UPES

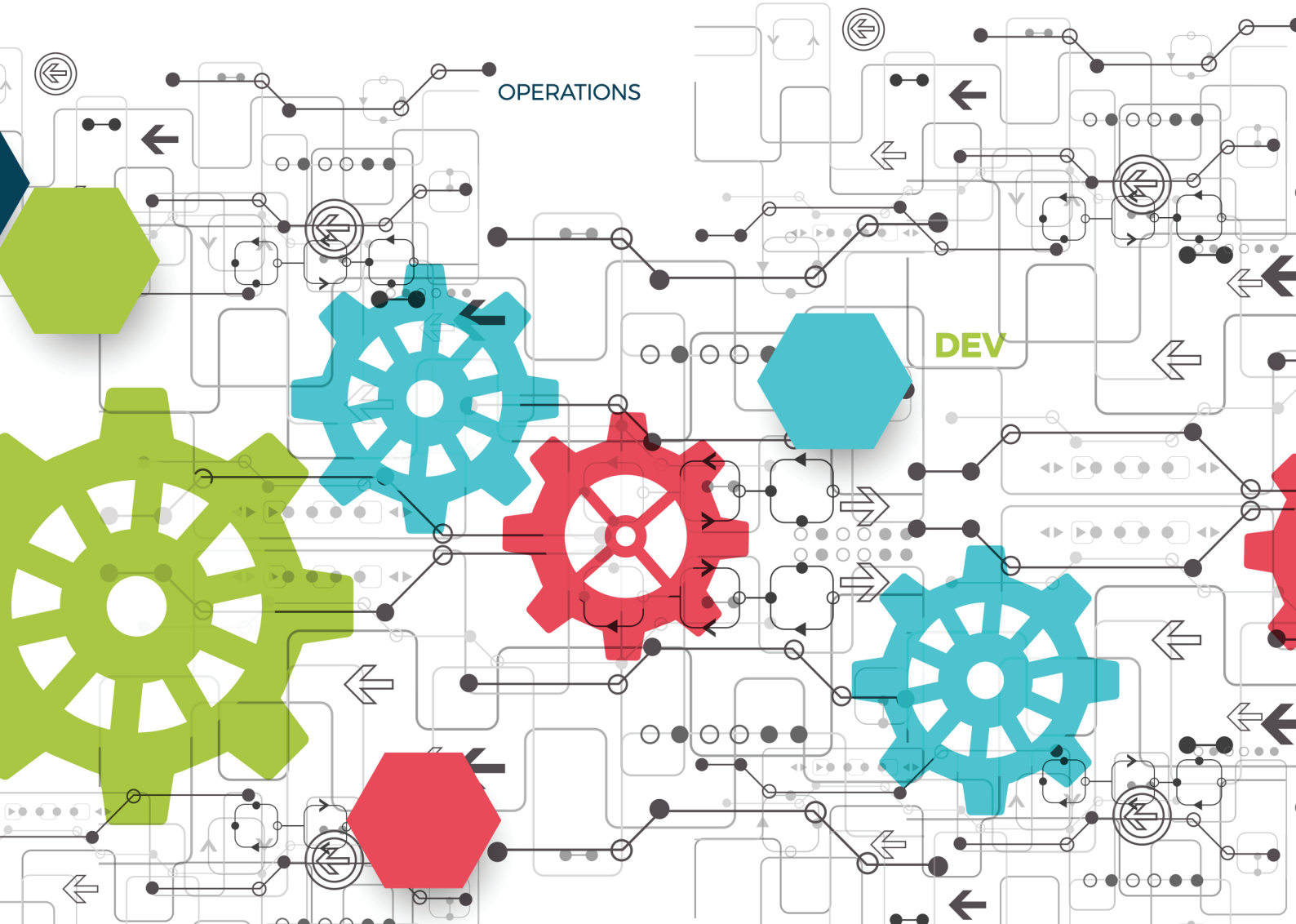
Xebia

B.Tech Computer Science and Engineering in DevOps

Advanced Linux

Semester 04 | Lab

Release 1.1.0



Copyright & Disclaimer

B. TECH CSE with Specialization in Advanced Linux

Version 1.0.0

Acknowledgements

We would like to sincerely thank the experts who have contributed to and shaped B. TECH CSE with Specialization in DevOps. Version 1.0.0
Semester 04

SME

Rajagopalan Varadan
A tech enthusiast who loves learning and working with cutting-edge technologies like DevOps, Big Data, Data science, Machine Learning, AWS & Open stack

Course Reviewers.

Aditya Kalia | Xebia
Maneet Kaur | Xebia
Sandeep Singh Rawat | Xebia
Abhishek Srivastava | Xebia
Rohit Sharma | Xebia

Review Board Members.

Anand Sahay | Xebia

Xebia Group consists of seven specialized, interlinked companies: Xebia, Xebia Academy, XebiaLabs, StackState, GoDataDriven, Xpirit and Binx.io. With offices in Amsterdam and Hilversum (Netherlands), Paris, Delhi, Bangalore and Boston, we employ over 700 people worldwide. Our solutions address digital strategy; agile transformations; DevOps and continuous delivery; big data and data science; cloud infrastructures; agile software development; quality and test automation; and agile software security.

ODW is dedicated to provide innovative and creative solutions that contribute in growth of emerging technologies. As a learning experience provider, ODW strengths include providing unique, up to date content by combining industry best practices with leading edge technology. ODW delivers high quality solutions and services which focus on digital learning transformation.

Advanced Linux Lab

Prerequisites for this lab:

- **Linux machine, either Ubuntu or CentOS**
- **CentOS 7**
- **Ubuntu 14.0+**

2.1 Scripting and the Shell

2.1.1 What is a Shell Program?

A shell program is nothing but a text files that has standard UNIX and shell commands. Each line in the file contains a single command. With shell script you can execute multiple commands at a time.

Shell programs are not compiled programs, it is interpreted. It means when you execute a shell program a new child shell is started and it reads each line in the file and carries out the instructions on that line.

2.1.2 Support for Programming

A programming language must provide the following options to execute programs:

- variables
- comments
- conditional commands
- repeated action commands

2.1.3 Creating a Shell Program

To create a shell program, perform the following steps:

- create a text file to hold the shell program
- decide which shell you will use
- add the required commands to the file
- save the file
- change the permissions on the file so it is executable
- run the shell program

2.1.4 Which Shell?

The first step is to decide which shell to use. You decision here will control the syntax of the shell commands you use. For the purposes of this subject you should always use the Bourne shell. The standard Linux shell bash supports the Bourne shell syntax.

Once you've made that decision the first line of your shell program should indicate which shell should be used to interpret your shell program. This is done by having the first line of the shell program have the following: #!path_to_shell For example, suppose that on my machine I want to use the Bourne shell (sh). The full path to the sh program on my system is /bin/sh (this will be the same on virtually all Unix systems). So the first line of every shell program I write will be:

#!/bin/sh

Add the commands

The next step is to add the commands you wish the shell program to perform. The commands can be any standard UNIX or shell command. Of course any shell commands you use must be supported by the shell that will interpret your shell program.

Make the file executable

Once you've added all the commands you can then save the file. In order to execute a shell program both the read and execute permissions must be set. By default when you save a text file the execute file permission will not be set. You will have to set this manually

The following script uses the read command which takes the input from the keyboard and assigns it as the value of the variable PERSON and finally prints it on STDOUT.

Lab:

1. Create name.sh file in your working directory and add the below code. This script will take the input from user and print the name

```
#!/bin/sh

echo "What is your name?"

read NAME

echo "Hello, $NAME"
```


Type the below command to provide execute permission to the test script

```
$chmod +x test.sh

$./test.sh

What is your name?

John Doe

Hello, John Doe

$
```

2. Ping a host using a script and print the return code

```
#!/bin/sh
#
# 3/26/2017
#
ping -c 1 google.com   .    # ping google.com just 1 time
echo Return code: $?
```

i. Create ping.sh file and add the above code.

ii. Type chmod +x ping.sh

iii. Execute the script ./ping.sh

Sample output:

PING google.com (108.177.8.138) 56(84) bytes of data.

