A single Linux system can provide several different kinds of services, ranging from security to administration and including more obvious Internet services like websites and FTP sites, email, and printing. Security tools such as SSH and Kerberos run as services, along with administrative network tools such as DHCP and LDAP. The network connection interface is itself a service that you can restart at will. Each service operates as a continually running daemon looking for requests for its particular services. In the case of a web service, the requests will come from remote users. You can turn services on or off by starting or shutting down their daemons. The process of starting up or shutting down a service is handled by service scripts. This Chapter will help students understand and configure different servers like DNS Server (BIND), Mail Server,Web Server,File Server (Samba),DHCP Server. Along with Installation and Configuration of a SSH server and ftp server and client.

16.2 DNS Server (BIND)

DNS is usually implemented using one or more centralized servers that are authoritative for certain domains. When a client host requests information from a nameserver, it usually connects to port 53. The nameserver then attempts to resolve the name requested. If it does not have an authoritative answer, or does not already have the answer cached from an earlier query, it queries other nameservers, called root nameservers, to determine which nameservers are authoritative for the name in question, and then queries them to get the requested name.

In a DNS server such as BIND (Berkeley Internet Name Domain), all information is stored in basic data elements called resource records (RR). The resource record is usually a fully qualified domain name (FQDN) of a host, and is broken down into multiple sections organized into a tree-like hierarchy. This hierarchy consists of a main trunk, primary branches, secondary branches, and so on.

There are two nameserver configuration types:

Authoritative: It answers to resource records that are part of their zones only. This category includes both primary (master) and secondary (slave) nameservers.

Recursive: offer resolution services, but they are not authoritative for any zone. Answers for all resolutions are cached in a memory for a fixed period of time, which is specified by the retrieved resource record.
BIND consists of a set of DNS-related programs. It contains a nameserver called named, an administration utility called rndc, and a debugging tool called dig.

16.2.1. Configuring the named Service

When the named service is started, it reads the configuration from the files as described in following Table 1: The named service configuration files.

Table-1: **The named service configuration files**

Path	**Description**
/etc/named.conf	The main configuration file.
/etc/named/	An auxiliary directory for configuration files that is included in the main configuration file.

The configuration file consists of a collection of statements with nested options surrounded by opening and closing curly brackets. Note that when editing the file, you have to be careful not to make any syntax error, otherwise the named service will not start. A typical /etc/named.conf file is organized as follows:

```
statement-1 ["statement-1-name"] [statement-1-class] {
 option-1;
 option-2;
 option-N;
};
statement-2 ["statement-2-name"] [statement-2-class] {
 option-1;
 option-2;
 option-N;
};
statement-N ["statement-N-name"] [statement-N-class] {
 option-1;
 option-2;
 option-N;
};
```

The following types of statements are commonly used in /etc/named.conf:

Acl

The acl (Access Control List) statement allows you to define groups of hosts, so that they can be permitted or denied access to the nameserver. It takes the following form:

```
aclacl-name {
match-element;
 ...
};
```

The *acl-name* statement name is the name of the access control list, and the *match-element* option is usually an individual IP address (such as 10.0.1.1) or a CIDR (Classless Inter-Domain Routing) network notation (for example, 10.0.1.0/24). For a list of already defined keywords, see below "Predefined access control lists".

Table: Predefined access control lists

Keyword	**Description**
any	Matches every IP address
localhost	Matches IP address that is in use by the local system
localnets	Matches any IP address on any network to which the local system is connected.
None	Does not match any IP address

```
acl black-hats {
  10.0.2.0/24;
  192.168.0.0/24;
  1234:5678::9abc/24;
};
acl red-hats {
  10.0.1.0/24;
};
options {
blackhole { black-hats; };
  allow-query { red-hats; };
  allow-query-cache { red-hats; };
};
```

include

The include statement allows you to include files in the /etc/named.conf, so that potentially sensitive data can be placed in a separate file with restricted permissions. It takes the following form:

```
include "file-name"
```

The *file-name* statement name is an absolute path to a file.

```
include "/etc/named.rfc1912.zones";
```

options

The options statement allows you to define global server configuration options as well as to set defaults for other statements. It can be used to specify the location of the named working directory, the types of queries allowed, and much more. It takes the following form:

```
options {
option;
  ...
};
```

For a list of frequently used *option* directives, shown in below Table

Table :Commonly used options

Option	**Description**
allow-query	Specifies which hosts are allowed to query the nameserver for authoritative resource records. It accepts an access control list, a collection of IP addresses, or networks in the CIDR notation. All hosts are allowed by default.
allow-query-cache	Specifies which hosts are allowed to query the nameserver for non-authoritative data such as recursive queries. Only localhost and localnets are allowed by default.
blackhole	Specifies which hosts are *not* allowed to query the nameserver. This option should be used when particular host or network floods the server with requests. The default option is none.
directory	Specifies a working directory for the named service. The default option is /var/named/.

forwarders	Specifies a list of valid IP addresses for nameservers to which the requests should be forwarded for resolution.
forward	Specifies the behaviour of the forwarders directive. It accepts the following options: first — The server will query the nameservers listed in the forwardersdirective before attempting to resolve the name on its own. only — When unable to query the nameservers listed in the forwardersdirective, the server will not attempt to resolve the name on its own.
listen-on	Specifies the IPv4 network interface on which to listen for queries. On a DNS server that also acts as a gateway, you can use this option to answer queries originating from a single network only. All IPv4 interfaces are used by default.

16.2.2 Common Resource Records

The following resource records are commonly used in zone files:

A

The *Address* record specifies an IP address to be assigned to a name. It takes the following form:

```
hostname IN A IP-address
```

If the *hostname* value is omitted, the record will point to the last specified *hostname*.

Following are the requests for server1.example.com are pointed to 10.0.1.3 or 10.0.1.5.

```
server1  IN  A  10.0.1.3
IN  A  10.0.1.5
```

CNAME

The *Canonical Name* record maps one name to another. Because of this, this type of record is sometimes referred to as an *alias record*. It takes the following form:

```
alias-name IN CNAME real-name
```

- CNAME records are most commonly used to point to services that use a common naming scheme, such as wwwfor Web servers. However, there are multiple restrictions for their usage:
- CNAME records should not point to other CNAME records. This is mainly to avoid possible infinite loops.
- CNAME records should not contain other resource record types (such as A, NS, MX, etc.). The only exception are DNSSEC related records (that is, RRSIG, NSEC, etc.) when the zone is signed.

