init.d/vsftpd restart  
#chkconfig vsftpd on                    {updates  and queries runlevel information for system services}  
  
**14.**Check yum server  
#yum list  
  
  
to install rpm using yum server  
#yum install bind\* -y  
  
to search rpm using yum server  
#yum list | grep rpm\_name   
  
to remove rpm using yum server  
#yum remove rpm\_name  
  
**Client Side:**  
  
**1.**Create Repository File  
#vi /etc/yum.repos.d/1.repo  
  
Then Entry-  
  
[server]  
name=Rhel6  
baseurl=ftp://ip address of server/pub/yum                   {Server IP and Define Proper path}  
gpgcheck=0  
enabled=1  
  
Now Save And Exit.  
  
**2.**Check yum server  
#yum repolist

APT is the Advanced Package Tool is the advanced interface to the Debian packaging system and provides the apt-get program. It features complete installation ordering, multiple source capability and several other unique features, see the User's Guide in /usr/share/doc/apt-doc/guide.html/index.html (you will have to install the apt-doc package).

apt-get provides a simple way to retrieve and install packages from multiple sources using the command line. Unlike dpkg, apt-get does not understand .deb files, it works with the packages proper name and can only install .deb archives from a source specified in /etc/apt/sources.list. apt-get will call dpkg directly after downloading the .deb archives[[5](https://www.debian.org/doc/manuals/debian-faq/footnotes.en.html" \l "f5)] from the configured sources.

Some common ways to use apt-get are:

* To update the list of package known by your system, you can run:
* apt-get update

(you should execute this regularly to update your package lists)

* To upgrade all the packages on your system (without installing extra packages or removing packages), run:
* apt-get upgrade
* To install the foo package and all its dependencies, run:
* apt-get install foo
* To remove the foo package from your system, run:
* apt-get remove foo
* To remove the foo package and its configuration files from your system, run:
* apt-get --purge remove foo
* To upgrade all the packages on your system, and, if needed for a package upgrade, installing extra packages or removing packages, run:
* apt-get dist-upgrade

(The command upgrade keeps a package at its installed obsolete version if upgrading would need an extra package to be installed, for a new dependency to be satisfied. The dist-upgrade command is less conservative.)

Note that you must be logged in as root to perform any commands that modify the system packages.

Note that apt-get now installs recommended packages as default and is the preferred program for package management from console to perform system installation and major system upgrades for its robustness.

The apt tool suite also includes the apt-cache tool to query the package lists. You can use it to find packages providing specific functionality through simple text or regular expression queries and through queries of dependencies in the package management system. Some common ways to use apt-cache are:

* To find packages whose description contain word:
* apt-cache search word
* To print the detailed information of a package:
* apt-cache show package
* To print the packages a given package depends on:
* apt-cache depends package
* To print detailed information of the versions available for a package and the packages that reverse-depends on it:
* apt-cache showpkg package

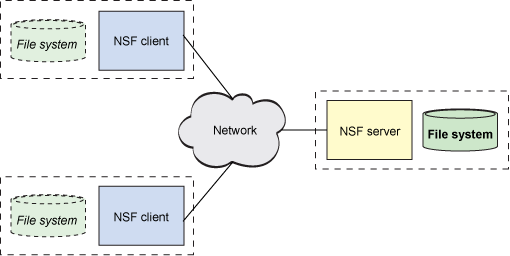
DPKG PACKAGE MANAGEMENT

This is the main package management program. dpkg can be invoked with many options. Some common uses are:

* Find out all the options: dpkg --help.
* Print out the control file (and other information) for a specified package: dpkg --info foo\_VVV-RRR.deb
* Install a package (including unpacking and configuring) onto the file system of the hard disk: dpkg --install foo\_VVV-RRR.deb.
* Unpack (but do not configure) a Debian archive into the file system of the hard disk: dpkg --unpack foo\_VVV-RRR.deb. Note that this operation does *not* necessarily leave the package in a usable state; some files may need further customization to run properly. This command removes any already-installed version of the program and runs the preinst script associated with the package.
* Configure a package that already has been unpacked: dpkg --configure foo. Among other things, this action runs the postinst script associated with the package. It also updates the files listed in the conffiles for this package. Notice that the 'configure' operation takes as its argument a package name (e.g., foo), *not* the name of a Debian archive file (e.g., foo\_VVV-RRR.deb).
* Extract a single file named "blurf" (or a group of files named "blurf\*" from a Debian archive: dpkg --fsys-tarfile foo\_VVV-RRR.deb | tar -xf - 'blurf\*'
* Remove a package (but not its configuration files): dpkg --remove foo.
* Remove a package (including its configuration files): dpkg --purge foo.
* List the installation status of packages containing the string (or regular expression) "foo\*": dpkg --list 'foo\*'.

**NFS (Network File System ) Server**

NFS Network File System is a server-client protocol used for sharing files on Unix, Linux systems. It allows sharing files from a central server allowing several users to access and modify the same files from different clients making all the changes on the files visible on all clients.



NFS Server

In order to configure a node as NFS server the packages 'nfs-utils' and 'rpcbind' must be installed and running :

Port NO - 2049 (for nfs) - 111 port map

Package- nfs-utils

**Configuration and status files**  
\*/etc/exports  
\*/var/lib/nfs/rmtab  
\*/var/lib/nfs/xtab  
\*/etc/hosts.allow  
\*/etc/hosts.deny

**Daemons**  
\*rpc.portmap  
\*rpc.mountd  
\*rpc.nfsd  
\*rpc.statd  
\*rpc.lockd  
\*rpc.rquotad

**Scripts and commands**  
\*/etc/rc.d/init.d/nfs  
\*nfstat  
\*showmount  
\*rpcinfo  
\*exportfs

**Sever Side**

**To install packages**

#rpm -ivh portmap-4.0-63.i386.rpm

#rpm -ivh nfs-utils-lib-devel-1.0.8.i386.rpm

Or

Yum install nfs\* -y

To create directory.

# mkdir /share

To export directory.

#vim /etc/exports

/share \*(rw,sync ,no\_root\_squash ,no\_all\_squash)

/share 192.168.0.0/24(ro,sync)

rw – Writable permission to shared folder

ro -- Read-only permission to shared folder

sync – Synchronize shared directory

no\_root\_squash – Enable root privilege

no\_all\_squash – Enable user’s authority

#exportfs -r (Re exports the nfs share)

#service nfs restart

#chkconfig nfs on

#chkconfig nfs --list

**For Verification on server**

#exportfs -v

#showmount -a

**On Clinet Side**

#yum install showmount rpcbind

**Create a Directory**

#mkdir -p /mnt/data

**For Checking on client side**

#Showmount -e 192.168.1.25

**Mounting**

**#mount -t nfs remote machine ip:share\_name   local\_mount\_point**

**#mount -t nfs   192.168.1.25:/share /mnt/data (Temporary)**

**For permanent**

#Vi /etc/fstab

192.168.1.25:/share /mnt/data nfs defaults 0 0

:wq

#mount -a

**Checking**

#cd /mnt/data

ll

or

ls

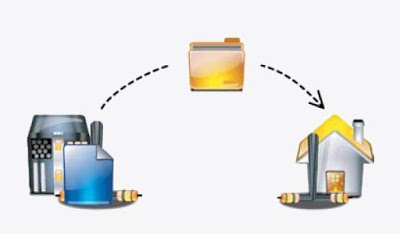
### FTP server

## Using the Very Secure FTP Server (vsFTPd)

The Very Secure FTP Server (vsFTPd) is the only FTP server software included in the Red Hat Linux distribution. Although it hasn't been around as long as WU-FTPD, vsFTPd is becoming the FTP server of choice for sites that need to support thousands of concurrent downloads. It was also designed to secure your systems against most common attacks.