--- google.com ping statistics ---

1 packets transmitted, 0 received, 100% packet loss, time 0ms

Return code: 1

3. Write a shell script which takes arguments while executing the script and map it to First name and Surname and print number of arguments that is entered.

```
#!/bin/bash

# example of using arguments to a script
echo "My first name is $1"
echo "My surname is $2"
echo "Total number of arguments is $#"
```

i. Create arg.sh file and add the above code.

ii. Type chmod +x arg.sh

iii. Execute the script ./arg.sh John Doe

Sample output:

./arg.sh John Doe

My first name is John

My surname is Doe

Total number of arguments is 2

4. Usage of echo command

When you use '**echo**' command without any option then a newline is added by default. **'-n'** option is used to print any text without new line and **'-e'** option is used to remove backslash characters from the output.


```
#!/bin/bash
echo "Printing text with newline"
echo -n "Printing text without newline"
echo -e "\nRemoving \t backslash \t characters\n"
```

i. Create echo.sh file and add the above code.

ii. Type chmod +x echo.sh

iii. Execute the script ./echo.sh

Sample output:

./echo.sh

Printing text with newline

Printing text without newline

Removing backslash characters

5. Create a script with while loop

```
#!/bin/bash
valid=true
count=1
while [ $valid ]
do
echo $count
if [ $count -eq 5 ];
then
break
fi
((count++))
done
```

i. Create while-loop.sh file and add the above code.

ii. Type chmod +x while-loop.sh

iii. Execute the script ./while-loop.sh

Sample output:

./while-loop.sh

1

2

3

4

5

6. Create a script with for loop

```
#!/bin/bash
for (( counter=10; counter>0; counter-- ))
do
echo -n "$counter "
done
printf "\n"
```

i. Create for-loop.sh file and add the above code.

ii. Type chmod +x for-loop.sh

iii. Execute the script ./for-loop.sh

Sample output:

./for-loop.sh

10 9 8 7 6 5 4 3 2 1

7. Create a script to test if statement

Here, 10 is assigned to the variable, n. if the value of $n is less than 10 then the output will be “It is a one digit number”, otherwise the output will be “It is a two digit number”. For comparison, ‘-lt’ is used here. For comparison, you can also use ‘-eq’ for equality, ‘-ne’ for not equality and ‘-gt’ for greater than in bash script.

```
#!/bin/bash
n=10
if [ $n -lt 10 ];
then
echo "It is a one digit number"
else
echo "It is a two digit number"
fi
```

i. Create sample-if.sh file and add the above code.

ii. Type chmod +x sample-if.sh

iii. Execute the script ./sample-if.sh

Sample output:

./sample-if.sh

It is a two digit number

8. Create a shell script to test if and AND conditions

```
!/bin/bash

echo "Enter username"
read username
echo "Enter password"
read password

if [[ ( $username == "admin" && $password == "secret" ) ]]; then
echo "valid user"
else
echo "invalid user"
fi
```

i. Create if-And.sh file and add the above code.

ii. Type chmod +x if-And.sh

iii. Execute the script ./if-And.sh

Sample output 1:

./if-and.sh

Enter username

admin

Enter password

secret

valid user

Sample output 2:

./if-and.sh

Enter username

admin

Enter password

admin

invalid user

9. Create a shell script to test if and OR conditions

'||' is used to define OR logic in if condition. Create a file named 'if_with_OR.sh' with the following code to check the use of OR logic of if statement. Here, the value of n will be taken from the user. If the value is equal to 15 or 45 then the output will be "You won the game", otherwise the output will be "You lost the game".

```
#!/bin/bash

echo "Enter any number"
read n

if [[ ( $n -eq 15 || $n  -eq 45 ) ]]
then
echo "You won the game"
else
echo "You lost the game"
fi
```

i. Create if-OR.sh file and add the above code.

ii. Type chmod +x if-OR.sh

iii. Execute the script ./if-OR.sh

Sample output 1:

./if-OR.sh

Enter any number

15

You won the game

Sample Output 2:

./if-OR.sh

Enter any number

10

You lost the game

10. Create a script to test if and else condition

The use of else if condition is little different in bash than other programming language. ‘elif’ is used to define else if condition in bash. Create a file named, ‘elseif_example.sh’ and add the following script to check how else if is defined in bash script.

```
#!/bin/bash

echo "Enter your lucky number"
read n

if [ $n -eq 101 ];
then
echo "You got 1st prize"
elif [ $n -eq 510 ];
then
echo "You got 2nd prize"
elif [ $n -eq 999 ];
then
echo "You got 3rd prize"

else
echo "Sorry, try for the next time"
fi
```

i. Create if-else.sh file and add the above code.

ii. Type chmod +x if-else.sh

iii. Execute the script ./if-else.sh

Sample output 1:

./if-else.sh

Enter your lucky number

101

You got 1st prize

Sample output 2:

./if-else.sh

Enter your lucky number

100

Sorry, try for the next time

11. Create a script to work with 'awk' command

Awk is a programming language which allows easy manipulation of structured data and the generation of formatted reports. Awk stands for the names of its authors "**A**ho, **W**einberger, and **K**ernighan".

Some of the key features of Awk are as follows:

- Awk views a text file as records and fields
- Like common programming language, Awk has variables, conditionals and loops
- Awk has arithmetic and string operators
- Awk can generate formatted reports

WHAT CAN WE DO WITH AWK?

1. AWK Operations:

(a) Scans a file line by line

(b) Splits each input line into fields

(c) Compares input line/fields to pattern

(d) Performs action(s) on matched lines

2. Useful For:

(a) Transform data files

(b) Produce formatted reports

3. Programming Constructs:

(a) Format output lines

(b) Arithmetic and string operations

(c) Conditionals and loops

Lab:

Consider the following text file as the input file for all cases below.

```
$cat > employee.txt
ajay manager account 45000
sunil clerk account 25000
varun manager sales 50000
amit manager account 47000
tarun peon sales 15000
deepak clerk sales 23000
```



```
sunil peon sales 13000
satvik director purchase 80000
$ awk '{print}' employee.txt
```

Output:

```
ajay manager account 45000
sunil clerk account 25000
varun manager sales 50000
amit manager account 47000
tarun peon sales 15000
deepak clerk sales 23000
sunil peon sales 13000
satvik director purchase 80000
```

In the above example, no pattern is given. So the actions are applicable to all the lines. Action print without any argument, prints the whole line by default, so it prints all the lines of the file without failure.

```
$ awk '/manager/ {print}' employee.txt
```

Output:

```
ajay manager account 45000
varun manager sales 50000
```

```
amit manager account 47000
```

In the above example, the awk command prints all the line which matches with the ‘manager’.

Splitting a Line into Fields: For each record, i.e., line, the awk command splits the record delimited by whitespace character by default and stores it in the $n variables. If the line has 4 words, it will be stored in $1, $2, $3 and $4 respectively. Also, $0 represents the whole line.

```
$ awk '{print $1,$4}' employee.txt
```

Output:

```
ajay 45000
sunil 25000
varun 50000
amit 47000
tarun 15000
deepak 23000
sunil 13000
satvik 80000
```

12. Create a script to work with ‘sed’ command

SED command in UNIX stands for stream editor and it can perform lots of function on file like, searching, find and replace, insertion or deletion.

Lab:

Example:

Consider the below text file as an input.

$cat > file.txt

```
unix is great os. unix is opensource. unix is free os.

learn operating system.

unix linux which one you choose.

unix is easy to learn.unix is a multiuser os.Learn unix
.unix is a powerful.
```

1. **Replacing or substituting string:** Sed command is mostly used to replace the text in a file. The below simple sed command replaces the word “unix” with “linux” in the file.

$sed 's/unix/linux/' file.txt

Output:

```
linux is great os. unix is opensource. unix is free os.

learn operating system.

linux linux which one you choose.

linux is easy to learn.unix is a multiuser os.Learn unix
.unix is a powerful.
```

2. **Replacing the nth occurrence of a pattern in a line:** Use the /1, /2, etc., flags to replace the first, second occurrence of a pattern in a line. The below

command replaces the second occurrence of the word “unix” with “linux” in a line.

```
$sed 's/unix/linux/2' file.txt
```

Output:

```
unix is great os. linux is opensource. unix is free os.

learn operating system.

unix linux which one you choose.

unix is easy to learn.linux is a multiuser os.Learn unix
.unix is a powerful.
```

3. **Replacing all the occurrence of the pattern in a line:** The substitute flag /g (global replacement) specifies the sed command to replace all the occurrences of the string in the line.

```
$sed 's/unix/linux/g' file.txt
```

Output :

```
linux is great os. linux is opensource. linux is free os.

learn operating system.

linux linux which one you choose.

linux is easy to learn.linux is a multiuser os.Learn linux
.linux is a powerful.
```


4. **Replacing string on a specific line number:** You can restrict the sed command to replace the string on a specific line number. An example is

```
$sed '3 s/unix/linux/' geekfile.txt
```

Output:

```
unix is great os. unix is opensource. unix is free os.
learn operating system.
linux linux which one you choose.
unix is easy to learn.unix is a multiuser os.Learn unix
.unix is a powerful.
```

The above sed command replaces the string only on the third line.