Other resource record that point to the fully qualified domain name (FQDN) of a host (that is, NS, MX, PTR) should not point to a CNAME record.
The A record binds a host name to an IP address, while the CNAME record points the commonly used www host name to it.

```
server1  IN  A      10.0.1.5
www      IN  CNAME  server1
```

MX
The *Mail Exchange* record specifies where the mail sent to a particular namespace controlled by this zone should go. It takes the following form:

```
IN MX preference-valueemail-server-name
```

The *email-server-name* is a fully qualified domain name (FQDN). The *preference-value* allows numerical ranking of the email servers for a namespace, giving preference to some email systems over others. The MX resource record with the lowest *preference-value* is preferred over the others. However, multiple email servers can possess the same value to distribute email traffic evenly among them.
The first mail.example.com email server is preferred to the mail2.example.com email server when receiving email destined for the example.com domain.

```
example.com.  IN  MX  10  mail.example.com.
```

```
IN  MX  20  mail2.example.com.
```

NS

The *Nameserver* record announces authoritative nameservers for a particular zone. It takes the following form:

```
IN NS nameserver-name
```

The *nameserver-name* should be a fully qualified domain name (FQDN). Note that when two nameservers are listed as authoritative for the domain, it is not important whether these nameservers are secondary nameservers, or if one of them is a primary server. They are both still considered authoritative.

```
IN  NS  dns1.example.com.
IN  NS  dns2.example.com.
```

PTR

The *Pointer* record points to another part of the namespace. It takes the following form:

```
last-IP-digit IN PTR FQDN-of-system
```

The *last-IP-digit* directive is the last number in an IP address, and the *FQDN-of-system* is a fully qualified domain name (FQDN).

PTR records are primarily used for reverse name resolution, as they point IP addresses back to a particular name.

SOA

The *Start of Authority* record announces important authoritative information about a namespace to the nameserver. Located after the directives, it is the first resource record in a zone file. It takes the following form:

```
@  IN  SOA  primary-name-serverhostmaster-email (
serial-number
time-to-refresh
time-to-retry
time-to-expire
minimum-TTL )
```

The directives are as follows:

- The @ symbol places the $ORIGIN directive (or the zone's name if the $ORIGIN directive is not set) as the namespace being defined by this SOA resource record.
- The *primary-name-server* directive is the host name of the primary nameserver that is authoritative for this domain.
- The *hostmaster-email* directive is the email of the person to contact about the namespace.
- The *serial-number* directive is a numerical value incremented every time the zone file is altered to indicate it is time for the named service to reload the zone.
- The *time-to-refresh* directive is the numerical value secondary nameservers use to determine how long to wait before asking the primary nameserver if any changes have been made to the zone.
- The *time-to-retry* directive is a numerical value used by secondary nameservers to determine the length of time to wait before issuing a refresh request in the event that the primary nameserver is not answering. If the primary server has not replied to a refresh request before the amount of time specified in the *time-to-expire* directive elapses, the secondary servers stop responding as an authority for requests concerning that namespace.
- In BIND 4 and 8, the *minimum-TTL* directive is the amount of time other nameservers cache the zone's information. In BIND 9, it defines how long negative answers are cached for. Caching of negative answers can be set to a maximum of 3 hours (that is, 3H).

16.3 Mail Server

Linux offers many advanced applications to serve and access email. This section describes modern email protocols in use today, and some of the programs designed to send and receive email.

16.3.1 Email Protocols

Today, email is delivered using a client/server architecture. An email message is created using a mail client program. This program then sends the message to a server. The server then forwards the message to the recipient's email server, where the message is then supplied to the recipient's email client.

To enable this process, a variety of standard network protocols allow different machines, often running different operating systems and using different email programs, to send and receive email.

The following protocols discussed are the most commonly used in the transfer of email.

16.3.2 Mail Transport Protocols

Mail delivery from a client application to the server, and from an originating server to the destination server, is handled by the *Simple Mail Transfer Protocol* (*SMTP*).The primary purpose of SMTP is to transfer email between mail servers. However, it is critical for email clients as well. To send email, the client sends the message to an outgoing mail server, which in turn contacts the destination mail server for delivery. For this reason, it is necessary to specify an SMTP server when configuring an email client.

Under Red Hat Enterprise Linux, a user can configure an SMTP server on the local machine to handle mail delivery. However, it is also possible to configure remote SMTP servers for outgoing mail. One important point to make about the SMTP protocol is that it does not require authentication. This allows anyone on the Internet to send email to anyone else or even to large groups of people. It is this characteristic of SMTP that makes junk email or *spam* possible. Imposing relay restrictions limits random users on the Internet from sending email through your SMTP server, to other servers on the internet. Servers that do not impose such restrictions are called *open relay* servers.

16.3.4 Mail Access Protocols

There are two primary protocols used by email client applications to retrieve email from mail servers: the Post Office Protocol (POP) and the Internet Message Access Protocol (IMAP).

POP: The default POP server under Red Hat Enterprise Linux is Dovecot and is provided by the dovecot package.

When using a POP server, email messages are downloaded by email client applications. By default, most POP email clients are automatically configured to delete the message on the email server after it has been successfully transferred, however this setting usually can be changed.

POP is fully compatible with important Internet messaging standards, such as Multipurpose Internet Mail Extensions (MIME), which allow for email attachments. POP works best for users who have one system on which to read email. It also works well for users who do not have a persistent connection to the Internet or the network containing the mail server. Unfortunately for those with slow network connections, POP requires client programs upon authentication to download the entire content of each message. This can take a long time if any messages have large attachments. The most current version of the standard POP protocol is POP3. There are, however, a variety of lesser-used POP protocol variants:

- APOP — POP3 with MD5 authentication. An encoded hash of the user's password is sent from the email client to the server rather than sending an unencrypted password.
- KPOP — POP3 with Kerberos authentication.
- RPOP — POP3 with RPOP authentication. This uses a per-user ID, similar to a password, to authenticate POP requests. However, this ID is not encrypted, so RPOP is no more secure than standard POP.

16.3.5 IMAP

The default IMAP server under Red Hat Enterprise Linux is Dovecot and is provided by the dovecot package.

When using an IMAP mail server, email messages remain on the server where users can read or delete them. IMAP also allows client applications to create, rename, or delete mail directories on the server to organize and store email.

IMAP is particularly useful for users who access their email using multiple machines. The protocol is also convenient for users connecting to the mail server via a slow connection, because only the email header information is downloaded for messages until opened, saving bandwidth. The user also has the ability to delete messages without viewing or downloading them.

For convenience, IMAP client applications are capable of caching copies of messages locally, so the user can browse previously read messages when not directly connected to the IMAP server.

IMAP, like POP, is fully compatible with important Internet messaging standards, such as MIME, which allow for email attachments.

For added security, it is possible to use SSL encryption for client authentication and data transfer sessions. This can be enabled by using the imaps service, or by using the stunnel program.

Other free, as well as commercial, IMAP clients and servers are available, many of which extend the IMAP protocol and provide additional functionality.

16.3.5 Dovecot

The imap-login and pop3-login processes which implement the IMAP and POP3 protocols are spawned by the master dovecot daemon included in the dovecot package. The use of IMAP and POP is configured through the /etc/dovecot/dovecot.conf configuration file; by default dovecot runs IMAP and POP3 together with their secure versions using SSL. To configure dovecot to use POP, complete the following steps:

1. Edit the /etc/dovecot/dovecot.conf configuration file to make sure the protocols variable is uncommented (remove the hash sign (#) at the beginning of the line) and contains the pop3 argument. For example:

```
protocols = imap pop3 lmtp
```

   When the protocols variable is left commented out, dovecot will use the default values as described above.