Red Hat, Inc. itself uses vsFTPd on its own FTP servers (ftp.redhat.com). Other organizations in the Linux/GNU world have also made the switch to vsFTPd, including Debian Linux (ftp.debian.org) and the GNU Project (ftp.gnu.org).

Besides security and scalability, vsFTPd was designed for simplicity. Therefore, fewer options exist for configuring vsFTPd than you find in WU-FTPD, so you are expected to rely on standard Linux file and directory permissions to provide refined access to your server. Getting started with vsFTPd, or using it to replace WU-FTPD, is fairly straightforward.

[](http://1.bp.blogspot.com/-L8pr0Yf49QY/UjAmZVWW4zI/AAAAAAAAAjo/Dnxk9XTdR5A/s1600/ftp1.jpg)

|  |
| --- |
| **Description** |

|  |  |
| --- | --- |
| **Package** | vsftpd |
| **Port** | 20,21 |
| **Daemon\*** | vsftpd |
| **Scripts** | /etc/init.d/vsftpd |
| **Configuration File** | /etc/vsftpd/vsftpd.conf  /etc/vsftpd/ftpuser {users that deny}  /etc/vsftpd/ftp\_user {users that deny |

### Quick-starting vsFTPd

By enabling the vsFTPd service, you can almost instantly have an FTP service running with the default values (set in the /etc/vsftpd/vsftpd.conf file). The following is a quick procedure for getting your vsFTPd server up and running.

|  |  |  |
| --- | --- | --- |
|  | Note? | If you have been using the WU-FTP server on your computer and you are switching to vsFTPd, you need to turn off WU-FTP. To do that, change disable=no to disable=yes in the /etc/xinetd.d/wu-ftpd file. Then, once you have completed the following procedure, vsFTPd will take control of the default FTP configuration, allowing access to the /var/ftp directory and listening on the default FTP port. Because the two packages have some different default settings, however, you may want to do additional tuning to get vsFTPd to perform as you would like it to. |

1. To use the vsFTPd server, you must make sure that the vsFTPd software package is installed.
2. # **rpm -qa vsftpd** or yuminstall vsftpd -y
3. Enable the vsFTPd server by typing the following line (as root user):
4. # **chkconfig vsftpd on**
5. Start the vsFTPd server as follows:
6. # **/etc/init.d/vsftpd start**
7. Try to log in to the FTP server as anonymous (using any e-mail address as the password) (if you are connecting over a network, use the more-secure sftp command instead):
8. $ **ftp localhost**
9. Connected to yourhost
10. 220 (vsFTPd 1.2.0)
11. 530 Please login with USER and PASS
12. Name (localhost:chris): **anonymous**
13. 331 Please specify the password.
14. Pasword: **\*\*\*\*\*\***
15. 230 Login successful.
16. Remote system type is UNIX.
17. Using binary mode to transfer files.

ftp>

If you saw messages similar to the preceding, your vsFTPd server is now up and running. Next, try to access the server from another computer on the network to be sure that it is accessible.

|  |  |  |
| --- | --- | --- |
|  | Note? | If your FTP server is not accessible to the outside world, you may need to ensure that your network is configured properly and that your firewall allows access to port 21. Refer to Appendix C for information on getting your network services working. |

The next section explains the /etc/vsftpd/vsftpd.conf configuration file.

### Configuring vsFTPd

Most of the configuration of vsFTPd is done in the /etc/vsftpd/vsftpd.conf file. Although many values are not set explicitly in vsftpd.conf, you can override the defaults by setting *option*=*value* statements in this file. You can set such things as which users have access to your vsFTPd server, how logging is done, and how timeouts are set.

Go through the following section for more information about how vsFTPd is configured by default and how you can further configure your vsFTPd server.

#### User accounts

Users who can access your vsFTPd server are, by default, the anonymous user and any users with real-user accounts on your system. (A *guest* user is simply a real user account that is restricted to its own home directory.) The following lines set these user access features:

anonymous\_enable=YES

local\_enable=YES

The anonymous\_enable line lets users log in anonymously using either the anonymous or ftp user name. Any users with local accounts (in /etc/passwd) can log into the FTP server with local\_enable set to YES. An exception to this rule is that, by default, all user accounts listed in the /etc/vsftpd.user\_list file are denied access.

|  |  |  |
| --- | --- | --- |
|  | Note? | If you want to disable access by anonymous users, don't just comment out anonymous\_enable. Anonymous access is on by default, so you must set anonymous\_enable=NO to disable it. |

Check the vsftpd.user\_list file to see which users are denied access to the vsFTPd server. Note that root and other administrative logins are excluded. You can add other users to this list or change the location of the list by setting the userlist\_file parameter to the file you want. To add a user to the vsftpd.user\_list or use the userlist\_file parameter to create a new list, you must also have userlist\_enable set to YES (as it is by default). For example:

userlist\_file=/etc/vsftpd.user\_list\_local

userlist\_enable=YES

If you like, you can change the meaning of the /etc/vsftpd.user\_list file so that only the users in that list are allowed to use the vsFTPd service. Set userlist\_deny=NO and change the /etc/vsftpd.user\_list to include only names of users to whom you want to grant access to the server. (All other users, including anonymous and ftp, will be denied access.)

#### Setting FTP access

The vsFTPd server software provides a simple and seemingly secure approach to access permissions. Instead of using settings in the FTP service to selectively prevent downloads and uploads of particular directories (as WU-FTPD does), you can use standard Linux file and directory permissions to limit access. There are, however, the following general settings in the /etc/vsftpd/vsftpd.conf file to let users get files from and put files onto your vsFTPd server.

##### Downloading files

Any users with valid logins (anonymous or real users, excluding some administrative logins) can download files from the vsFTPd server, by default. The ability to download a particular file or a file from a particular directory is governed by the following basic Linux features:

* **File and directory permissions** **—** Standard file and directory permissions apply as a means of limiting access to particular files, even in accessible file systems. So, if the root user puts a file with 600 permission (read/write to root only) in the /var/ftp directory, an anonymous user is not able to download that file.
* **Root directory —** The root directory (chroot) for anonymous users is /var/ftp. The root directory for regular users is the entire computer's root directory (/), although their current directory after connecting to FTP is /home/*user*, where *user* is the user name. So an anonymous user is restricted to downloads from the /var/ftp directory structure, while a regular user potentially has access to the whole file system. Another possibility is to create *guest* accounts by restricting some or all users to their home directories.

You can use the chroot\_local\_user option to change the root directory for regular users so that they are restricted to their home directory. In general you will not want to do this, because using the same user name and password for general Linux logins doesn't place such restrictions on your users. To restrict all regular users to their home directory when using vsFTPd, add this line to the vsftpd.conf file:

chroot\_local\_user=YES

To enable the concept of *guest* users, you can choose to limit only selected users to their home directories. You do this by setting chroot\_list\_enable to YES, then adding a list of guest users to a file noted with the chroot\_list\_file option. The following example lets you add such a list (one user name per line) to the /etc/vsftpd.chroot\_list file:

chroot\_list\_enable=YES

chroot\_list\_file=/etc/vsftpd.chroot\_list

One setting you can add to the vsftpd.conf file can affect how files are downloaded. To enable ASCII downloads, you would have to enable that feature as follows:

ascii\_download\_enable=YES

Without making that change, all downloads are done in binary mode. Although vsFTPd will seem to allow the user to change to ascii mode, ascii mode will not work if this setting is NO.

##### Uploading (writing) files from local users

Two values set in the vsftpd.conf file allow the uploading of files during a vsFTPd session. These defaults allow any users with regular, local user accounts to upload files:

write\_enable=YES

local\_umask=022

The write\_enable value must be YES if you intend to allow any users the ability to write to the FTP server. The umask=022 value sets the default file permission used when a local user creates a file on the server. (The 022 value causes files created to have 644 permission, allowing the user read and write permission and everyone else only read permission.)