13. Process automation

Create a file http.sh and add below code to it. This script will check if http server is up and running. It will start if http service is down, it if is already up, it will just notify that webserver is already running.

```
chmod +x http.sh
#!/bin/bash
http=`ps -ef|grep -i httpd | grep -v grep| wc -l`
if [ $http -eq 0 ]
then
/etc/init.d/httpd start
```

```
else
echo .Webserver is up.
Fi
```

2.2 Adding New Users, Backups and Syslog Files

2.2.1 User accounts

There are three types of user accounts on a Linux system −

Root account

This is also called superuser and would have complete admin rights of the system. A superuser can run any command on the system without any restriction. This user should be assumed as a system administrator.

System accounts

System accounts are those needed for the operation of system-specific components for example mail accounts and the sshd accounts. These accounts are usually needed for some specific function on your system, and any modifications to them could adversely affect the system.

User accounts

User accounts provide interactive access to the system for users and groups of users. General users are typically assigned to these accounts and usually have limited access to critical system files and directories.

Unix supports a concept of Group Account which logically groups a number of accounts. Every account would be a part of another group account. A Unix group plays important role in handling file permissions and process management.

Managing Users and Groups

There are four main user administration files −

- /etc/passwd − Keeps the user account and password information. This file holds the majority of information about accounts on the Unix system.
- /etc/shadow − Holds the encrypted password of the corresponding account. Not all the systems support this file.
- /etc/group − This file contains the group information for each account.
- /etc/gshadow − This file contains secure group account information.

Lab: **Create a new user in Linux**

Type this command to create a new group called test: `$groupadd test`

Run below command to create a user called “newuser” and add it to test group by setting home directory to /home/newuser and assign Bourne shell:

```
$useradd -d /home/newuser -g test -s /bin/sh newuser
```

Set Password for newuser by typing below command:

```
$ passwd newuser
```

Sample output of this command:

```
$ passwd newuser
```

Changing password for user newuser.

New password:

Retype new password:

passwd: all authentication tokens updated successfully.

2.2.2 Backups in Linux

One of the biggest challenges for Linux administrator is implementing a dependable backup system.

The first data that you need to save is the metadata. Metadata is the data about that data, that is, the information about how the system is physically configured. In this case, it's how the operating system disk is partitioned. This information is not typically included in a normal backup. Making a copy of the MBR and partition table is a way to maintain this information in a format that can be used to recreate the root disk partition.

The MBR and partition table are contained in the first 512 bytes of your hard drive. An MBR has three parts: the boot block is stored in the first 446 bytes, the

paritition table is stored in the next 64 bytes, and the boot code signature takes up the remaining 2 bytes.

To save the disk partition information, run these commands:

Lab:

Create a directory /backups

```
$ fdisk -l >/backups/fdisk.txt
$ cp /etc/fstab /backups/fstab.txt
```

To back up the MBR and partition table, run the following command. It creates a backup to a file called */backups/mbr*.

```
$ dd if=/dev/sdb of=/backups/mbr bs=512 count=1
```

This can be used later to restore the MBR and the partition table.

2.2.3 Syslog in Linux

Log files related to daemons and system processes are located in the /var/log directory. These log files use a standard protocol called **syslog**, handled by the syslogd daemon. Every standard application makes use of syslogd to log information. This recipe describes how to use syslogd to log information from a shell script.

Log files help you deduce what is going wrong with a system. It is a good practice to log progress and actions with log file messages. The logger command will place data into log files with syslogd.

These are some of the standard Linux log files. Some distributions use different names for these files:

Log file	Description
`/var/log/boot.log`	Boot log information
`/var/log/httpd`	Apache web server log
`/var/log/messages`	Post boot kernel information
`/var/log/auth.log` `/var/log/secure`	User authentication log
`/var/log/dmesg`	System boot up messages
`/var/log/mail.log` `/var/log/maillog`	Mail server log
`/var/log/Xorg.0.log`	X server log

Lab:

The logger command allows scripts to create and manage log messages:

Place a message in the syslog file /var/log/messages:

```
$ logger LOG_MESSAGE
$ logger This is a test log line
$ tail -n 1 /var/log/messages
```

Sample output:

Dec 15 11:53:22 newuser root: This is a test log line

The /var/log/messages log file is a general purpose log file. When the logger command is used, it logs to /var/log/messages by default.

The -t flag defines a tag for the message:

```
$ logger -t TAG This is a message
$ tail -n 1 /var/log/messages
```

Sample output:

Dec 15 11:54:55 newuser TAG: This is a message

Type below command to get a list of man pages for syslog:

```
$ man -k syslog
```

Sample output:

logger (1) - a shell command interface to the syslog(3) system l...

rsyslog.conf (5) - rsyslogd(8) configuration file

rsyslogd (8) - reliable and extended syslogd

syslog (2) - read and/or clear kernel message ring buffer; set c...

syslog (3) - send messages to the system logger

vsyslog (3) - send messages to the system logger

2.3 Configuring DNS Server

Requirements for this section:

Ubuntu 14.0 +

A DNS is a domain name system server that answers queries about domain names by providing the relative IP address. It is a must for www working; without it, users will have to learn the IP addresses of every website that they want to visit and type them manually in their browser (or add them manually to host files), which is impossible.

There are a lot of programs that provide DNS services under Ubuntu, but the most common one is BIND.

2.3.1 The BIND Installation

Lab:

Install BIND on Ubuntu using apt-get.

You can either use the tasksel utility, which will automatically install for you the two bind9 and bind9-doc packages. Alternatively, you can use the apt-get tool, as follows:

```
$sudo apt-get update
$sudo apt-get install bind9 bind9-doc
```


Once the BIND packages are installed, it is fully configured and functional with the default parameters that allow it to work as a caching DNS for recursive queries. You can of course customize it to fit your needs. We will discuss how to do that in the following sections.

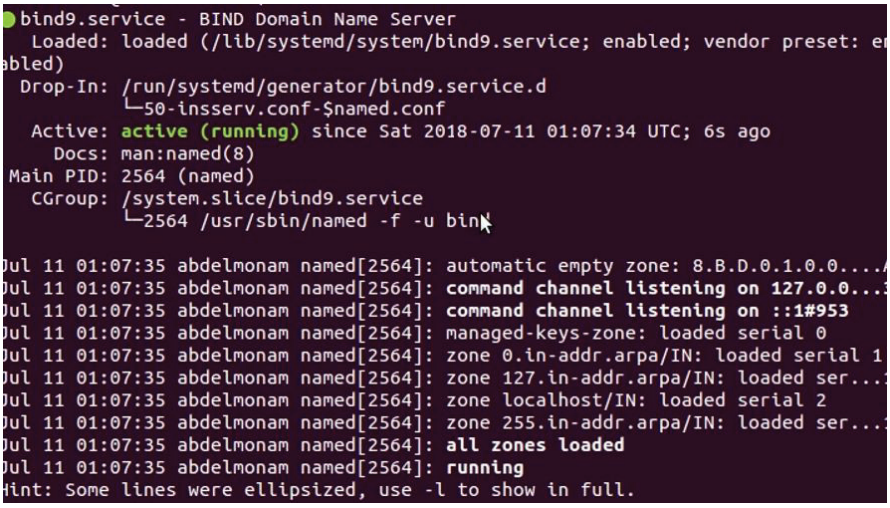
You can check BIND's status by using the following command:

```
$sudo service bind9 status
```

In case it is not running, you can launch it by using the following command:

```
$sudo service bind9 start
```

The output of this command is shown in the following screenshot:



2.3.2 UBUNTU'S BIND Conventions

BIND files may have an organization in Ubuntu that is different from the ones in other distributions. The following are the basic files and directories:

/etc/bind: This is the main directory for the BIND configuration files. It also contains the zonefiles.

/etc/bind/named.conf: This is BIND's main configuration file. It includes scripts from other files, with the most notable ones
being: /etc/bind/named.conf.default-zones, which contains a default zone such as local host; /etc/bind/named.conf.local, which contains zones added by the system administrator (use this file when you would like to add zones);
and /etc/bind/named.conf.options, which contains additional options such as forwarders' addresses.

/etc/bind/db.*: The zone files, which contain information related to each zone, have the convention of starting with db., and each file identifies a particular zone.

/etc/init.d/bind9: This is the file that contains the BIND startup parameters that are needed to launch the service.

/var/log/syslog: By default, BIND uses this file to store its logs. This file contains the log of a lot of other services. So, if you would like to have only the BIND log, you can simply search for the named keyword using the grep command, as follows:

Type the below command to view the Bind log:

```
$grep named /var/log/syslog
```


2.3.3 Configuring BIND

- Configuring BIND is simple. It is always about creating new zones (or editing the existing ones). This is done by performing the following three steps:
- Either create a zone file relative to the zone that needs to be added, or edit an existing one.
- Add a reference to the zone file in /etc/bind/named.conf.local
- Reload the BIND service by using the following command:

```
$sudo service bind9 reload
```

2.3.4 Zone File Configuration

The details related to a specific domain name are stored in a DNS server in an entity called zone. Each zone is represented by a file under the /etc/bind/ directory. In this section, we will take a look at how to configure a DNS zone file.