2. Make the change operational for the current session by running the following command:

```
~]# service dovecot restart
```

3. Make the change operational after the next reboot by running the command:

```
~]# chkconfig dovecot on
```

Unlike SMTP, both IMAP and POP3 require connecting clients to authenticate using a user name and password. By default, passwords for both protocols are passed over the network unencrypted.

To configure SSL on dovecot:

- Edit the /etc/dovecot/conf.d/10-ssl.conf configuration to make sure the ssl_cipher_list variable is uncommented, and append :!SSLv3:

```
ssl_cipher_list = ALL:!LOW:!SSLv2:!EXP:!aNULL:!SSLv3
```

These values ensure that dovecot avoids SSL versions 2 and also 3, which are both known to be insecure. This is due to the vulnerability described in POODLE: SSLv3 vulnerability (CVE-2014-3566).

16.4 Web Server

HTTP (Hypertext Transfer Protocol) server, or a *web server*, is a network service that serves content to a client over the web. This typically means web pages, but any other documents can be served as well.

16.4.1 The Apache HTTP Server

This section focuses on the **Apache HTTP Server 2.2**, a robust, full-featured open source web server developed by the Apache Software Foundation, that is included in Red Hat Enterprise Linux 6. It describes the basic configuration of the httpd service, and covers advanced topics such as adding server modules, setting up virtual hosts, or configuring the secure HTTP server.There are important differences between the Apache HTTP Server 2.2 and version 2.0, and if you are upgrading from a previous release of Red Hat Enterprise Linux, you will need to update the httpd service configuration accordingly. This section reviews some of the newly added features, outlines important changes, and guides you through the update of older configuration files.

The Apache HTTP Server version 2.2 introduces the following enhancements:

- Improved caching modules, that is, mod_cache and mod_disk_cache.
- Support for proxy load balancing, that is, the mod_proxy_balancer module.
- Support for large files on 32-bit architectures, allowing the web server to handle files greater than 2GB.
- A new structure for authentication and authorization support, replacing the authentication modules provided in previous versions.

Running the httpd Service

This section describes how to start, stop, restart, and check the current status of the Apache HTTP Server. To be able to use the httpd service, make sure you have the httpd installed. You can do so by using the following command:

```
~]# yum install httpd
```

16.4.2 Starting the Service

To run the httpd service, type the following at a shell prompt as root:

```
~]# service httpd start
Starting httpd:                          [  OK  ]
```

If you want the service to start automatically at the boot time, use the following command:

```
~]#chkconfighttpd on
```

This will enable the service for runlevel 2, 3, 4, and 5..

16.4.3 Stopping the Service

To stop the running httpd service, type the following at a shell prompt as root:

```
~]# service httpd stop
Stopping httpd:                          [  OK  ]
```

To prevent the service from starting automatically at the boot time, type:

```
~]#chkconfighttpd off
```

This will disable the service for all runlevels. Alternatively, you can use the **Service Configuration** utility also under "Enabling and Disabling a Service" option.

16.4.5 Restarting the Service

There are three different ways to restart a running httpd service:

1. To restart the service completely, enter the following command as root:

```
2. ~]# service httpd restart
3. Stopping httpd:                       [  OK  ]
```

```
Starting httpd:                          [  OK  ]
```

This stops the running httpd service and immediately starts it again. Use this command after installing or removing a dynamically loaded module such as PHP.

4. To only reload the configuration, as root, type:

```
~]# service httpd reload
```

This causes the running httpd service to reload its configuration file. Any requests being currently processed will be interrupted, which may cause a client browser to display an error message or render a partial page.

5. To reload the configuration without affecting active requests, enter the following command as root:

```
~]# service httpd graceful
```

This causes the running httpd service to reload its configuration file. Any requests being currently processed will use the old configuration.

Alternatively, you can use the Service Configuration utility also under "Starting, Restarting, and Stopping a Service" option.

16.4.6 Verifying the Service Status

To verify that the httpd service is running, type the following at a shell prompt:

```
~]# service httpd status
httpd (pid 19014) is running...
```

When the httpd service is started, by default, it reads the configuration from locations that are listed in Table below, "The httpd service configuration files".

Table : The httpd service configuration files

Path	Description
/etc/httpd/conf/httpd.conf	The main configuration file.
/etc/httpd/conf.d/	An auxiliary directory for configuration files that are included in the main configuration file.

Although the default configuration should be suitable for most situations, it is a good idea to become at least familiar with some of the more important configuration options. Note that for any changes to take effect, the web server has to be restarted first. To check the configuration for possible errors, type the following at a shell prompt:

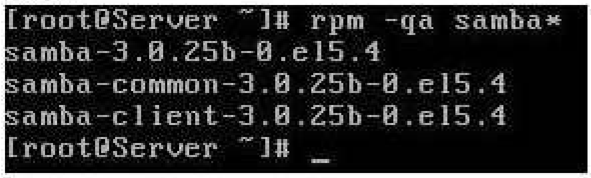
```
~]# service httpdconfigtest
Syntax OK
```

16.5 File Server(Samba)

To make the recovery from mistakes easier, it is recommended that you make a copy of the original file before editing it.

Most Linux systems are the part of networks that also run Windows systems. Using Linux **Samba servers**, your Linux and Windows systems can share directories and printers. This is most use full situation where your clients are window native and you want to use the linux security features.

samba rpm is required to configure samba server. check them if not found then install



```
[root@Server ~]# rpm -qa samba*
samba-3.0.25b-0.el5.4
samba-common-3.0.25b-0.el5.4
samba-client-3.0.25b-0.el5.4
[root@Server ~]# _
```

Now check smb, portmap, xinetd service in system service it should be on

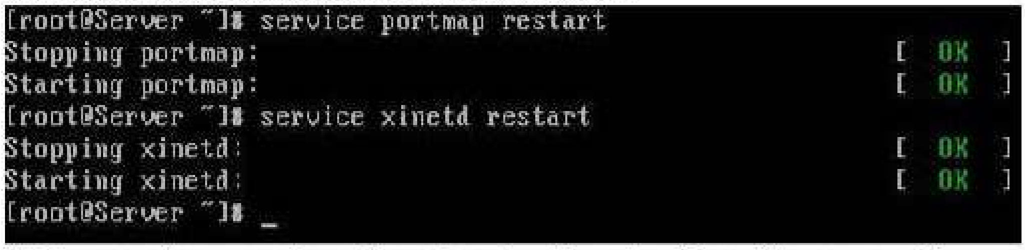
#setup Select System service from list

[*]portmap

[*]xinetd

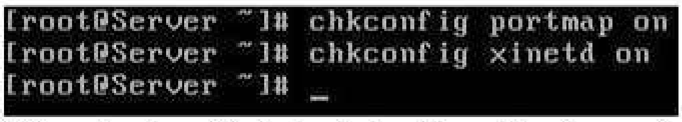
[*]smb

Now restart xinetd and portmap and smb service



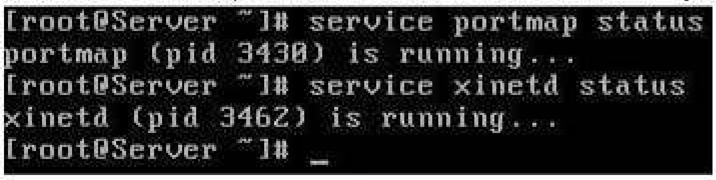
```
[root@Server ~]# service portmap restart
Stopping portmap:                                          [  OK  ]
Starting portmap:                                          [  OK  ]
[root@Server ~]# service xinetd restart
Stopping xinetd:                                           [  OK  ]
Starting xinetd:                                           [  OK  ]
[root@Server ~]# _
```