As with downloading, uploading in ascii mode is prohibited by default. Though ascii downloads create a potential security hole for draining resources from your server, ascii uploads are apparently not as dangerous and can be useful for uploading text files. To allow ascii uploads, add the following line:

ascii\_upload\_enable=YES

##### Uploading (writing) files from anonymous users

The ability to upload files is turned off for anonymous FTP users. If you want to turn it on, add the following line to the vsftpd.conf file:

anon\_upload\_enable=YES

You must also make sure that the /var/ftp directory contains one or more directories with write permissions open to anonymous users. For example, you might want to create an incoming directory and open its permissions (chmod 777 /var/ftp/incoming).

Files uploaded by anonymous users will be created with 600 permission by default (read/write for the ftp user, not accessible to any other users so that even the user who uploaded the file can't remove it). To allow 644 permission, for example, you can add the following line:

anon\_umask=077

Once you allow the anonymous user to upload files, you can grant limited ability to change the files he or she uploads. By adding the following line, you can allow anonymous users to rename or delete any files owned by anonymous users (providing the files are in directories for which the users have write permission):

anon\_other\_write\_enable=YES

If you also want to allow anonymous users to create their own directories, add the following:

anon\_mkdir\_write\_enable=YES

By default, the ftp user is given ownership of uploaded files from anonymous users. If you want to indicate that anonymous uploads be owned by a different user (of your choice), you can use the chown\_uploads and chown\_username options. For example, if you have a user account named mynewuser, you can set these options as follows:

chown\_uploads=YES

chown\_username=mynewuser

Of course, you can create and use any user name you want. You should not, however, use the root login or any other administrative login for this purpose (for security reasons).

#### Adding message files

Although vsFTPd doesn't support the arrangement of README and welcome files the WU-FTP supports, you can add .message files to any accessible directory on your vsFTPd server. Then, if you use the default dirmessage\_enable option as follows, the text from the .message file will be displayed when the user enters the directory:

dirmessage\_enable=YES

You will probably at least want to add a .message file to the root directory of the FTP server for anonymous users. By default, that location is /var/ftp/.message. If you want to use files other than .message files, you can set the message\_file option. For example, to have text from the .mymessage file displayed when you enter a directory, you can add the following line:

message\_file=.mymessage

You can also set a one-line message to appear before the login prompt. You can do this by entering the following line, replacing the text with anything you want to say:

ftpd\_banner=Welcome to My FTP service.

#### Logging vsFTPd activities

Logging is enabled in vsFTPd by default, and the activities of your vsFTPd site are written to the /var/log/vsftpd.log file. The following options enable logging in that way:

xferlog\_enable=YES

xferlog\_file=/var/log/vsftpd.log

You can turn off logging if you like (not recommended, since logging allows you to watch for potential break-ins) by changing YES to NO. Or you can change the location of the log file by changing the value of the xferlog\_file option. The log file is output in standard format (that is, the same way WU-FTPD does it), based on the following option:

xferlog\_std\_format=YES

#### Setting timeouts

The following timeouts are set by default in vsFTPd (these values are built in, so you don't have to make any changes to the /etc/vsftpd/vsftpd.conf file for them to take effect):

accept\_timeout=60

connect\_timeout=60

idle\_session\_timeout=600

data\_connection\_timeout=120

The accept\_timeout=60 and connect\_timeout=60 values determine how long the client has to establish a PASV or PORT style connection, respectively, before the connection times out. Both are set to 60 seconds. The idle\_session\_timeout=600 option causes the FTP session to be dropped if the user has been inactive for more than 10 minutes (600 seconds). The data\_connection\_timeout value sets the amount of time, during which no progress occurs, that the server will wait before dropping the connection (the default here is 120 seconds, or two minutes).

#### Navigating the vsFTPd site

In general, vsFTPd does not offer as many shortcuts for clients trying to navigate through the FTP site as does WU-FTPD. For example, most shell wildcard characters that a user might expect to use, such as question marks and brackets, are not supported.

There is, however, one very useful wildcard character you can use with the ls command, and one option you can turn. The asterisk (\*) wildcard can be used with the ls command. Multiple asterisks in the same line are supported. You can add support for the –R option of ls so that a user can recursively list the contents of the current directory and all subdirectories. To turn on this feature, which is off by default, you can add the following line to the vsftpd.conf file:

ls\_recurse\_enable=YES

File Transfer from windows to linux or linux to windows we can use the Filezilla

application.

**Example:**

**Server Side-**

**1.** Install vsftpd Package..

#yum install vsftpd\* -y

**2.** Create a directory

# mkdir /var/ftp/Data       {Data is a Directory Name}

**3.**Create some file in this Directory /var/ftp/Data/

for Ex-

#cd /var/ftp/Data

#touch a1{1..10}

**4.** Restart The Services

#/etc/init.d/vsftpd restart

**Client Side-**

**1.** Install the ftp package

#yum install ftp\* -y

**2.**Access the ftp

#ftp 192.168.0.5           {192.168.0.5- server IP}

user: anonymous

password: anonymous        {Note-By Defaults Ftp user name and password - anonymous }

ftp>!ls                                 {List all file  !ls for Remote machine, ls for local machine}

ftp>cd Data                        {change directory}

ftp>!ls

ftp>get a1                           {To copy one file at a time from the remote ftp server to the local system}

ftp>bye                               {Exit the FTP session }

**Server Side-**

* **Disable anonymous  authentication-**

1. Open configuration file..  
#vi /etc/vsftpd/vsftpd.conf

anonymous\_enable=YES       {yes=anonymous enable,no=Disable}

Save and exit.  
  
2. Restart service..  
#/etc/init.d/vsftpd restart

* **Anonymous upload-**(Obviously need to create a directory writable by the FTP user)

1. Open configuration file..  
#vi /etc/vsftpd/vsftpd.conf  
  
#anon\_upload\_enable=YES   (remove the comment line no 27}

Save and exit.  
  
2. Restart service..  
#/etc/init.d/vsftpd restart

**FTP UPLOAD**

1. mkdir /var/ftp/upload

2. chmod 1777 /var/ftp/upload

3. vi /etc/vsftpd/vsftpd.conf

anon\_upload\_enable=YES

:wq

service vsftpd restart

chkconfig vsftpd on

ftp 192.168.0.5           {192.168.0.5- server IP}

user: anonymous

password: anonymous       {Note-By Defaults Ftp user name and password - anonymous }

ftp>!ls                                 {List all file  !ls for Remote machine, ls for local machine}

ftp>cd upload                       {change directory}

ftp>!ls

ftp>put a1      {To upload one file at a time from the remote ftp server to the local system}

* **Disable and enable selinux**

**Note-**  
**Selinux-** (Security-Enhanced Linux)It is developed by NSA (National Security Agency).Selinux is a extra layer of the firewall. and it has define different-2 mode.                       
                                          
**Getsebool-** Getsebool  reports  where  a  particular SELinux boolean.                                  
                           -a     Show all SELinux booleans.  
  
**Setsebool- S**etsebool  sets  the current state of a particular SELinux  boolean

                           -p option is given, all pending values are written to the policy file on disk.