The zone files have the same structure. So, an easy way of configuring a new zone is by copying an existing zone file (such as /etc/bind/db.local) and then modifying it as required.

By default, timers in the zone file are set in seconds. You can use other time units such as the day by adding the letter d to the value, the week by adding the letter w, the hour using the letter h, and and so on.

The following are the most important fields in the zone file configuration:

- **SOA**: **Start of Authority**, this server is considered to be the best source of information
- **NS**: This is the name servers for the zone file
- **A**: This is IPv4 address
- **AAAA**: This is IPv6 address
- **MX**: This is the mail server

You also need to configure a reversed DNS file for every zone that you create. The structure is also the same, so you can just copy one of the default reversed zones, such as /etc/bind/db.127.

To reference a zone file inside /etc/bind/named.conf.local, you need to just copy a sample of an existing reference and modify it.

Lab:

Edit /etc/bind/named.conf.local and add the below configuration:

```
$ vi /etc/bind/named.conf.local
zone "localhost" {
  type master;
  file "/etc/bind/db.local";
};
zone "127.in-addr.arpa" {
```



```
  type master;
  file "/etc/bind/db.127";
};
```

When you reload BIND, it will take into consideration the modified zone file that has a higher serial number. So, be sure to increment the serial field in the zone file each time you modify it. If you don't do this, your modifications will be ignored. Most system administrators use the yyyyMMddHHmm format for this field.

2.3.5 DNS Redundancy

DNS is a critical service for web and e-mail applications. So, it is important to have a redundancy of the main DNS server (called **salve**), which has the same information as the **master** DNS server.

To configure this, you have to add the salve server to the zone file in the master server by using the NS record. Then, when referencing the zone file, you should add the allow-transfer option to the zone definition inside the master DNS to allow the slave DNS to copy the data from it. You can also add the also-notify option so that the master DNS will notify the slave DNS for every modification inside that zone. On the other hand, in the slave server, the zone definition should contain the slave type and point to the master server. As an example, if we assume that the IP address of the master DNS is *192.168.1.1* and the IP address of the slave DNS is *192.168.1.2*, we should get the following output:

On the master server you will find the following output:

```
zone "localhost" {
 type master;
 file "/etc/bind/db.local";
 allow-transfer { 192.168.1.2; };
 also-notify { 192.168.1.2; };
};
zone "127.in-addr.arpa" {
 type master;
 file "/etc/bind/db.127";
 allow-transfer { 192.168.1.2; };
 also-notify { 192.168.1.2; };
};
```

On the slave server you will find the following output:

```
zone "localhost" {
 type slave;
 file "/etc/bind/db.local";
 masters { 192.168.1.1; };
};
zone "127.in-addr.arpa" {
```



```
 type slave;
 file "/etc/bind/db.127";
 masters { 192.168.1.1; };
};
```

2.3.6 DNS Testing

To test whether your DNS is working, you should request to server and check the correctness of the information. To do this, you should either use a computer in which you point your name server to this DNS, or simply add the IP address of that DNS server as a parameter.

The famous NsLookup tool is the simplest tool to test DNS and reverse DNS.

2.4 The Network File System

Requirements for this section:

Ubuntu 14.0 +

A **Network File System** (**NFS**) is a great method of sharing files from a Linux or UNIX server to a Linux or UNIX server. Windows systems can access NFS shares as well, but there may be an additional licensing penalty if you need to upgrade to a different release. NFS is preferred in a Linux or UNIX environment though, since it fully supports Linux- and UNIX-style permissions. Samba can certainly support per-user access restrictions and benefit greatly from a centralized directory server. NFS is a bit more involved to set up, but in the long run, I think it's easier and integrates better.

While it wasn't mandatory to have a special parent directory with Samba, NFS really does want its own directory to house all of its shares. I this section we will use /exports as an example, so you should make sure that directory exists:

Lab:

```
$sudo apt-get update
$sudo mkdir /exports
```

Next, let's install the required NFS packages on our server. The following command will install NFS and its dependencies:

```
$sudo apt install nfs-kernel-server
```

Once you install the nfs-kernel-server package, the nfs-kernel-server daemon will start up automatically. It will also create a default /etc/exports file (which is the main file that NFS reads its share information from), but it doesn't contain any useful settings, just some commented lines. Let's back up the /etc/exports file, since we'll be creating our own:

```
$Su
$do mv /etc/exports /etc/exports.orig
```

To set up NFS, let's first create some directories that we will share to other users. Each share in NFS is known as an **Export**. Let’s use the following directories as examples, but you can export any directory you like:

/exports/backup

/exports/documents

/exports/public

In the /etc/exports file, insert the following four lines:

```
$Sudo vi /etc/exports

/exports *(ro,fsid=0,no_subtree_check)

/exports/backup
192.168.1.0/255.255.255.0(rw,no_subtree_check)

/exports/documents
192.168.1.0/255.255.255.0(ro,no_subtree_check)

/exports/public
192.168.1.0/255.255.255.0(rw,no_subtree_check)
```

The first line is **Export Root**. The next three lines are individual shares or exports. The backup, documents, and public directories are being shared from the /exports parent directory. Each of these lines is not only specifying which directory is being shared with each export, but also which network is able to access them. In this case, after the directory is called out in a line, we're also setting which network is able to access them (192.168.1.0/255.255.255.0 in our case). This means that if you're connecting from a different network, your access will be denied. Each connecting machine must be a member of the 192.168.1.0/24 network in order to proceed (so make sure you change this to match your IP scheme). Finally, we include some options for each export, for example, rw,no_subtree_check.

The next option in each example is no_subtree_check. This option is known to increase reliability and is mainly implied by default. However, not including it may

make NFS complain when it restarts, but nothing that will actually stop it from working. Particularly, this option disables what is known as **subtree checking**, which has had some stability issues in the past. Normally, when a directory is exported, NFS might scan parent directories as well, which is sometimes problematic, and can cause issues when it comes to open file handles.

There are several other options that can be included in an export, and you can read more about them by checking the man page for export:

```
$man export
```

One option you'll see quite often in the wild is no_root_squash. Normally, the root user on one system is mapped to nobody on the other for security reasons. In most cases, one system having root access to another is a bad idea. The no_root_squash option disables this, and it allows the root user on one end to be treated as the root user on the other. I can't think of a reason, personally, where this would be useful (or even recommended), but I have seen this option quite often in the wild, so I figured I would bring it up. Again, check the man pages for export for more information on additional options you can pass to your exports.

Next, we have one more file to edit before we can actually seal the deal on our NFS setup. The /etc/idmapd.conf file is necessary for mapping permissions on one node to another. In Chapter 2, *Managing Users*, we talked about the fact that each user has an ID (UID) assigned to them. The problem, though, is that from one system to another, a user will not typically have the same UID. For example, user jdoemay be UID 1001 on server **A**, but 1007 on server **B**. When it comes to NFS, this greatly confuses the situation, because UIDs are used in order to

reference permissions. Mapping IDs with idmapd allows this to stay consistent and handles translating each user properly, though it must be configured correctly and consistently on each node. Basically, as long as you use the same domain name on each server and client and configure the /etc/idmapd.conf file properly on each, you should be fine.

To configure this, open /etc/idmapd.conf in your text editor. Look for an option that is similar to the following:

```
$sudo vi /etc/idmapd.conf
#sudo Domain = localdomain
```

First, **remove the # symbol** from that line to uncomment it. Then, change the domain to match the one used within the rest of your network. Basically, just be as consistent as you can and you'll have a much easier time overall. You'll also want to edit the /etc/idmapd.conf file on each node that will access your file server, to ensure they are configured the same as well.

With our /etc/exports and /etc/idmapd.conf files in place, and assuming you've already created the exported directories on your filesystem, we should be all set to restart NFS to activate our configuration:

```
$sudo systemctl restart nfs-kernel-server
```

After restarting NFS, you should check the daemon's output via systemctl to ensure that there are no errors:

```
$systemctl status -l nfs-kernel-server
```

As long as there are no errors, our NFS server should be working. Now, we just need to learn how to mount these shares on another system. Unlike Samba, using a Linux file manager and browsing the network will not show NFS exports; we'll need to mount them manually. Client machines, assuming they are Debian based (Ubuntu fits this description) will need the nfs-common package installed in order to access these exports:

```
$sudo apt install nfs-common
```

With the client installed, we can now use the mount command to mount NFS exports on a client. For example, with regards to our documents export, we can use the following variation of the mountcommand to do the trick:

```
$ sudo mount myserver:/documents /mnt/documents
```

Replace myserver with either your server's hostname or its IP address. From this point forward, you should be able to access the contents of the documents export on your file server. Notice, however, that the exported directory on the server was /exports/documents, but we only asked for /documents instead of the full path with the example mount command. The reason this works is because we identified an export root of /exports. To save you from flipping back, here's the first line from the /etc/exports file, where we identified our export root:

/exports *(ro,fsid=0,no_subtree_check)

With the export root, we basically set the base directory for our NFS exports. We set it as read-only (ro), because we don't want anyone making any changes to the /exports directory itself.