To keep on these services after reboot on then via chkconfig command



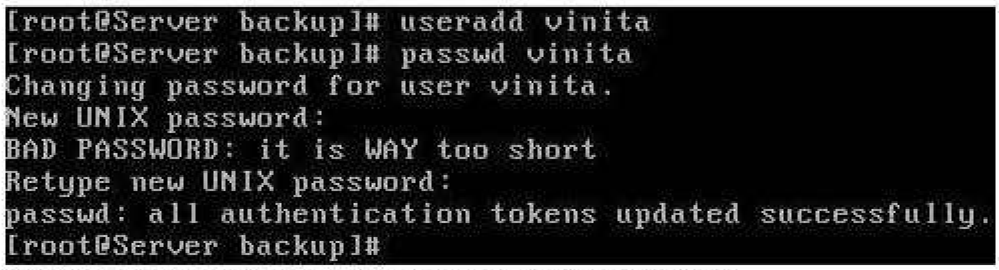
```
[root@Server ~]# chkconfig portmap on
[root@Server ~]# chkconfig xinetd on
[root@Server ~]# _
```

After reboot verify their status. It must be in running condition



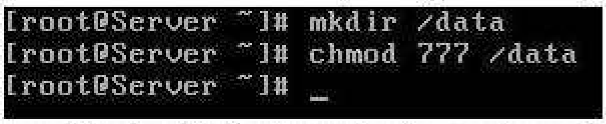
```
[root@Server ~]# service portmap status
portmap (pid 3430) is running...
[root@Server ~]# service xinetd status
xinetd (pid 3462) is running...
[root@Server ~]# _
```

Create a normal user named vinita



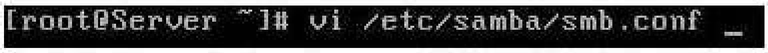
```
[root@Server backup]# useradd vinita
[root@Server backup]# passwd vinita
Changing password for user vinita.
New UNIX password:
BAD PASSWORD: it is WAY too short
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@Server backup]#
```

Now create /**data** directory and grant it **full permission**



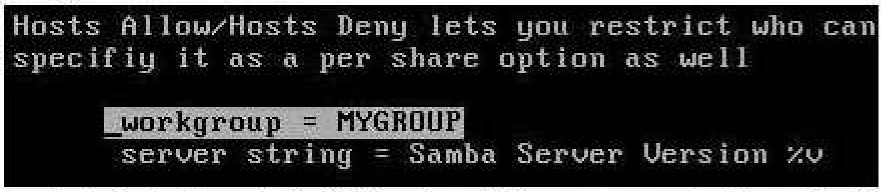
```
[root@Server ~]# mkdir /data
[root@Server ~]# chmod 777 /data
[root@Server ~]# _
```

open **/etc/samba/smb.conf** main samba configuration files



```
[root@Server ~]# vi /etc/samba/smb.conf _
```

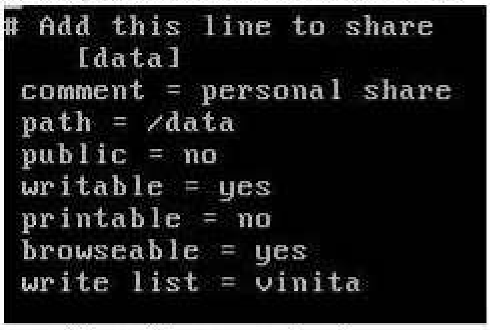
By default name of workgroup is **MYGROUP** in **smb.conf** file. you can change it with desire name



```
Hosts Allow/Hosts Deny lets you restrict who can
specifiy it as a per share option as well

      _workgroup = MYGROUP
       server string = Samba Server Version %v
```

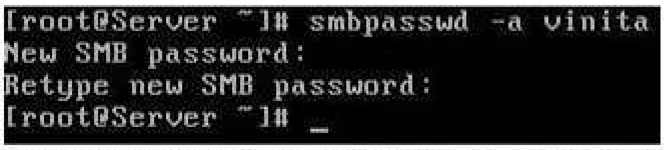
our task is to share **data** folder for **vinita** user so go in the end of file and do editing as shown here in this image



```
# Add this line to share
    [data]
 comment = personal share
 path = /data
 public = no
 writable = yes
 printable = no
 browseable = yes
 write list = vinita
```

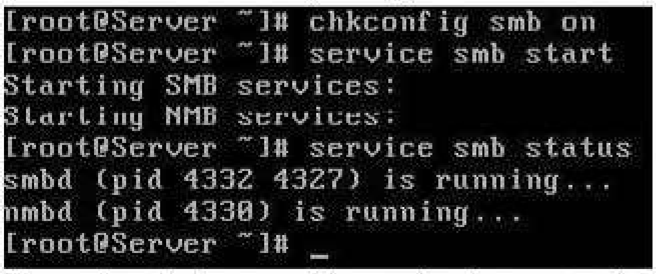
save file with **:wq** and exit

Now add vinita user to **samba user**



```
[root@Server ~]# smbpasswd -a vinita
New SMB password:
Retype new SMB password:
[root@Server ~]# _
```

we have made necessary change now on **smb service** and check it status



```
[root@Server ~]# chkconfig smb on
[root@Server ~]# service smb start
Starting SMB services:
Starting NMB services:
[root@Server ~]# service smb status
smbd (pid 4332 4327) is running...
nmbd (pid 4330) is running...
[root@Server ~]# _
```

if you already have on this service then restart it with **service smb restart** commands.

16.6 DHCP 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.

16.6.1 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.

16.6.2 Configuring a DHCPv4 Server

The dhcp package contains an Internet Systems Consortium (ISC) DHCP server. First, install the package as the superuser:

```
~]# yum install dhcp
```

Installing the dhcp package creates a file, /etc/dhcp/dhcpd.conf, which is merely an empty configuration file:

```
~]# cat /etc/dhcp/dhcpd.conf
# DHCP Server Configuration file.
#   see /usr/share/doc/dhcp*/dhcpd.conf.sample
```

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, which is explained in detail below.

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

16.6.3 Configuration File

The first step in configuring a DHCP server is to create the configuration file that stores the network information for the clients. Use this file to declare options and global options for client systems.

The configuration file can contain extra tabs or blank lines for easier formatting. Keywords are case-insensitive and lines beginning with a hash sign (#) are considered comments.

There are two types of statements in the configuration file:

- Parameters — State how to perform a task, whether to perform a task, or what network configuration options to send to the client.
- Declarations — Describe the topology of the network, describe the clients, provide addresses for the clients, or apply a group of parameters to a group of declarations.