# getsebool -a | grep ftp

#  setsebool -P allow\_ftpd\_anon\_write on

# sesebool -P  allow\_ftpd\_full\_access on

# chmod 777 /var/ftp/pub

# /etc/init.d/vsftpd restart

**Client Side-**  
# ftp  192.168.0.10  
user  anonymous  
password anonymous  
ftp> ls  
ftp> cd pub  
ftp>ls  
ftp>lcd /var/ftp/                     {Change current directory}  
  
ftp> put a1 /Data/fiel1           {Upload lfile with remote name rfile}  
  
**Server Side-**

* **Create ftp user and password-**

**1.** Create user

#adduser -c 'FTP USER TOM' -m tom

#passwd tom

**2.**Open configuration file..

#vi /etc/vsftpd/vsftpd.conf

#anonymous\_enable=YES

                                   {Uncomment this to allow local users to log in.}

**3.** Restart service..

#/etc/init.d/vsftpd restart

DNS SERVER

A DNS server, or name server, is used to resolve an IP address to a hostname or vice versa.  
You can set up four different types of DNS servers:

* A **master DNS server for your domain(s),** which stores authoritative records for your domain.
* A **slave DNS server,** which relies on a master DNS server for data.
* A **caching-only DNS server,** which stores recent requests like a proxy server. It otherwise refers to other DNS servers.
* A **forwarding-only DNS server,** which refers all requests to other DNS servers.

Before configuring BIND to create a DNS server, you must understand some basic DNS concepts.

The entire hostname with its domain such as ***server.example.com*** is called a fully qualified domain name (FQDN). The right-most part of the FQDN such as .com or .net is called the ***top level domain,*** with the remaining parts of the FQDN, which are separated by periods, being sub-domains.

These sub-domains are used to divide FQDNs into zones, with the DNS information for each zone being maintained by at least one ***authoritative name server.***

The authoritative server that contains the master zone file, which can be modified to update DNS information about the zone, is called the ***primary master server,*** or just ***master server.***

The additional name servers for the zone are called ***secondary servers*** or ***slave servers.*** Secondary servers retrieve information about the zone through a zone transfer from the master server or from another secondary server. DNS information about a zone is never modified directly on the secondary server

### chroot features

chroot feature is run named as user **named**, and it also limit the files named can see. When installed, **named** is fooled into thinking that the directory **/var/named/chroot** is actually the **root or /** directory. Therefore, named files normally found in the **/etc** directory are found in **/var/named/chroot/etc** directory instead, and those you would expect to find in **/var/named** are actually located in **/var/named/chroot/var/named.**

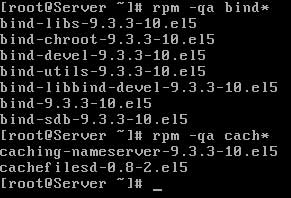
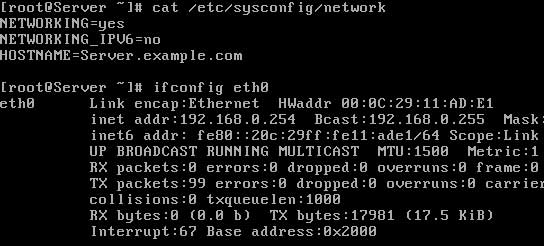
The advantage of the chroot feature is that if a hacker enters your system via a BIND exploit, the hacker's access to the rest of your system is isolated to the files under the chroot directory and nothing else. This type of security is also known as a chroot jail.

### Configure dns server

In this example we will configure a dns server and will test from client side.

For this example we are using three systems one linux server one linux clients and one window clients.

**bind** and **caching-nameserver** rpm is required to configure dns. check them for install if not found install them.

  
set hostname to **server.example.com** and ip address to **192.168.0.254**  


main configuration file for dns server is **named.conf.** By default this file is not created in **/var/named/chroot/etc/** directory. Instead of named.conf a sample file **/var/named/chroot/etc/named.caching-nameserver.conf** is created. This file is use to make a caching only name server. You can also do editing in this file after changing its name to **named.conf** to configure master dns server or you can manually create a new **named.conf** file.

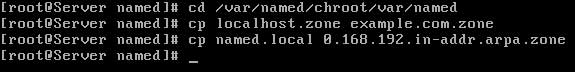
In our example we are creating a new named.conf file   
vi named

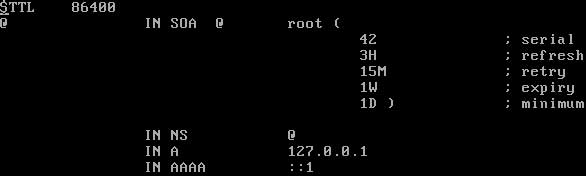
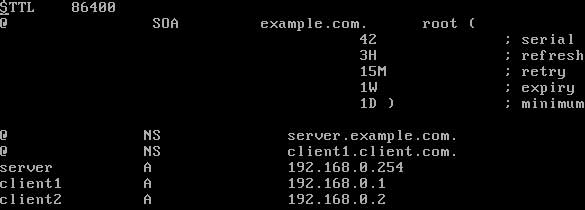
We are using bind's **chroot** features so all our necessary files will be located in chroot directory. Set directory location to **/var/named.** Further we will set the location of **forward zone** and **reverse lookup zone** files. If you cannot create this file manually then download this file and copy to **/var/named/chroot/etc/**

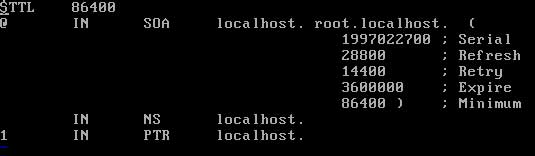
Or do editing exactly as shown here in image   
  
save this file with **:wq** and exit

### Configure zone file

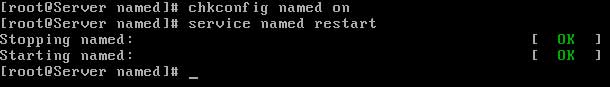
We have defined two zone files **example.com.zone** for forward zone and **0.168.192.in-addr.arpa** for reverse zone. These files will be store in **/var/named/chroot/var/named/** location. We will use two sample files for creating these files.

Change directory to **/var/named/chroot/var/named** and copy the sample files to name which we have set in named.conf  


Now open forward zone file **example.com.zone**  
example.com.zone  
By default this file will look like this   
  
Change this file exactly as shown in image below  
  
If you feel difficulty to modify this file then download this configured file and copy to **/var/named/chroot/var/named**

Now open reverse lookup zone file **0.168.192.in-addr.arpa**   
reverse lookup zone  
By default this file will look like this   
  
Change this file exactly as shown in image below  
  
If you feel difficulty to modify this file then download this configured file and copy to **/var/named/chroot/var/named**

Now changed the ownership of these zone files to **named** group   
chgrp

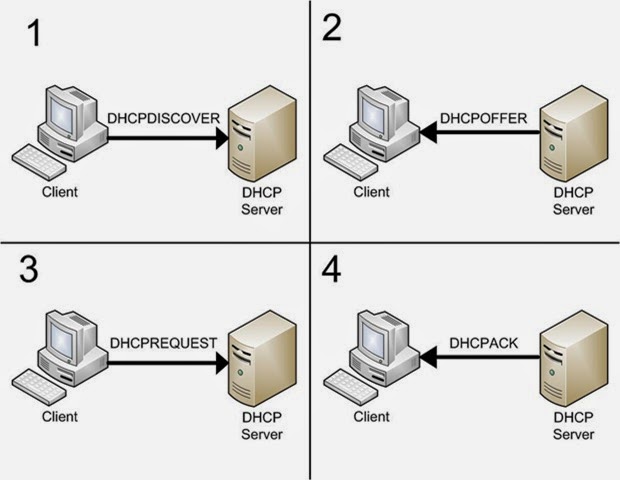
Now start the named service   


If service restart without any error means you have successfully configured master name server in our next article we will learn how to configure salve dns server and test it.

**DHCP Server**

DHCP (BOOTP) Server

Dynamic Host Configuration Protocol (DHCP) is a network protocol that automatically assigns TCP/IP information to client machines. Each DHCP client connects to the centrally located DHCP server, which returns the network configuration (including the IP address, gateway, and DNS servers) of that client.