So, at this point, you'll have three directories being exported from your file server, and you can always add others as you go. You can restart NFS to activate new exports, but that's not really a good idea while users may be connected to them, since that will disrupt their access. Thankfully, the following command will cause NFS to reread the /etc/exports file without disrupting existing connections. This will allow you to activate new exports immediately without having to wait for users to finish what they're working on:

```
$sudo exportfs -a
```

2.5 Sending a test email using standard SMTP commands

Requirements for this section :

Ubuntu 14.04 VM and a user who has root privileges

Steps:

#install mail utils

apt-get install mailutils

Let us send a test email from your machine

root@server: mail -s "This is the subject" xyz@abc.com <<< 'This is the message'

Check the logs of the mail utility

root@server: tail /var/log/mail.log

Jan 24 08:57:27 server postfix/master[66570]: daemon started -- version 3.2.2, configuration /etc/postfix

Jan 24 08:57:28 server postfix/pickup[66571]: 139C218CA6FC: uid=502 from=<adminz>

Jan 24 08:57:28 server postfix/cleanup[66573]: 139C218CA6FC: message-id=<20190124032728.139C218CA6FC@Admins-MacBook-Pro-4.local>

Jan 24 08:57:28 server postfix/qmgr[66572]: 139C218CA6FC: from=<admin@server.local>, size=381, nrcpt=1 (queue active)

Jan 24 08:57:31 server/smtp[66575]: 139C218CA6FC: to=<xyz@abc.com>, relay=aspmx.l.google.com[74.125.68.26]:25, delay=4.3, delays=0.46/0.05/2.8/1, dsn=2.0.0, status=sent (250 2.0.0 OK 1548300451 t184si20651639pfb.22 - gsmtp)

Jan 24 08:57:31 server postfix/qmgr[66572]: 139C218CA6FC: removed

You should see the email in your inbox. Don’t forget to check the spam folder.

2.6 Configuring mail server

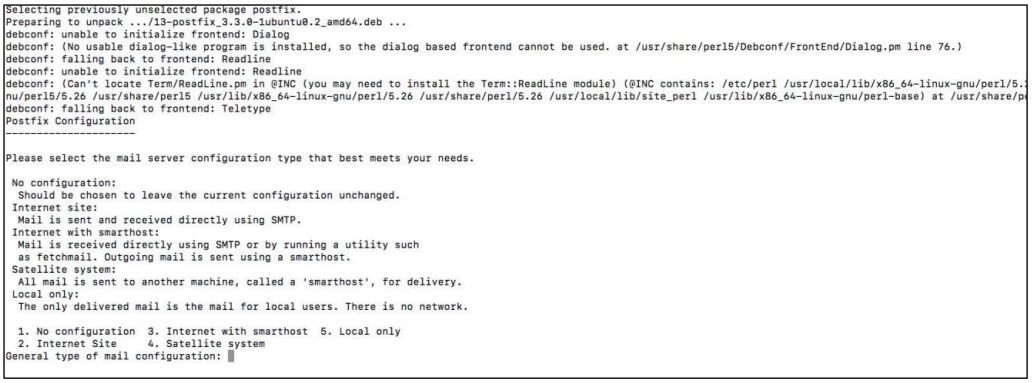
Requirements for this section :

Ubuntu 14.04 VM and a user who has root privileges

Steps:

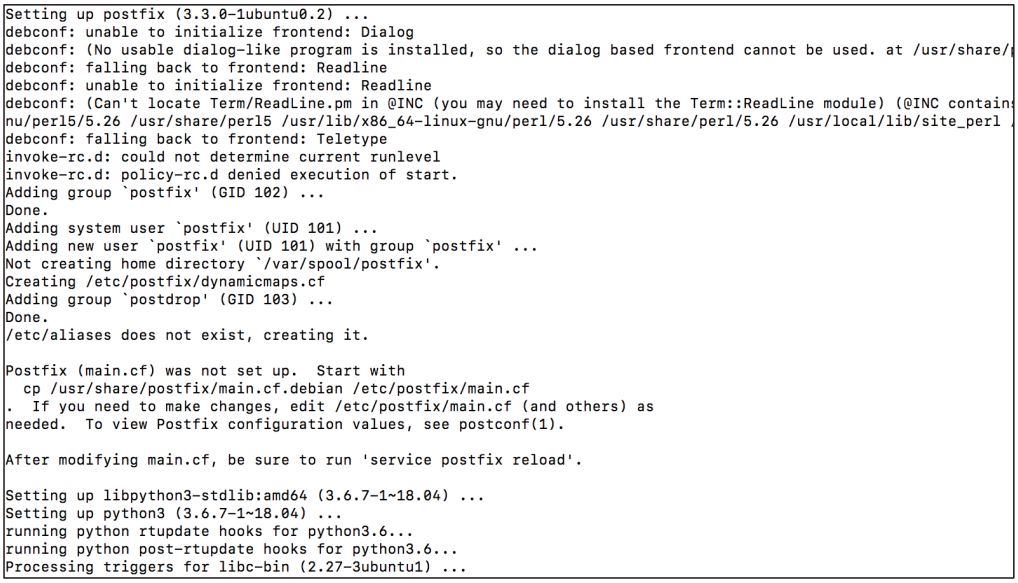
1. **Install postfix and configure it**
    a. **apt-get install postfix**

b. You should something like this(or a colored screen with similar options):



c. Choose No configuration (option 1)

d. This will not set up any mail config and you should see a message like this:



e. We now need to modify /etc/postfix/main.cf file

f. Do the following:

i. cp /usr/share/postfix/main.cf.debian /etc/postfix/main.cf

ii. vi /etc/postfix/main.cf , Modify the file to reflect the following settings.

```
#myorigin = /etc/mailname

myhostname=mail.alphabet.com
mydomain=alphabet.com
myorigin=$mydomain
mydestination = $myhostname, localhost.$mydomain, $mydomain, mail.$mydomain, www.$mydomain
mail_spool_directory = /var/spool/mail
mynetworks = 127.0.0.0/8, 192.168.1.0/24
inet_protocols = ipv4
smtpd_banner = $myhostname ESMTP $mail_name (Ubuntu)
biff = no

# appending .domain is the MUA's job.
append_dot_mydomain = no

# Uncomment the next line to generate "delayed mail" warnings
#delay_warning_time = 4h

readme_directory = no

# See http://www.postfix.org/COMPATIBILITY_README.html -- default to 2 on
# fresh installs.
compatibility_level = 2
root@80d20f9dd26b:/#
```

iii. do *service postfix start* to start the mail server

2. Send test email

a. mail -s "This is the subject" xyz@abc.com <<< 'This is the message'
#check inbox and spam folder

2.7 Package management – yum/apt

Requirements for this section :

Ubuntu 14.04 VM and a user who has root privileges

In-order to install a new package following the steps given below :

Steps:

apt-get install <package-name>

Eg: sudo *apt-get install nginx*

This command will install nginx if it is not present and update it to the latest version if it present.