The parameters that start with the keyword option are referred to as options. These options control DHCP options; whereas, parameters configure values that are not optional or control how the DHCP server behaves.

16.7 Installation and Configuration of a SSH server and client

The **openssh-server** RPM package is required to configure a Red Hat Enterprise Linux system as an OpenSSH server. If it is not already installed, install it with **rpm** commands as described in our pervious article. After it is installed, start the service as root with the command **service sshd start** . The system is now an **SSH server** and can accept connections. To configure the server to automatically start the service at boot time, execute the command **chkconfig sshd on** as root. To stop the server, execute the command **service sshd stop.** To verify that the server is running, use the command **service sshd status.**

Configure ssh server

In this example we will configure a ssh server and will invoke connection from client side.

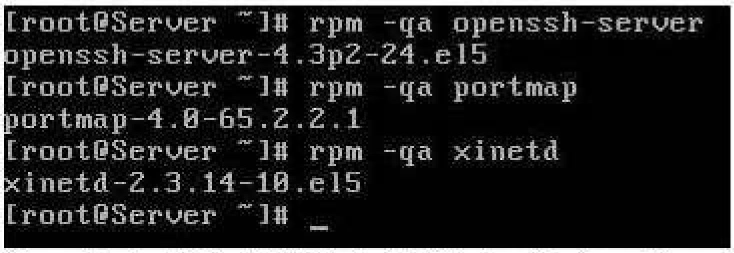
For this example we are using two systems one linux server one linux clients .

To complete these per quest of ssh server Follow this link

Network configuration in Linux

- A linux server with ip address 192.168.0.254 and hostname Server
- A linux client with ip address 192.168.0.1 and hostname Client1
- Updated /etc/hosts file on both linux system
- Running portmap and xinetd services
- Firewall should be off on server

Three rpm are required to configure ssh server. **openssh-server, portmap, xinetd** check them if not found then install



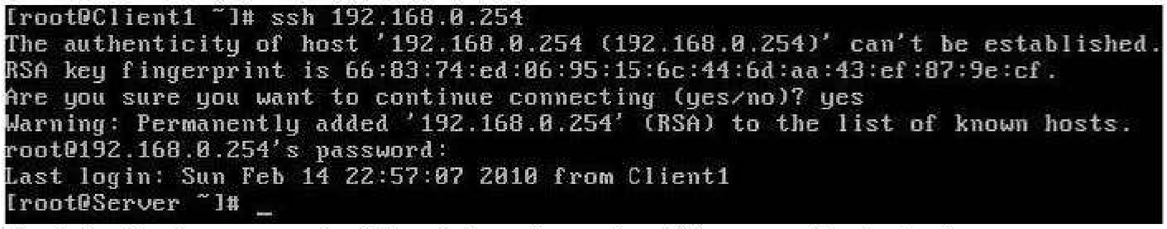
```
[root@Server ~]# rpm -qa openssh-server
openssh-server-4.3p2-24.el5
[root@Server ~]# rpm -qa portmap
portmap-4.0-65.2.2.1
[root@Server ~]# rpm -qa xinetd
xinetd-2.3.14-10.el5
[root@Server ~]# _
```

Now check **sshd, portmap, xinetd** service in system service it should be on

```
#setup
Select  System service from list
[*]portmap
[*]xinetd
[*]sshd
```

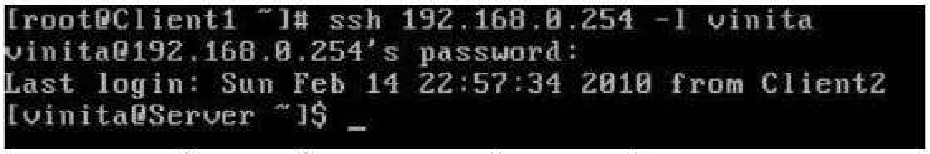
n Linux client

ping from **ssh server** and run **ssh** command and give **root password**



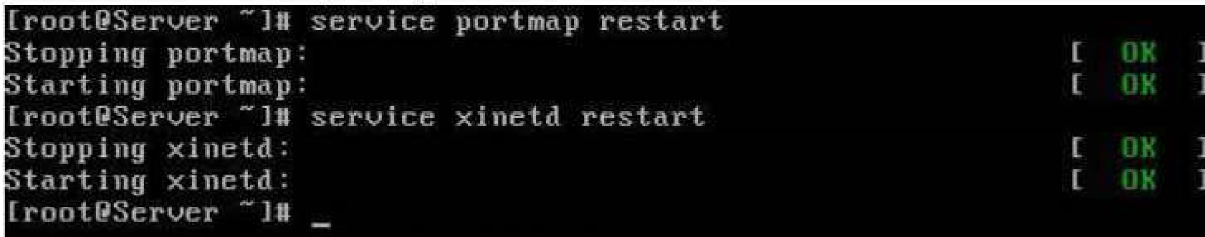
```
[root@Client1 ~]# ssh 192.168.0.254
The authenticity of host '192.168.0.254 (192.168.0.254)' can't be established.
RSA key fingerprint is 66:83:74:ed:06:95:15:6c:44:6d:aa:43:ef:87:9e:cf.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.254' (RSA) to the list of known hosts.
root@192.168.0.254's password:
Last login: Sun Feb 14 22:57:07 2010 from Client1
[root@Server ~]# _
```

By default **ssh** command will enable root session. If you want to login from normal user then specify his name with -l options.

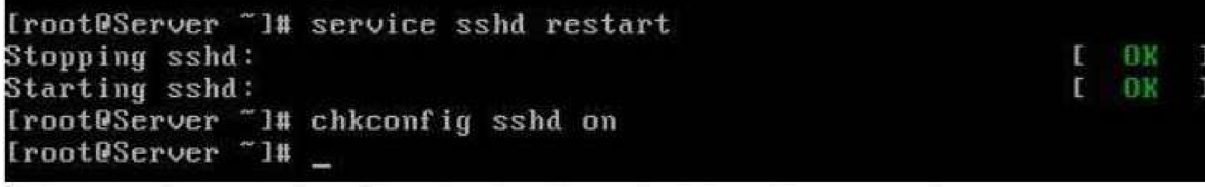


```
[root@Client1 ~]# ssh 192.168.0.254 -l vinita
vinita@192.168.0.254's password:
Last login: Sun Feb 14 22:57:34 2010 from Client2
[vinita@Server ~]$ _
```

Now restart **xinetd** and **portmap** and **sshd** service

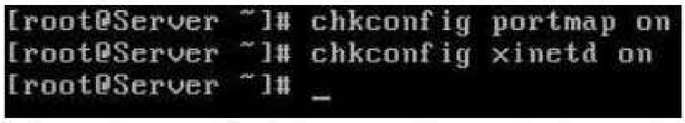


```
[root@Server ~]# service portmap restart
Stopping portmap:                                          [  OK
Starting portmap:                                          [  OK
[root@Server ~]# service xinetd restart
Stopping xinetd:                                           [  OK
Starting xinetd:                                           [  OK
[root@Server ~]# _
```