[](http://1.bp.blogspot.com/-MJDbgI4NTsU/U7zRXbwMg6I/AAAAAAAAH6A/DHURbvg72h8/s1600/DHCP_DORA.jpg)

  \*\* DHCP uses DORA process to release the ip adresses to clients. ( Ref.Above image )

  \*\* DHCP server works with port number "67".

DHCPD follows the DORA process:  
D- Discovery (Client)  
O- Offer (Server)  
R- Request (Client)  
A- Acknowledgement (Server)

**Features:-**  
  
  
**1-** Dynamic Host Configuration Protocol (DHCP) is a network protocol that automatically assigns TCP/IP information to client machines.    
**2-**  Each DHCP client connects to the centrally located DHCP server, which returns the network configuration (including the IP address, gateway, and DNS servers) of that client.   
**3-** DHCP is also useful if you want to change the IP addresses of a large number of systems.  
**4-** Includes all sorts of setting: IPv4, IPv6, DNS, NTP, NIS, Etc.  
**5-** DHCP is an UDP application (UDP:67)  
  
**Package-**      dhcp  
**Port-**            67  
**Daemon-**       dhcpd  
**Script-**          /etc/init.d/dhcpd  
**Conf File-**     /etc/dhcp/dhcpd.conf

Why Use DHCP?

DHCP is useful for automatic configuration of client network interfaces. When configuring the client system, you can choose DHCP instead of specifying an IP address, netmask, gateway, or DNS servers. The client retrieves this information from the DHCP server. DHCP is also useful if you want to change the IP addresses of a large number of systems. Instead of reconfiguring all the systems, you can just edit one configuration file on the server for the new set of IP addresses. If the DNS servers for an organization changes, the changes happen on the DHCP server, not on the DHCP clients. When you restart the network or reboot the clients, the changes go into effect.

If an organization has a functional DHCP server correctly connected to a network, laptops and other mobile computer users can move these devices from office to office.

**Configure DHCP Server-**  
  
**Step-1** Set Static IP address in dhcp server  
  
 **# vim /etc/sysconfig/network-scripts/ifcfg-eth0**  
  
IP Address- 192.168.0.254  
NetMask-    255.255.255.0  
Broadcast Address- 255.255.0.255  
  
 **# ifconfig eth0**  
  
**Step-2** Install dhcp Package  
  
**# yum install dhcp -y**  
      
**Step-3** Check dhcp Documentation File  
  
 **# rpm -ql dhcp**  
  
/etc/dhcp                  (Container for DHCPD Configuration)  
/etc/dhcp/dhcpd.conf            (IPv4 Config)  
/etc/dhcp/dhcpd6.conf           (IPv4 Config)  
/var/lib/dhcpd                        (Container for leases)  
/var/lib/dhcpd/dhcpd.leases   (IPv4 leases)  
/var/lib/dhcpd/dhcpd6.leases (IPv6 leases)  
  
**Step-4** Configure '/etc/dhcp/dhcpd.conf' file  
  
 **# cd /etc/dhcp/  
 # ls**  
  
dhclient.d  dhcpd6.conf  dhcpd.conf  
 **# vim /etc/dhcp/dhcpd.conf**

**Now Copy '/dhcpd.conf.sample' file to '/etc/dhcp/dhcpd.conf'  
  
 # cp -rvf /usr/share/doc/dhcp-4.1.1/dhcpd.conf.sample /etc/dhcp/dhcpd.conf  
  
Again open '/etc/dhcp/dhcpd.conf' file  
  
 # vim /etc/dhcp/dhcpd.conf              
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
# option definitions common to all supported networks...  
option domain-name "ashu.com"; ##--> Change Domain Name  
option domain-name-servers  server.ashu.com;##--> Change Domain Name  Server  
  
# define lease line  
default-lease-time 600;  
max-lease-time 7200;  
  
  
# If this DHCP server is the official DHCP server for the local  
# network, the authoritative directive should be uncommented.  
authoritative;                ##---> Uncoment  this line  
  
# Use this to send dhcp log messages to a different log file (you also  
# have to hack syslog.conf to complete the redirection).  
   
log-facility local6;                        
  
##--> Note- After DHCPD to log using a different Facility: i.e. 'local6' Because boot message are logged via: 'local7'  
##--> Checklog file' # vim /etc/rssyslog.conf   
##--> and mentation 'local6.\*   /var/log/dhcpd.log'   
##--> Save boot messages also to boot.log  
##--> local7.\*           /var/log/boot.log  
##--> local6.\*           /var/log/dhcpd.log  
  
  
#This is a very basic subnet declaration.  
    
subnet 192.168.0.0 netmask 255.255.255.0 { ##---> Define Your Subnen mask and netmak,  
range 192.168.0.10  192.168.0.50;   ##---> Define dhcp provide ip Range   
#option routers rtr-239-0-1.example.org, rtr-239-0-2.example.org;  
}  
  
##--> Note- And Coment all line..  
#  
#  
#  
#  
#  
  
:wq!**

**Step-5 Restart dhcp Service  
  
 # /etc/init.d/dhcpd restart;chkconfig dhcpd on  
  
or # service dhcpd restart  
 # chkconfig dhcpd on  
  
Step-6 Check Dhcpd Status  
  
 # service dhcpd status  
 # chkconfig --list dhcpd  
  
dhcpd   0:off    1:off    2:on    3:on    4:on    5:on    6:off  
DHCP use to assign ip address automaticaly in the netwok system.**

# cat /etc/dhcp/dhcpd.conf

The sample configuration file can be found at /usr/share/doc/dhcp-<version>/dhcpd.conf.sample. You should use this file to help you configure /etc/dhcp/dhcpd.conf

               cp /usr/share/doc/dhcp-<version\_number>/dhcpd.conf.sample /etc/dhcp/dhcpd.conf

 \*\* DHCP also uses the file /var/lib/dhcpd/dhcpd.leases to store the client lease database.

Range parameter:

To configure a DHCP server that leases a dynamic IP address to a system within a subnet, modify Below Example, with your values. It declares a default lease time, maximum lease time, and network configuration values for the clients. This example assigns IP addresses in the range 192.168.1.10 and 192.168.1.100 to client systems.

Example,

"

Range parameter

"

default-lease-time 600;

max-lease-time 7200;

option subnet-mask 255.255.255.0;

option broadcast-address 192.168.1.255;

option routers 192.168.1.254;

option domain-name-servers 192.168.1.1, 192.168.1.2;

option domain-search "[example.com](http://example.com/)";

subnet 192.168.1.0 netmask 255.255.255.0 {

   range 192.168.1.10 192.168.1.100;

}

Static IP address using DHCP:

To assign an IP address to a client based on the MAC address of the network interface card, use the hardware ethernet parameter within a host declaration. As demonstrated in Below Example, the host ftp declaration specifies that the network interface card with the MAC address 00:A0:78:8E:9E:AA always receives the IP address 192.168.1.4.

Note that you can also use the optional parameter host-name to assign a host name to the client.

Example.

"

Static IP address using DHCP

"

host ftp {

   option host-name "[ftp.example.com](http://ftp.example.com/)";

   hardware ethernet 00:A0:78:8E:9E:AA;

   fixed-address 192.168.1.4;

}

Definitions:

default\_lease\_time [seconds]—This option defines the length of time, in seconds, for an IP address lease if the client does not request a specific lease length.

max\_lease\_time [seconds]—This option defines the maximum length, in seconds, of a lease length. This is the maximum lease length a client may receive regardless of what it requests.

option domain\_name ["domain"]—Defines the domain name.

option domain\_name\_servers [address\_list]—Lists the addresses of the DNS name servers.

option ntp\_servers [address\_list]—Lists the addresses of the NTP (Network Time Protocol) servers the client is to use.

option routers [address\_list]—Defines the default router.

option subnet\_mask [mask]—Defines the subnet mask. If this option is undefined, the network mask from the subnet statement is used.