However if you just update a package which is already installed, you should do the following:

sudo apt-get --only-upgrade install nginx

This command will update the already install nginx package. But, if nginx is not present it will simply skip it. Therefore, you should always execute this command

sudo apt-get install <package-name>

In order to remove a package excute the following command

sudo apt-get remove <package-name>

2.8 Configuring login banner message

Requirements for this section :

Ubuntu 14.04 VM and a user who has root privileges

Here are the steps for Configuring login banner message

root@server: touch /etc/update-motd.d/cat 98-my-message #Create message file

root@server: chmod 777 /etc/update-motd.d/98-my-message #Set permissions

The contents of the file should be as following:

root@server:# cat /etc/update-motd.d/98-my-message

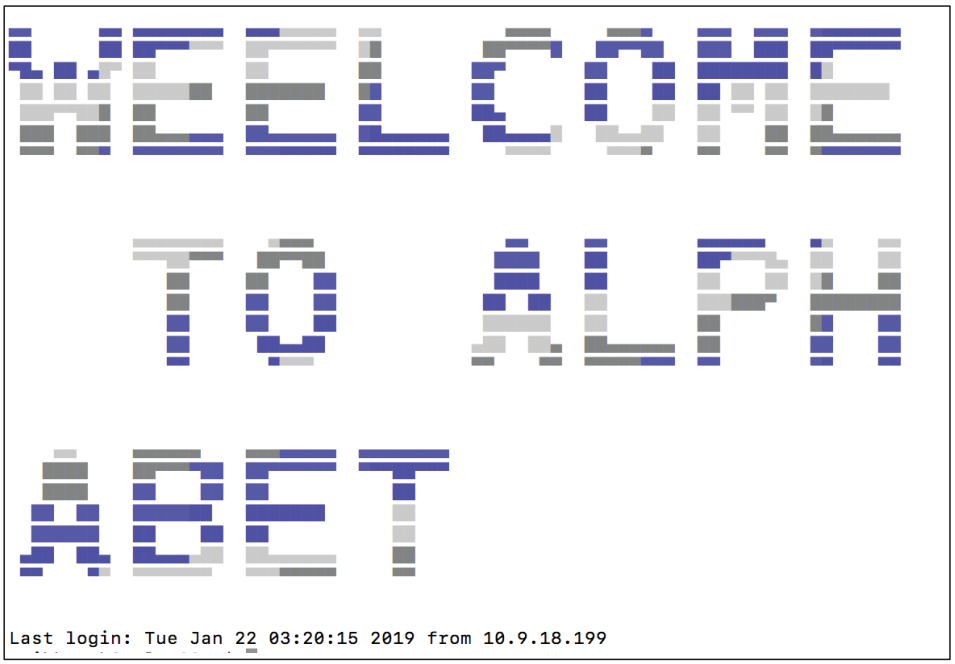
#!/bin/bash

toilet -f mono8 -F metal WELCOME TO ALPHABET

toilet is used for colored text and providing effects to the message.

After placing the contents save the file and log out of the machine. The next time when you log in, you should see the banner as the message you provided.

```
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 4.4.0-124-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Tue Jan 22 03:18:40 UTC 2019

  System load:  0.0                Processes:           108
  Usage of /:   34.1% of 19.65GB   Users logged in:     0
  Memory usage: 16%                IP address for eth0: 10.20.0.6
  Swap usage:   0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

76 packages can be updated.
5 updates are security updates.

New release '16.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2019.
```

2.9 Adding additional network interfaces

Requirements for this section :

Ubuntu 14.04 VM and a user who has root privileges

For adding additional network interfaces ,attach the network interface to your machine and see if it is available.

root@server: ls /sys/class/net/

eth0 eth1 l0

However when you do ifconfig -a you will only see eth0 and l0, for eth1 to be available

Add the secondary network interface **to /etc/network/interfaces**


```
auto eth1
iface enp0s8 inet dhcp

root@server:~# cat /etc/network/interfaces

# This file describes the network interfaces available on your system

# and how to activate them. For more information, see interfaces(5).

# The loopback network interface

auto lo

iface lo inet loopback

# Source interfaces

# Please check /etc/network/interfaces.d before changing this file

# as interfaces may have been defined in /etc/network/interfaces.d

# NOTE: the primary ethernet device is defined in

# /etc/network/interfaces.d/eth0

# See LP: #1262951

source /etc/network/interfaces.d/*.cfg

#New interface

auto eth1
iface enp0s8 inet dhcp
```


Now execute **sudo service networking restart** and after this the new interface can be seen when you execute **ifconfig -a**

2.10 Converting Linux machine into a Switch using scripts

Use this: https://help.ubuntu.com/community/NetworkConnectionBridge

2.11 Scheduling tasks with Cron and anacron

Requirements for this section:

Ubuntu 14.04 VM and a user who has root privileges

Steps:

You can create a simple script to print date time and make sure permissions are **755**

root@server: cat /tmp/myscrip.sh

#!/bin/bash

echo $(date "+%Y-%m-%d %H:%M:%S") >> /tmp/date.log

root@server: crontab -l #To see currently scheduled tasks

root@server: crontab -e #To add new task. This will open up the crontab console

#Add the following line in the console. This print date time in the /tmp/date.log every minute.

*** * * * * /tmp/myscript.sh**

#Save and exit

```
#check if installed properly

root@sserver:~# crontab -l

# Edit this file to introduce tasks to be run by cron.

#

# Each task to run has to be defined through a single line

# indicating with different fields when the task will be run

# and what command to run for the task

#

# To define the time you can provide concrete values for

# minute (m), hour (h), day of month (dom), month (mon),

# and day of week (dow) or use '*' in these fields (for 'any').#

# Notice that tasks will be started based on the cron's system

# daemon's notion of time and timezones.

#

# Output of the crontab jobs (including errors) is sent through

# email to the user the crontab file belongs to (unless redirected).

#

# For example, you can run a backup of all your user accounts

# at 5 a.m every week with:
```


 # 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/

 # For more information see the manual pages of crontab(5) and cron(8)

#

 # m h dom mon dow   command

*** * * * * /tmp/myscript.sh**

#Now check if crontab is working fine or not.

root@server: tail /tmp/date.log

2019-01-22 03:47:01

2019-01-22 03:48:01

#Date time will be appened to this file every minute. similarly you can schedule your own scripts at any specific time by setting *** options in the crontab.**

2.12 Configuring FTP server

Requirements for this section :

Ubuntu 14.04 VM and a user who has root privileges

Steps:

1. **Install and Configure VsFTPd** **("Very Secure FTP Daemon")**
   a. Make sure port 20 and 21 are open on your machine
   b. `apt-get install vsftpd`
   c. `sudo cp /etc/vsftpd.conf /etc/vsftpd.conf.orig`

d. sudo vi /etc/vsftpd.conf #Add config. It should look like this

```
root@server:~# cat /etc/vsftpd.conf
listen=NO
listen_ipv6=YES
anonymous_enable=NO
local_enable=YES
dirmessage_enable=YES
use_localtime=YES
xferlog_enable=YES
connect_from_port_20=YES
secure_chroot_dir=/var/run/vsftpd/empty
pam_service_name=vsftpd
rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
ssl_enable=NO
root@server:~#
```

e. service vsftpd restart

f. service vsftpd status

```
root@server:~# service vsftpd restart
root@server:~# service vsftpd status
● vsftpd.service - vsftpd FTP server
   Loaded: loaded (/lib/systemd/system/vsftpd.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2019-01-25 05:31:27 UTC; 5s ago
  Process: 4297 ExecStartPre=/bin/mkdir -p /var/run/vsftpd/empty (code=exited, status=0/SUCCESS)
 Main PID: 4306 (vsftpd)
    Tasks: 1
   Memory: 552.0K
      CPU: 4ms
   CGroup: /system.slice/vsftpd.service
           └─4306 /usr/sbin/vsftpd /etc/vsftpd.conf

Jan 25 05:31:27 server systemd[1]: Starting vsftpd FTP server...
Jan 25 05:31:27 server systemd[1]: Started vsftpd FTP server.
root@server:~#
```

2. In order to connect to the ftp server you should have a user on that server. If you have a user on that remote ftp machine, then you can connect as follows:

a. From a remote machine which can access your ftp server on port 21, do
ftp <remote-host-ip>

in our case the ip of the remote host is 10.10.10.100
eg:

```
server-master:~# ftp 10.0.10.100
Connected to 10.0.10.100.
220 (vsFTPd 3.0.3)
Name (10.0.10.100:vaibhavth): test
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
226 Directory send OK.
ftp>
```

2.13 Network Management and Debugging

Requirements for this section:

CentOS 7

Wireshark and GUI to view wireshark snalysis

Networks are critical for any organization, network issues tend to magnify problems.

Network management is the art and science of keeping a network healthy. It generally includes the following tasks:

• Fault detection for networks, gateways, and critical servers

• Schemes for notifying an administrator of problems

• General network monitoring, to balance load and plan expansion

• Documentation and visualization of the network

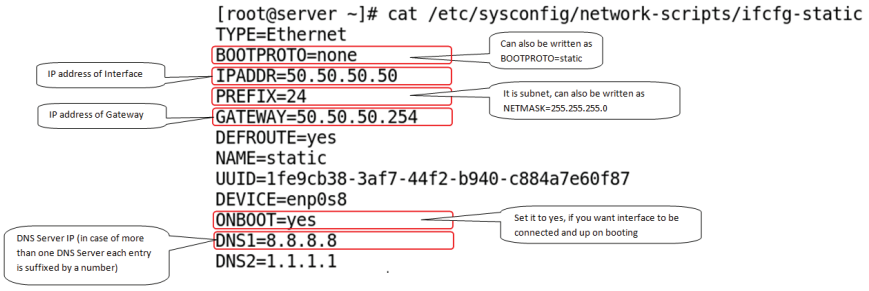
• Administration of network devices from a central site

Unfortunately, even the best network management system cannot prevent all failures. It is critical to have a well-documented network and a high-quality staff available to handle the inevitable collapses.