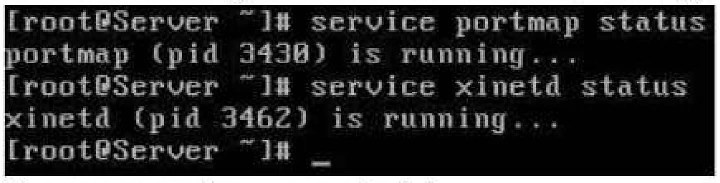
```
[root@Server ~]# service sshd restart
Stopping sshd:                                             [  OK
Starting sshd:                                             [  OK
[root@Server ~]# chkconfig sshd on
[root@Server ~]# _
```

To keep on these services after reboot on then via **chkconfig** command



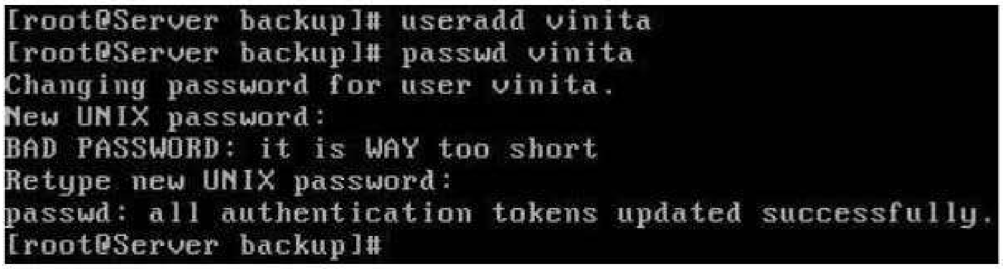
```
[root@Server ~]# chkconfig portmap on
[root@Server ~]# chkconfig xinetd on
[root@Server ~]# _
```

After reboot verify their status. It must be in running condition



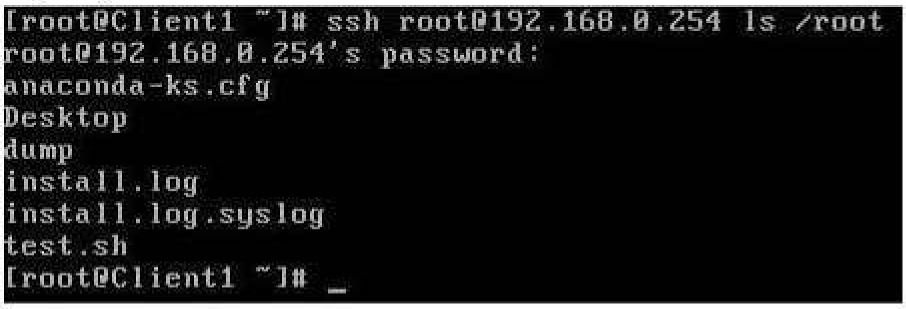
```
[root@Server ~]# service portmap status
portmap (pid 3430) is running...
[root@Server ~]# service xinetd status
xinetd (pid 3462) is running...
[root@Server ~]# _
```

Create a normal user named vinita



```
[root@Server backup]# useradd vinita
[root@Server backup]# passwd vinita
Changing password for user vinita.
New UNIX password:
BAD PASSWORD: it is WAY too short
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@Server backup]#
```

With **ssh** you can run any command on server without login **(user password require)**



```
[root@Client1 ~]# ssh root@192.168.0.254 ls /root
root@192.168.0.254's password:
anaconda-ks.cfg
Desktop
dump
install.log
install.log.syslog
test.sh
[root@Client1 ~]# _
```

16.8 Installation and Configuration of a FTP server and client

Ftp server is used to transfer files between server and clients. All major operating system supports ftp. ftp is the most used protocol over internet to transfer files. Like most Internet operations, FTP works on a client/ server model. FTP client programs can enable users to transfer files to and from a remote system running an FTP server program.

This article is written for RHEL Any Linux system can operate as an FTP server. It has to run only the server software—an FTP daemon with the appropriate configuration. Transfers are made between user accounts on client and server systems. A user on the remote system has to log in to an account on a server and can then transfer files to and from that account's directories only.

A special kind of user account, named **ftp**, allows any user to log in to it with the username **"anonymous."** This account has its own set of directories and files that are considered public, available to anyone on the network who wants to download them.

The numerous FTP sites on the Internet are FTP servers supporting FTP user accounts with anonymous login. Any Linux system can be configured to support anonymous FTP access, turning them into network FTP sites. Such sites can work on an intranet or on the Internet.

Configuring the ftp Server

The **vsftpd** RPM package is required to configure a Red Hat Enterprise Linux system as an ftp server. If it is not already installed, install it with **rpm** commands as described in our pervious article. After it is installed, start the service as root with the command **service vsftpd start** . The system is now an **ftp server** and can accept connections. To configure the server to automatically start the service at boot time, execute the command **chkconfig vsftpd on** as root. To stop the server, execute the command **service vsftpd stop.** To verify that the server is running, use the command **service vsftpd status.**

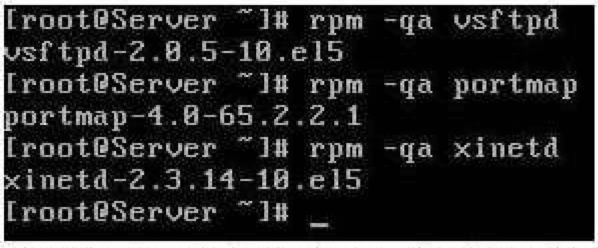
Configure vsftpd server

In this example we will configure a **vsftpd** server and will transfer files from client side.

For this example we are using three systems one linux server one linux clients Network configuration in Linux

- A linux server with ip address 192.168.0.254 and hostname Server
- A linux client with ip address 192.168.0.1 and hostname Client1
- Updated /etc/hosts file on both linux system
- Running portmap and xinetd services
- Firewall should be off on server

Three rpm are required to configure ssh server. **vsftpd, portmap, xinetd** check them if not found then install

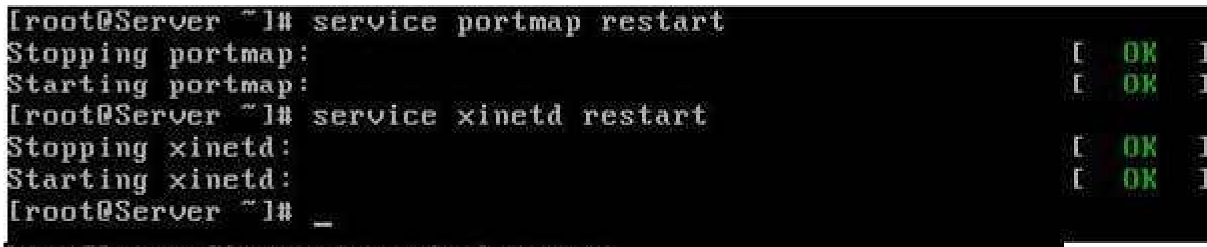


```
[root@Server ~]# rpm -qa vsftpd
vsftpd-2.0.5-10.el5
[root@Server ~]# rpm -qa portmap
portmap-4.0-65.2.2.1
[root@Server ~]# rpm -qa xinetd
xinetd-2.3.14-10.el5
[root@Server ~]# _
```