# chkconfig  dhcpd  on

# service  dhcpd status

**Configuring a DHCP Client**

**:**

vi

 /etc/sysconfig/network-scripts/ifcfg-eth0

DEVICE=eth0

BOOTPROTO=dhcp

ONBOOT=yes

 save an exit

# service network restart

#ifconfig eth0

 #ifup eth0

**How to Bind Client MAC Address or Provide Particular IP to Client PC...**  
  
**# vim /etc/dhcp/dhcpd.conf**

# Fixed IP addresses can also be specified for hosts. These addresses  
# should not also be listed as being available for dynamic assignment.  
# Hosts for which fixed IP addresses have been specified can boot using  
# BOOTP or DHCP.   Hosts for which no fixed address is specified can only  
# be booted with DHCP, unless there is an address range on the subnet  
# to which a BOOTP client is connected which has the dynamic-bootp flag  
# set.  
host ashu {                                           ##--> Host Nmae  
  hardware ethernet 00:0c:29:5d:1e:2c; ##--> Client PC Mac  
  fixed-address 192.168.0.20;               ##--> Defint IP  
}   
  
:wq

**apache (web sever)**

Webserver  to host the websites. We can host multiple websites on same webserver. Apache is probably the most popular Linux-based Web server application in use. Once you have DNS correctly setup and your server has access to the Internet, you'll need to configure Apache to accept surfers wanting to access your Web site.   
  ex: apache, tomcat, nginx, lighttpd and IIS

Apache is the default webserver in RHEL.

http (apache):  
**package :** httpd  
**service :** httpd  
**config file** : /etc/httpd/httpd.conf  
**logs:** /var/log/httpd  
**port no**: 80  
**Default Document Root** :  /var/www/html    ( dir where the website content available)  
**Default DirectoryIndex** : index.html (home page for that website from that documentroot)  
  
**Install apache**

yum  install  httpd

**Verification**

rpm  -qa   httpd

**Service on**

chkconfig httpd on  
service httpd   restart

or

/etc/init.d/httpd restart

**General Configuration Steps**

The configuration file used by Apache is /etc/httpd/conf/httpd.conf in Redhat / Fedora distributions and /etc/apache\*/httpd.conf in Debian / Ubuntu distributions. As for most Linux applications, you must restart Apache before changes to this configuration file take effect.

**Where To Put Your Web Pages**

All the statements that define the features of each web site are grouped together inside their own <VirtualHost> section, or container, in the httpd.conf file. The most commonly used statements, or directives, inside a <VirtualHost> container are:

* **servername**: Defines the name of the website managed by the <VirtualHost> container. This is needed in named virtual hosting only, as I'll explain soon.
* **DocumentRoot**: Defines the directory in which the web pages for the site can be found.

By default, Apache searches the DocumentRoot directory for an index, or home, page named index.html. So for example, if you have a servername of www.my-site.com with a DocumentRoot directory of /home/www/site1/, Apache displays the contents of the file /home/www/site1/index.html when you enter http://www.site1.com in your browser.

Some editors, such as Microsoft FrontPage, create files with an .htm extension, not .html. This isn't usually a problem if all your HTML files have hyperlinks pointing to files ending in .htm as FrontPage does. The problem occurs with Apache not recognizing the topmost index.htm page. The easiest solution is to create a symbolic link (known as a shortcut to Windows users) called index.html pointing to the file index.htm. This then enables you to edit or copy the file index.htm with index.html being updated automatically. You'll almost never have to worry about index.html and Apache again!

This example creates a symbolic link to index.html in the /home/www/site1 directory.

[root@linux tmp]# cd /home/www/site1

[root@linux site1]# ln -s index.htm index.html

[root@linux site1]# ll index.\*

-rw-rw-r-- 1 root root 48590 Jun 18 23:43 index.htm

lrwxrwxrwx 1 root root 9 Jun 21 18:05 index.html -> index.htm

[root@linux site1]#

The l at the very beginning of the index.html entry signifies a link and the -> the link target.

**The Default File Location**

By default, Apache expects to find all its web page files in the /var/www/html/ directory with a generic DocumentRoot statement at the beginning of httpd.conf. The examples in this chapter use the /home/www directory to illustrate how you can place them in other locations successfully.

**File Permissions And Apache**

Apache will display Web page files as long as they are world readable. You have to make sure you make all the files and subdirectories in your DocumentRoot have the correct permissions.

It is a good idea to have the files owned by a nonprivileged user so that Web developers can update the files using FTP or SCP without requiring the root password.

To do this:

1.     Create a user with a home directory of /home/www.

2.     Recursively change the file ownership permissions of the /home/www directory and all its subdirectories.

3.     Change the permissions on the /home/www directory to 755, which allows all users, including the Apache's httpd daemon, to read the files inside.

[root@linux tmp]# useradd -g users www

[root@linux tmp]# chown -R www:users /home/www

[root@linux tmp]# chmod 755 /home/www

Now we test for the new ownership with the ll command.

[root@linux tmp]# ll /home/www/site1/index.\*

-rw-rw-r-- 1 www users 48590 Jun 25 23:43 index.htm

lrwxrwxrwx 1 www users 9 Jun 25 18:05 index.html -> index.htm

[root@linux tmp]#

**Note:** Be sure to FTP or SCP new files to your web server as this new user. This will make all the transferred files automatically have the correct ownership.

If you browse your Web site after configuring Apache and get a "403 Forbidden" permissions-related error on your screen, then your files or directories under your DocumentRoot most likely have incorrect permissions. Appendix II, "Codes, Scripts, and Configurations," has a short script that you can use to recursively set the file permissions in a directory to match those expected by Apache. You may also have to use the Directory directive to make Apache serve the pages once the file permissions have been correctly set. If you have your files in the default /home/www directory then this second step becomes unnecessary.

we can host the websites in three ways,  
  
1. Ipbased ( we can resolve the website with both name and ip )   
2. Name based ( we need to resolve with name only )  
3. port based for name based hosting in the config file,  NameVirtualHost ip:80 ( check line number 990)

**Named Virtual Hosting**

You can make your Web server host more than one site per IP address by using Apache's named virtual hosting feature. You use the NameVirtualHost directive in the /etc/httpd/conf/httpd.conf file to tell Apache which IP addresses will participate in this feature.

The <VirtualHost> containers in the file then tell Apache where it should look for the Web pages used on each Web site. You must specify the IP address for which each <VirtualHost> container applies.

**Named Virtual Hosting Example**

Consider an example in which the server is configured to provide content on 172.16.15.20. In the code that follows, notice that within each <VirtualHost> container you specify the primary Web site domain name for that IP address with the ServerName directive. The DocumentRoot directive defines the directory that contains the index page for that site.

You can also list secondary domain names that will serve the same content as the primary ServerName using the ServerAlias directive.

Apache searches for a perfect match of NameVirtualHost, <VirtualHost>, and ServerName when making a decision as to which content to send to the remote user's Web browser. If there is no match, then Apache uses the first <VirtualHost> in the list that matches the target IP address of the request.

This is why the first <VirtualHost> statement contains an asterisk: to indicate it should be used for all other Web queries.

NameVirtualHost 172.16.15.20:80

<VirtualHost ip:80>  
    
   DocumentRoot   <dir for wesite content>  
   ServerName   <website name>  
   DirectoryIndex  <homepage>  
    
</VirtualHost>

<VirtualHost 172.16.15.20:80>

DocumentRoot /var/www/html/site1

servername www.site1.com

DirectoryIndex index.hmtl

</VirtualHost>

<VirtualHost 172.16.15.20:80>

DocumentRoot /var/www/html/site2

servername www.site2.com

DirectoryIndex index.hmtl

</VirtualHost>

Be careful with using the asterisk in other containers. A <VirtualHost> with a specific IP address always gets higher priority than a <VirtualHost> statement with an \* intended to cover the same IP address, even if the ServerName directive doesn't match. To get consistent results, try to limit the use of your <VirtualHost \*> statements to the beginning of the list to cover any other IP addresses your server may have.