Network configuration:

We can configure networking manually using network interface configuration files. At any one time, we can either configure networking using network interface files manually or using NetworkManager, but not both simultaneously. The location of the network interface files (scripts) varies from one Linux distribution to another. In the case of CentOS 7, the network interface configuration files are stored in the /etc/sysconfig/network-scripts/ folder and their naming conventions begin with ifcfg-<device_name>

Configuration of the ifcfg file is explained as follows:



2.13.1Network Troubleshooting

Several good tools are available for debugging a network at the TCP/IP layer. Most give low-level information, so you must understand the main ideas of TCP/IP and routing in order to use the debugging tools.

In this section, we start with some general troubleshooting strategy. We then cover several essential tools, including **ping**, **traceroute**, **netstat**, **tcpdump**, and Wireshark.

Network Troubleshooting checklist:

- Do you have physical connectivity and a link light?
- Is your interface configured properly?
- Do your ARP tables show other hosts?
- Is there a firewall on your local machine?
- Is there a firewall anywhere between you and the destination?
- If firewalls are involved, do they pass ICMP ping packets and responses?

- Can you ping the localhost address (127.0.0.1)?
- Can you ping other local hosts by IP address?
- Is DNS working properly?
- Can you ping other local hosts by hostname?
- Can you ping hosts on another network?
- Do high-level services such as web and SSH servers work?

2.13.2Ping – Check to see if a Host is Alive

The **ping** command is embarrassingly simple, but in many situations it is the only command you need for network debugging. It sends an ICMP ECHO_REQUEST packet to a target host and waits to see if the host answers back.

You can use **ping** to check the status of individual hosts and to test segments of the network. Routing tables, physical networks, and gateways are all involved in processing a ping, so the network must be more or less working for **ping** to succeed. If **ping** doesn't work, you can be pretty sure that nothing more sophisticated will work either.

However, this rule does not apply to networks that block ICMP echo requests with a firewall. Make sure that a firewall isn't interfering with your debugging before you conclude that the target host is ignoring a **ping**. You might consider disabling a meddlesome firewall for a short period of time to facilitate debugging.

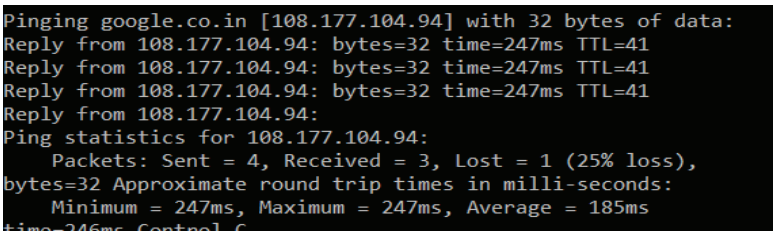
If your network is in bad shape, chances are that DNS is not working. Simplify the situation by using numeric IP addresses when pinging, and use **ping**'s **-n** option to

prevent **ping** from attempting to do reverse lookups on IP addresses—these lookups also trigger DNS requests.

Most versions of **ping** run in an infinite loop unless you supply a packet count argument. Type the interrupt character (usually <Control-C>) to get out.

Lab:

$ ping google.co.in



On Linux systems packet size can be set using –s flag:

```
$ ping -s 1500 cuinfo.cornell.edu
```

2.13.3Traceroute

traceroute, originally written by Van Jacobson, uncovers the sequence of gateways through which an IP packet travels to reach its destination. All modern operating systems come with some version of **traceroute**. The syntax is simple

traceroute *hostname*

There are a variety of options, most of which are not important in daily use. As usual, the *hostname* can be specified as either a DNS name or an IP address. The output is simply a list of hosts, starting with the first gateway and ending at the

destination. For example, a **traceroute** from our host jaguar to our host nubark produces the following output:

```
$ traceroute nubark
traceroute to nubark (192.168.2.10), 30 hops max, 38 byte packets
 1  lab-gw (172.16.8.254)  0.840 ms  0.693 ms  0.671 ms
 2  dmz-gw (192.168.1.254)  4.642 ms  4.582 ms  4.674 ms
 3  nubark (192.168.2.10)  7.959 ms  5.949 ms  5.908 ms
```

traceroute works by setting the time-to-live field (TTL, actually “hop count to live”) of an outbound packet to an artificially low number. As packets arrive at a gateway, their TTL is decreased. When a gateway decreases the TTL to 0, it discards the packet and sends an ICMP “time exceeded” message back to the originating host.

The first three **traceroute** packets have their TTL set to 1. The first gateway to see such a packet (lab-gw in this case) determines that the TTL has been exceeded and notifies jaguar of the dropped packet by sending back an ICMP message. The sender’s IP address in the header of the error packet identifies the gateway, and **traceroute** looks up this address in DNS to find the gateway’s hostname.

To identify the second-hop gateway, **traceroute** sends out a second round of packets with TTL fields set to 2. The first gateway routes the packets and decreases their TTL by 1. At the second gateway, the packets are then dropped and ICMP error messages are generated as before. This process continues until the TTL is equal to the number of hops to the destination host and the packets reach their destination successfully.

When debugging routing issues, it can be helpful to take a look at your site from the perspective of the outside world. Several web-based route tracing services let

you do this sort of inverse **traceroute** right from a browser window. Thomas Kernen maintains a list of these services at traceroute.org.

Let's do traceroute from a host in Switzerland to caida.org at the San Diego Supercomputer Center

```
linux$ traceroute caida.org
traceroute to caida.org (192.172.226.78), 30 hops max, 46 byte packets
 1  gw-oetiker.init7.net (213.144.138.193)  1.122 ms  0.182 ms  0.170 ms
 2  r1zur1.core.init7.net (77.109.128.209)  0.527 ms  0.204 ms  0.202 ms
 3  r1fra1.core.init7.net (77.109.128.250)  18.279 ms  6.992 ms  16.597 ms
 4  r1ams1.core.init7.net (77.109.128.154)  19.549 ms  21.855 ms  13.514 ms
 5  r1lon1.core.init7.net (77.109.128.150)  19.165 ms  21.157 ms  24.866 ms
 6  r1lax1.ce.init7.net (82.197.168.69)  158.232 ms  158.224 ms  158.271 ms
 7  cenic.laap.net (198.32.146.32)  158.349 ms  158.309 ms  158.248 ms
 8  dc-lax-core2--lax-peer1-ge.cenic.net (137.164.46.119)  158.60 ms * 158.71 ms
 9  dc-tus-agg1--lax-core2-10ge.cenic.net (137.164.46.7) 159 ms 159 ms 159 ms
10  dc-sdsc-sdsc2--tus-dc-ge.cenic.net (137.164.24.174) 161 ms 161 ms 161 ms
11  pinot.sdsc.edu (198.17.46.56)  161.559 ms  161.381 ms  161.439 ms
12  rommie.caida.org (192.172.226.78)  161.442 ms  161.445 ms  161.532 ms
```

Lab:

```
$traceroute localhost
```

Sample output:

traceroute to localhost (127.0.0.1), 30 hops max, 60 byte packets

1 localhost (127.0.0.1) 0.046 ms 0.017 ms 0.013 ms

```
$traceroute 8.8.8.8
```

Sample output:

Tracing route to google-public-dns-a.google.com [8.8.8.8]

over a maximum of 30 hops:

1 * * * Request timed out.

2 * * * Request timed out.

3 * * * Request timed out.

4 * * * Request timed out.

5 * * * Request timed out.

6 * * * Request timed out.

7 * * * Request timed out.

8 47 ms 44 ms 44 ms google-public-dns-a.google.com [8.8.8.8]

Trace complete.