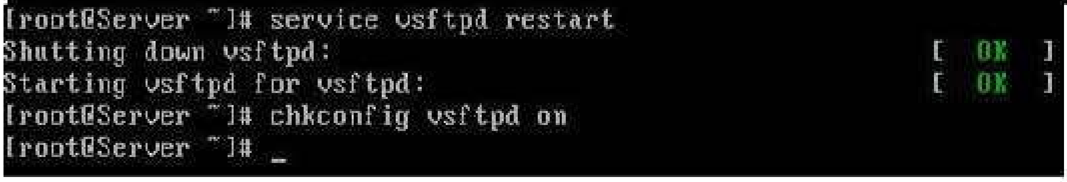
Now check **vsftpd, portmap, xinetd** service in system service it should be on

#setup
Select System service from list
[*]portmap
[*]xinetd
[*]vsftpd

Now restart **xinetd** and **portmap** and **vsftpd** service

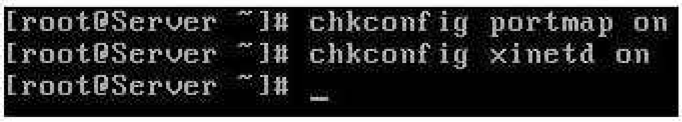


```
[root@Server ~]# service portmap restart
Stopping portmap:                                          [  OK  ]
Starting portmap:                                          [  OK  ]
[root@Server ~]# service xinetd restart
Stopping xinetd:                                           [  OK  ]
Starting xinetd:                                           [  OK  ]
[root@Server ~]# _
```



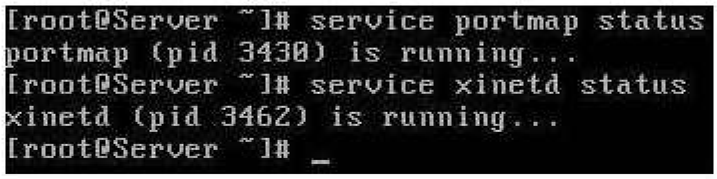
```
[root@Server ~]# service vsftpd restart
Shutting down vsftpd:                                      [  OK  ]
Starting vsftpd for vsftpd:                                [  OK  ]
[root@Server ~]# chkconfig vsftpd on
[root@Server ~]# _
```

To keep on these services after reboot on then via **chkconfig** command



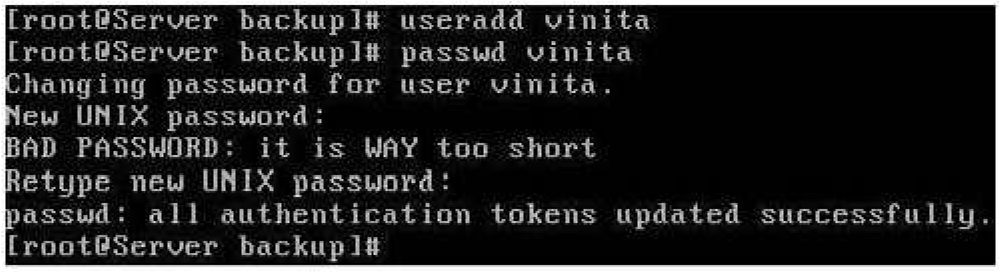
```
[root@Server ~]# chkconfig portmap on
[root@Server ~]# chkconfig xinetd on
[root@Server ~]# _
```

After reboot verify their status. It must be in running condition



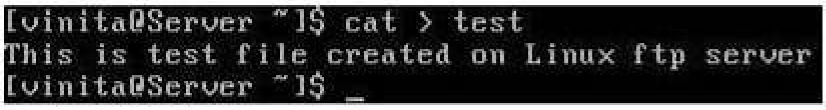
```
[root@Server ~]# service portmap status
portmap (pid 3430) is running...
[root@Server ~]# service xinetd status
xinetd (pid 3462) is running...
[root@Server ~]# _
```

Create a normal user named **vinita**



```
[root@Server backup]# useradd vinita
[root@Server backup]# passwd vinita
Changing password for user vinita.
New UNIX password:
BAD PASSWORD: it is WAY too short
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@Server backup]#
```

Login for this user on other terminal and create a **test** file



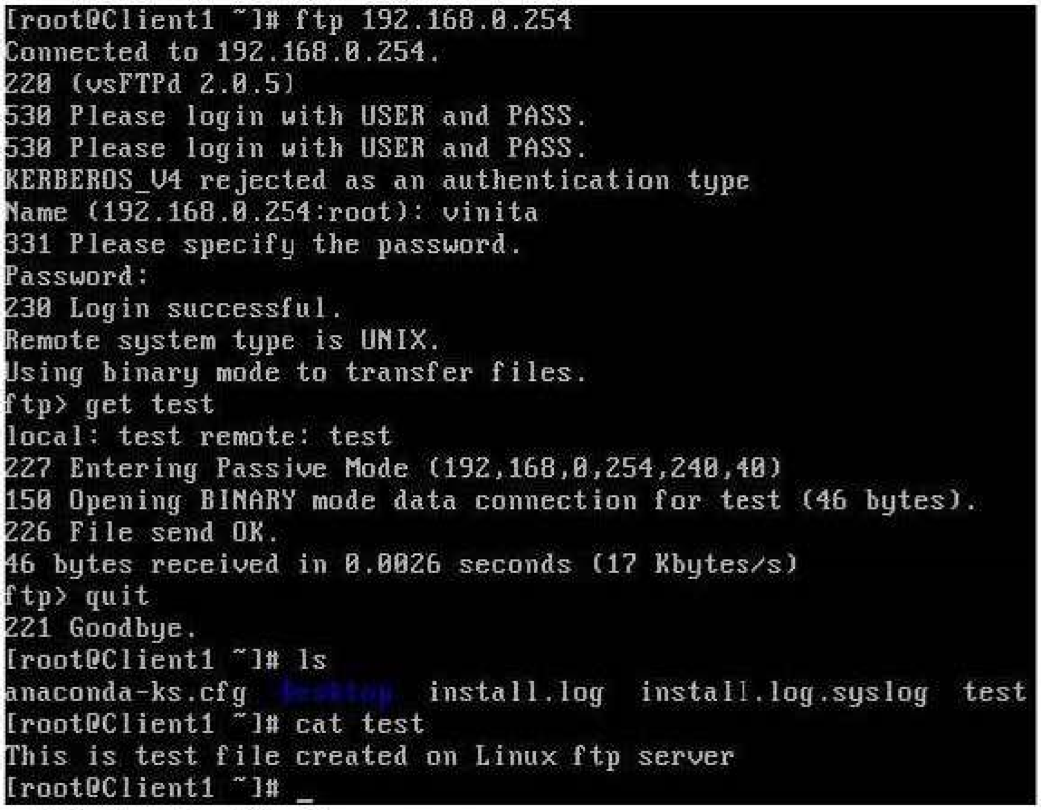
```
[vinita@Server ~]$ cat > test
This is test file created on Linux ftp server
[vinita@Server ~]$ _
```