You can also have multiple NameVirtualHost directives, each with a single IP address, in cases where your Web server has more than one IP address.

**IP-Based Virtual Hosting**

The other virtual hosting option is to have one IP address per Web site, which is also known as IP-based virtual hosting. In this case, you will not have a NameVirtualHost directive for the IP address, and you must only have a single <VirtualHost> container per IP address.Also, because there is only one Web site per IP address, the ServerName directive isn't needed in each <VirtualHost> container, unlike in named virtual hosting.

**IP Virtual Hosting Example: Single Wild Card**

In this example, Apache listens on all interfaces, but gives the same content. Apache displays the content in the first <VirtualHost \*> directive even if you add another right after it. Apache also seems to enforce the single <VirtualHost> container per IP address requirement by ignoring any ServerName directives you may use inside it.

<VirtualHost \*>

DocumentRoot /home/www/site1

</VirtualHost>

**IP Virtual Hosting Example: Wild Card and IP addresses**

In this example, Apache listens on all interfaces, but gives different content for addresses 172.16.15.26 and 172.16.15.27. Web surfers get the site1 content if they try to access the web server on any of its other IP addresses.

<VirtualHost \*>

DocumentRoot /home/www/site1

</VirtualHost>

<VirtualHost 97.158.253.26>

DocumentRoot /home/www/site2

</VirtualHost>

<VirtualHost 97.158.253.27>

DocumentRoot /home/www/site3

</VirtualHost>

**Configuration - Multiple Sites And IP Addresses**

To help you better understand the edits needed to configure the /etc/httpd/conf/httpd.conf file, I'll walk you through an example scenario.

Example

ServerName localhost

NameVirtualHost 172.16.15.26

NameVirtualHost 172.16.15.27

#

# Match a webpage directory with each website

#

<VirtualHost \*>

DocumentRoot /home/www/site1

</VirtualHost>

<VirtualHost 172.16.15.26>

DocumentRoot /home/www/site2

ServerName www.my-site.com

ServerAlias my-site.com, www.my-cool-site.com

</VirtualHost>

<VirtualHost 172.16.15.27>

DocumentRoot /home/www/site3

ServerName www.test-site.com

</VirtualHost>

**port based hosting**  
  
  Listen  8080 ( configuration file line no 990 )  
  
<VirtualHost ip:8080>     
   DocumentRoot   ....  
   ServerName   ....  
   DirectoryIndex  ....  
</VirtualHost>

**Troubleshooting Apache**

Troubleshooting a basic Apache configuration is fairly straightforward; you'll find errors in the /var/log/httpd/error\_log file during normal operation or displayed on the screen when Apache starts up. Most of the errors you'll encounter will probably be related to incompatible syntax in the <VirtualHosts> statement caused by typing errors.

**Testing Basic HTTP Connectivity**

The very first step is to determine whether your web server is accessible on TCP port 80 (HTTP).Lack of connectivity could be caused by a firewall with incorrect permit, NAT, or port forwarding rules to your Web server. Other sources of failure include Apache not being started at all, the server being down, or network-related failures.

If you can connect on port 80 but no pages are being served, then the problem is usually due to a bad Web application, not the Web server software itself.

It is best to test this from both inside your network and from the Internet.

**SSH (Secure Shell )Server**

**What Is SSH?**

There are a couple of ways that you can access a shell (command line) remotely on most Linux/Unix systems. One of the older ways is to use the telnet program, which is available on most network capable operating systems. Accessing a shell account through the telnet method though poses a danger in that everything that you send or receive over that telnet session is visible in plain text on your local network, and the local network of the machine you are connecting to. So anyone who can "sniff" the connection in-between can see your username, password, email that you read, and commands that you run. For these reasons you need a more sophisticated program than telnet to connect to a remote host.

An unencrypted telnet session SSH, which is an acronym for Secure Shell, was designed and created to provide the best security when accessing another computer remotely. Not only does it encrypt the session, it also provides better authentication facilities, as well as features like secure file transfer, X session forwarding, port forwarding and more so that you can increase the security of other protocols. It can use different forms of encryption ranging anywhere from 512 bit on up to as high as 32768 bits and includes ciphers like AES (Advanced Encryption Scheme), Triple DES, Blowfish, CAST128 or Arcfour. Of course, the higher the bits, the longer it will take to generate and use keys as well as the longer it will take to pass data over the connection.

An encrypted ssh session

These two diagrams on the left show how a telnet session can be viewed by anyone on the network by using a sniffing program like Ethereal (now called Wireshark) or tcpdump. It is really rather trivial to do this and so anyone on the network can steal your passwords and other information. The first diagram shows user jsmith logging in to a remote server through a telnet connection. He types his username jsmith and password C0lts06!, which are viewable by anyone who is using the same networks that he is using.

The second diagram shows how the data in an encrypted connection like SSH is encrypted on the network and so cannot be read by anyone who doesn't have the session-negotiated keys, which is just a fancy way of saying the data is scrambled. The server still can read the information, but only after negotiating the encrypted session with the client.

**Package-**                 openssh   
**Port-**                        22   
**Daemon-**                 sshd   
**Script-**                    /etc/init.d/sshd   
**Configuration file-**/etc/ssh/sshd\_config

**Private keys**   
  
Private key must be secure and used only by you to decrypt messages encrypted with you public key. Secure SSH encrypted communications are based on keeping the private key secure.   
  
**Public keys**   
  
Public key is publicly available. The recipient of your messages will encrypt the data with your public key that previously you have send. Only you using your private key will be able to decrypt that message.

**SSH Tools**

These are the most basic SSH tools than a Linux user must be aware.   
  
**sshd**   
  
The daemon service that implements the ssh server. By default it must be listening on port 22 TCP/IP.   
  
**ssh**   
  
The Secure Shell command ssh is a secure way to log and execute commands in to a remote machine using the private/public key encryption method replacing the insecure tools traditionally used for it: telnet, rlogin, rexec, rsh, etc.   
  
**scp**   
  
The Secure Copy command is a secure way to transfer files between computers using the private/public key encryption method replacing the insecure tool traditionally used for it: ftp.   
  
**ssh-keygen**   
  
This utility is used to create the public/private keys with the command 'ssh-keygen -t keytype' where keytype can be DSA (Digital Secure Algorithm) or RSA1 (RSA Security). Later in this lesson will be shown how to use it.   
  
**ssh-agent**   
  
This utility holds private keys used for RSA authentication. The idea is that the ssh-agent command is started in the beginning of an X session or a login session, and all other windows or programs are started as clients to the ssh-agent program. In this way all clients of the ssh-agent can remember through the use of environment variables the public/private keys used when ssh-agent was started, so the user will not be ask for this in all these client sessions.   
  
**ssh-add**   
  
Adds RSA identities to the authentication agent ssh-agent

**SSH Server**

The SSH server configuration file is **/etc/ssh/sshd\_config**. This file is well commented so just having a look on it one can understand the meaning of the main directives.   
  
**# cat /etc/ssh/sshd\_config**# This directive configures SSH version 2, which is more secure that version 1  **Protocol 2**# The following sends all logging attempts to the appropriate log file /var/log/secure  **SyslogFacility AUTHPRIV**# This directive authorizes authentication based on local user passwords  **PasswordAuthentication yes**# Set this to 'yes' to enable PAM authentication, account processing, and session processing  **UsePAM yes**# The following directive allows to open remote GUI tools executed through SSH using the local X Server  **X11Forwarding yes**# This directive supports the use of SSH encryption for secure FTP file transfers  **Subsystem sftp /usr/libexec/openssh/sftp-server**   
  
Once the configuration file has been set lets start the ssh server and make sure it will start at system boot.   
  