On Linux client **ping** from **ftp server** and run **ftp** command and give **username** and **password.**

after login you can download files from the specified directories

Most commonly commands used on ftp prompt are

put To upload files on server



```
[root@Client1 ~]# ftp 192.168.0.254
Connected to 192.168.0.254.
220 (vsFTPd 2.0.5)
530 Please login with USER and PASS.
530 Please login with USER and PASS.
KERBEROS_V4 rejected as an authentication type
Name (192.168.0.254:root): vinita
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> get test
local: test remote: test
227 Entering Passive Mode (192,168,0,254,240,40)
150 Opening BINARY mode data connection for test (46 bytes).
226 File send OK.
46 bytes received in 0.0026 seconds (17 Kbytes/s)
ftp> quit
221 Goodbye.
[root@Client1 ~]# ls
anaconda-ks.cfg  Desktop  install.log  install.log.syslog  test
[root@Client1 ~]# cat test
This is test file created on Linux ftp server
[root@Client1 ~]# _
```

get To download files from server

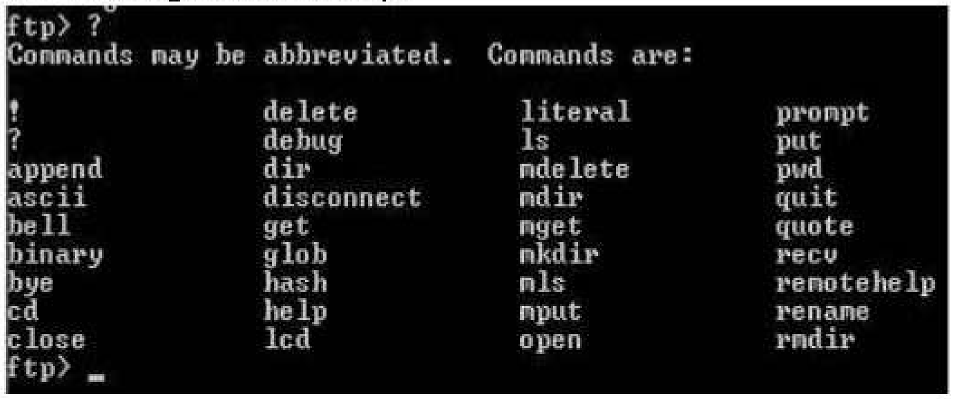
mput To upload all files

mget To download all files

?   To see all available command on ftp prompts

cd   To change remote directory

lcd To change local directory.



```
ftp> ?
Commands may be abbreviated.  Commands are:

!               delete          literal         prompt
?               debug           ls              put
append          dir             mdelete         pwd
ascii           disconnect      mdir            quit
bell            get             mget            quote
binary          glob            mkdir           recv
bye             hash            mls             remotehelp
cd              help            mput            rename
close           lcd             open            rmdir
ftp> _
```

16.9 Self Learning Exercise

Q.1 What service is used to translate domain names to IP addresses?

a) NFS
b) SMB
c) NIS
d) DNS

Q2. What protocol(s) is(are) allowed a user to retrieve her/his mail from the mail server to her/his mail reader?

a) POP3
b) FTP
c) MAP
d) All of the above

Q3. Which of the following server is used with the BIND package?

a) httpd
b) shttp
c) dns
d) named

Q4. Which of the following is the main Apache configuration file?

a) /etc/apachconf
b) /etc/httpd/config.ini
c) /etc/httpd/conf/httpd.conf
d) /etc/srm.conf

Q5. Which of the following command is used to access an SMB share on a Linux system?

a) A.NFS
b) B.SMD
c) C.smbclient
d) D.smbserver

Q.6. Secure shell (SSH) network protocol is used for

a) secure data communication

b) remote command-line login
c) remote command execution
d) all of the mentioned

Q.7. FTP is built on _____ architecture

a) Client-server
b) P2P
c) Both of the mentioned
d) None of the mentioned

16.10 Summary

- A DNS server, or name server, is used to resolve an IP address to a hostname or vice versa.DNS is usually implemented using one or more centralized servers that are authoritative for certain domains. When a client host requests information from a nameserver, it usually connects to port 53. The nameserver then attempts to resolve the name requested.
- Mail Server :Email is delivered using a client/server architecture. An email message is created using a mail client program. This program then sends the message to a server. The server then forwards the message to the recipient's email server, where the message is then supplied to the recipient's email client. SMTP, POP, IMAP protocols used for mailing tasks.
- DHCP Server: 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.
- ftp server is used to transfer files between server and clients. All major operating system supports ftp. ftp is the most used protocol over internet to transfer files. Like most Internet operations, FTP works on a client/ server model. FTP client programs can enable users to transfer files to and from a remote system running an FTP server program.Any Linux system can operate as an FTP server.

- Web Servers :Linux distributions provide several web servers for use on your system. The primary web server is Apache, which has almost become the standard web server for Linux distributions. It is a very powerful, stable, and fairly easy-to-configure system. Other web servers are also available, such as Tux, which is smaller, but very fast and efficient at handling web data that does not change. Linux distributions provide default configurations for the web servers, making them usable as soon as they are installed.
- Telnet and FTP are well-known protocol but they send data in plain text format, which can be captured by someone using another system on the same network, including the Internet.On the other hand, all data transferred using OpenSSH tools is encrypted, making it inherently more secure. The OpenSSH suite of tools includes ssh for securely logging in to a remote system and executing remote commands, scp for encrypting files while transferring them to a remote system, and sftp for secure FTP transfers.

16.11 Glossary

DNS: Domain Name System :The Domain Name System (DNS) is a hierarchical decentralized naming system for computers, services, or other resources connected to the Internet or a private network.

BIND: Berkeley Internet Name Domain: BIND is open source software that enables you to publish your Domain Name System (DNS) information on the Internet, and to resolve DNS queries for your users.

HTTP: Hyper Text Transfer Protocol: The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, and hypermedia information systems.

FTP: File Transfer Protocol: The File Transfer Protocol (FTP) is a standard network protocol used for the transfer of computer files from a server to a client using the Client–server model on a computer network.

SMTP: Simple Mail Transfer Protocol: s a TCP/IP protocol used in sending and receiving e-mail.

POP: Post Office Protocol: a protocol used to retrieve e-mail from a mail server.

DHCP: Dynamic Host Configuration Protocol: is a standardized network protocol used on Internet Protocol (IP) networks. The DHCP is controlled by a

DHCP server that dynamically distributes network configuration parameters, such as IP addresses, for interfaces and services.

16.12 Answers to Self-Learning Exercise

Q1. (d)	Q2. (a)
Q3. (c)	Q4. (c)
Q5. (c)	Q.6.(d)
Q.7. (a)	

16.13 Exercise

Q.1. Explain different email Protocols with their functions.

Q.2. Explain use of Domain Name server in World Wide Web.

Q.3. Explain difference between HTTP and FTP.

Q.4. Explain utilization DHCP servers.

Q.5 .Explain the utilization of SAMBHA Server.

References and Suggested Readings

1. http://www.basicconfig.com/linuxservers.html
2. http://www.tecmint.com/install-openssh-server-in-linux/
3. Richard Peterson, "Linux: The Complete Reference", Osborne McGrawHill.