**# /etc/init.d/sshd restart   
# chkconfig sshd on**

**SSH client**

The SSH client standard configuration file for all system is **/etc/ssh/ssh\_config**. Each user can have custom SSH client configurations in their ~/.ssh/config files.   
  
Some examples of SSH client tools can be :   
  
**ssh**   
  
Allows to login and execute shell commands on remote systems.   
  
**node01> ssh rhel6 -l khan**   
It will login as khan on rhel6 system.   
  
**node01> ssh rhel6 "ls -lrt /home/khan".**   
It will execute the command 'ls -lrt /home/khan' as user khan on rhel6 system. The command output is displayed on node01 the SSH client from where are launched the connection   
  
**scp**   
  
Used to transfer data between computer systems using SSH.   
  
**node01> scp /tmp/file.txt khan@rhel6:/tmp/file.txt**   
This command will transfer file /tmp/file.txt from SSH client node01 to SSH server rhel6 on /tmp directory using 'khan' account.   
  
**node01> scp -r khan@rhel6:/tmp/dir /tmp/**   
This command will transfer from SSH client rhel6 the directory /tmp/dir to the SSH server node01 on /tmp dir using 'khan' account. In this case node01 receives the data so node01 is the SSH server, sshd daemon must be running on node01.

**SSH Security**

**Firewall**

As has been commented the sshd server listen on port 22 TCP/IP so this port must be open in order to allow ssh server service through a firewall.   
  
**-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT**

**User and Host Based Security**

Some additional directives can be added to **/etc/sshd/sshd\_config** file in order to make the access to ssh server more restrictive.   
  
# Do not allow empty passwords   
**PermitEmptyPasswords no**   
  
# The following directive will not allow to root user to log on the system using ssh. (Do not allow remote root logins)   
**PermitRootLogin no**   
  
# Limit the users allowed to access a system via SSH. In this case only users 'khan' and 'charles' are allowed to login on the system using SSH   
**AllowUsers khan charles**   
  
# Or even more restrictive, only allow login through SSH users 'khan' and 'charles' from 192.168.1.101 node.   
**AllowUsers khan@192.168.1.101 charles@192.168.1.101**   
  
# In addition you can restrict the access to users. In this case all users less 'khan' are allowed to connect to the SSH server.   
**DenyUsers khan**

**SSH using only public/private keys**

If the system where SSH server is running is directly connected to the Internet it will be a good idea to disable password authentication on the SSH server and allow only public/private keys authentication. This will reduce dramatically the chance that a cracker has login on your system because the probability that he has to guess the pair user/private\_key is much lower that user/password pair. In order to accomplish this the following directives must be changed/added to **/etc/ssh/sshd\_config** file.   
  
**# cat /etc/ssh/sshd\_config**   
...   
# Do not allow password authentication   
**PasswordAuthentication no**   
  
# Allow public/private key authentication   
**PubkeyAuthentication yes   
AuthorizedKeysFile .ssh/authorized\_keys**   
...   
  
Next step is create the public/private key pair on the ssh client node01 from where are going to connect to the SSH server (rhel6).   
  
**node01> su - khan   
khan-$>ssh-keygen -t dsa**(It will ask for a passphrase in order to protect your private key on the local node)   
  
This command will create khan private key on **/home/khan/.ssh/id\_dsa** (permissions 600) and khan public key on **/home/khan/.ssh/id\_dsa.pub** (permissions 644)   
  
Change de .ssh directory permissions to 755.   
  
**khan-$> chmod 755 .ssh**   
  
Copy the content of /home/khan/.ssh/id\_dsa.pub (khan public key) generated on node01 (the node from we want to login as khan on SSH server) to /home/khan/.ssh/authorized\_keys on SSH server. If necessary create /home/khan/.ssh directory with permission 755 on SSH server.   
  
**khan-$> cat /home/khan/.ssh/id\_dsa.pub --> >> SSH server(rhel6):/home/khan/.ssh/authorized\_keys**   
  
On SSH server (rhel6) change the permissions of /home/khan/.ssh/authorized\_keys to 644.   
  
**# chmod 644 /home/khan/.ssh/authorized\_keys**   
  
The final step is restart the ssh server and verify that you can connect from SSH client (node01) to SSH server (rhel6) only using public/private key and not using the user password. Have a look on Lab2.   
  
**# /etc/init.d/sshd reload**   
  
Note: In order to use the private key on SSH client to connect to SSH server the passphrase introduced when the private key has been created with 'ssh-keygen' is asked. If you have left this passphrase empty you will be able to login to SSH server directly without passphrase BUT using your public/private keys. We do not recommend to left this passphrase empty but in any case this method is more secure that using standard password because in this case the cracker must guess the public/private keys that normaly are random strings with at least 512K of size !!!

**Using ssh-agent**

When we are running a graphical environment on SSH client as gnome or kde we can use the ssh-add utility in order to do not have to enter the passphrase every time we try to connect to the SSH server.   
  
**khan-$> exec /usr/bin/ssh-agent $SHELL   
khan-$> ssh-add**   
(--> Enter khan passphrase)   
  
The khan passphrase now is stored in the environment variables for 'khan' graphical session, so khan must not be to retype his passphrase any time that try to login to the SSH server from this graphical environment on SSH client.

**SSH Port Forwarding**

SSH can secure insecure TCP/IP protocols via port forwarding, SSH server becomes an encrypted conduit to the SSH client. Port forwarding maps a local port on the SSH client to a remote port on the SSH server.   
  
**client> ssh -l khan -L 2525:server.info.net:25 server.info.net**   
  
Once the user khan has been logged on server.info.net through this ssh connection an SSH encrypted Tunnel has been established between port 25 TCP/IP on server.info.net and port 2525 TCP/IP on client.info.com. In this way if you execute the command 'telnet localhost 2525' on client.info.com you are making the telnet directly to port 25 TCP/IP on server.info.net.   
  
**client> telnet localhost 2525   
Trying ::1...   
Connected to localhost.   
Escape character is '^]'.   
220 server.info.net ESMTP Sendmail 8.13.8/8.13.8; Fri, 25 Mar 2011 13:18:29 +0100**   
  
!!! IT IS MAGIC !!!   
  
If you want forward a port from a machine that is not running an SSH server, but another machine on the same network is, SSH can still be used to secure a SSH tunnel.   
  
**client> ssh -l khan -L 1100:pop.info.net:110 server.info.net**   
  
With this command you are making a ssh tunnel from pop.info.net:110 (that is not running an SSH server) to your local machine client.info.com:1100 connecting as user 'khan' on server.info.net that is in the same LAN as pop.info.net. As POP service does not encrypt the data itself, with the SSH tunnel the data is encrypted by SSH, so you are making more secure the connection to your pop service.   
  
Note: SSH Tunnels can be used to skip firewalls. Imagine that there is a firewall that blocks the connection between your local machine client.info.com and your POP service on pop.info.net port 110 TCP/IP. If the firewall is not blocking access to the SSH server on pop.info.net:22 (or a machine in the same LAN running SSH server) you can establish an SSH tunnel from client.info.com:110 and pop.info.net:1100 and skip the firewall. **!!! In reality you can forward any port and skip the firewall if you can connect through ssh!!!**   
  
Maybe for security reasons you want to disable port forwarding through your SSH server. In this case the following directive must be configured on the SSH server configuration file **/etc/ssh/sshd\_config** and then reload the SSH server service.