C:\Users\newuser>tracert 8.8.8.8

2.13.4Netstat

netstat collects lot of information about network on your machine, including interface statistics, routing information, and connection tables. There isn't really a unifying theme to the different sets of output, except that they all relate to the network. It exposes a variety of network information that doesn't fit anywhere else. Let's discuss the five most common uses of **netstat**:

- Inspecting interface configuration information
- Monitoring the status of network connections

- Identifying listening network services
- Examining the routing table
- Viewing operational statistics for various network protocols

Inspecting Interface Configuration Information

Lab:

netstat -i shows the configuration and state of each of the host's network interfaces along with the associated traffic counters. The output is generally vary by system:

Type below command to view the network interface details:

```
$ netstat –i
```

Sample output:

Kernel Interface table

Iface MTU Met RX-OK RX-ERR RX-DRP RX-OVR TX-OK TX-ERR TX-DRP TX-OVR Flg

eth1 1500 0 528948640 0 3 0 105451514 0 0 0 BMRU

lo 65536 0 299513467 0 0 0 299513467 0 0 0 LRU

netstat –a With no arguments, **netstat** displays the status of active TCP and UDP ports. Inactive ("listening") servers that are waiting for connections are normally hidden, but you can see them with **netstat –a**

```
$netstat –a
```

Sample output:

Active Internet connections (servers and established)

Proto Recv-Q Send-Q Local Address Foreign Address State

tcp 0 0 localhost:32000 *:* LISTEN

tcp 0 0 *:1002 *:* LISTEN

tcp 0 0 *:ssh *:* LISTEN

tcp 0 0 localhost:ipp *:* LISTEN

tcp 0 0 *:smtp *:* LISTEN

netstat –lp **netstat -l** to see only the listening ports. The output format is the same as for **netstat -a**. You can also add the **-p** flag to make **netstat** identify the specific process associated with each listening port.

```
$netstat –lp
```

Sample output:

Active Internet connections (only servers)

Proto Recv-Q Send-Q Local Address Foreign Address State
PID/Program name

tcp 0 0 localhost:32000 *:* LISTEN 12541/java

tcp 0 0 *:1002 *:* LISTEN 1796/python

tcp 0 0 *:ssh *:* LISTEN 1700/sshd

tcp 0 0 localhost:ipp *:* LISTEN 1827/cupsd

netstat -r displays the kernel's routing table. The following sample output is from a Red Hat machine with two network interfaces.

```
$netstat –r
```

Sample output:

Kernel IP routing table

Destination Gateway Genmask Flags MSS Window irtt Iface

16.0.0.0 * 255.255.248.0 U 0 0 0 eth1

link-local * 255.255.0.0 U 0 0 0 eth1

default 16.0.0.1 0.0.0.0 UG 0 0 0 eth1

netstat -s dumps the contents of counters that are scattered throughout the network code. The output has separate sections for IP, ICMP, TCP, and UDP.

```
$netstat –s
```

Sample output:

Ip:

649273586 total packets received

57 with invalid addresses

0 forwarded

0 incoming packets discarded

649270654 incoming packets delivered

404144768 requests sent out

3444 reassemblies required

574 packets reassembled ok

2 fragments received ok

4 fragments created

2.13.5tcpdump

tcpdump, yet another amazing network tool by Van Jacobson, is available as a package for most Linux distributions and can be installed from source on our other example systems. **tcpdump** has long been the industry-standard sniffer, and most other network analysis tools read and write trace files in **tcpdump** format, also known as **libpcap** format.

By default, **tcpdump** tunes in on the first network interface it comes across. If it chooses the wrong interface, you can force an interface with the **-i** flag. If DNS is broken or you just don't want **tcpdump**doing name lookups, use the **-n** option. This option is important because slow DNS service can cause the filter to start dropping packets before they can be dealt with by **tcpdump**.

The **-v** flag increases the information you see about packets, and **-vv** gives you even more data. Finally, **tcpdump** can store packets to a file with the **-w** flag and can read them back in with the **-r** flag.

Lab:

Open 2 separate consoles.

On Console1 type: **`$tcpdump host 8.8.8`**

On Console2 type: **`$ping 8.8.8.8`**

The filter specification **host 8.8.8.8** limits the display of packets to those that directly involve the machine 8.8.8.8, either as source or as destination.

Sample output:

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode

listening on eth1, link-type EN10MB (Ethernet), capture size 65535 bytes

04:20:55.659023 IP cxx.host.net > 8.8.8.8: ICMP echo request, id 59964, seq 1, length 64

04:20:55.906956 IP 8.8.8.8 > cxx.host.net: ICMP echo reply, id 59964, seq 1, length 64

With tcpdump you can collect the details based on host IP/subnet and port number. Sample command is provided below:

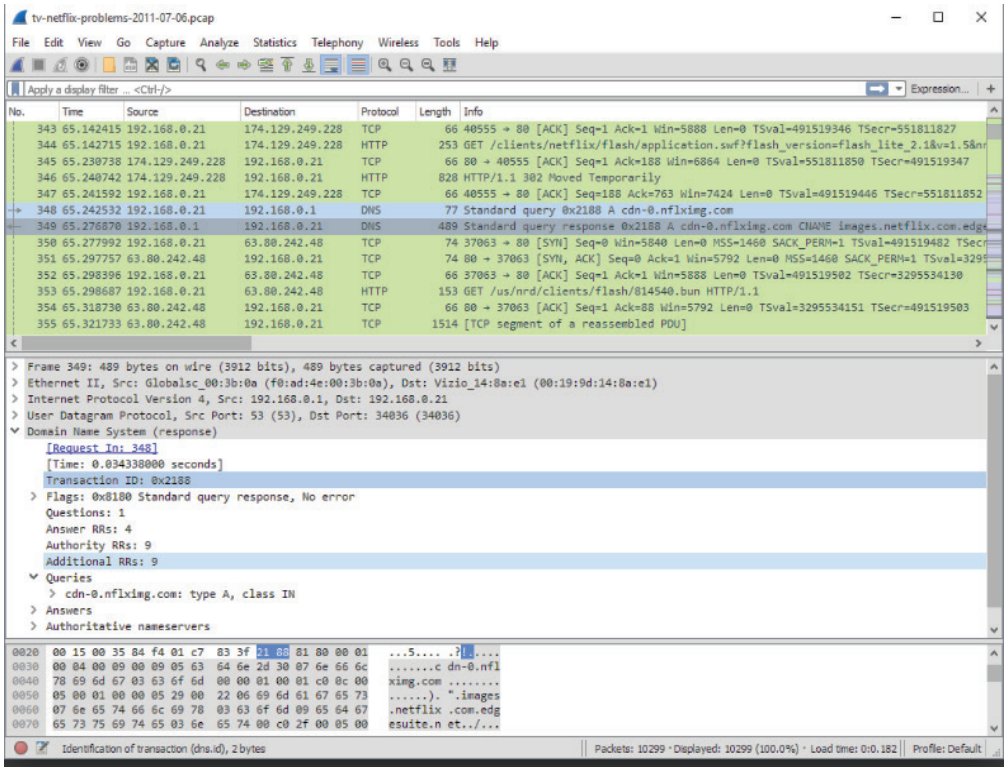
```
$ sudo tcpdump src net 192.168.1.0/24 and dst port 80
```

Analyzing tcdump output using Wireshark

Wireshark is one of the best open source network GUI packet analyzer available today. It is used to capture network packets and display the details of the packet data. Wireshark and tcpdump use libpcap to get live network data. It's often more easy to capture packets using tcpdump command and view using Wireshark. This is useful for troubleshooting the network or network security issues and to debug protocol implementations.

```
$ sudo tcpdump -w output.dump src net 192.168.1.0/24 and dst port 80
```

Open output.dump in wireshark. Sample screenshot is below:



2.13.6What Happens if we Type google.com in the Browser? (Use case)

Step 1. URL is typed in the browser

Step 2. If the requested object is in the browser cache and is fresh, move on to Step 8

Step 3. DNS lookup to find the IP address of the server

When we want to connect to google.com, we actually want to reach out to a server where google web services are hosted. One such server is having an IP address of 74.125.236.65. Now, if you type "http://74.125.236.65" in your browser, this will take you to google home page itself. This means, “http://google.com” and “http://74.125.236.65” are nothing but same stuff. But, it is not so. Google has multiple servers in multiple locations to cater to the huge volume of requests they receive per second. Thus we should let Google decide which server is best suited to our needs. Using "google.com" does the job for us. When we type "google.com", a DNS (Domain Name System) service come into play and resolves the URL to a proper IP address.

Following is a summary of steps happening while DNS service is at work:

Check browser cache: browsers maintain a cache of DNS records for some fixed duration. So, this is the first place to resolve DNS queries.

Check OS cache: if browser doesn't contain the record in its cache, it makes a system call to underlying Operating System to fetch the record as OS also maintains a cache of recent DNS queries.

Router Cache: if the above steps fail to get a DNS record, the search continues to your router which has its own cache.

ISP cache: if everything fails, the search moves on to your ISP. First, it tries in its cache, if not found - ISP's DNS recursive search comes into the picture. DNS lookup is again a complex process which finds the appropriate IP address from a list of many options available for websites like Google. You can read more about this here.

Step 4. Browser initiates a TCP connection with the server

Step 5. Browser sends a HTTP request to the server

The browser sends a GET request to the server according to the specification of HTTP (Hyper Text Transfer Protocol) protocol.

GET http://google.com/ HTTP/1.1

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:29.0) Gecko/20100101 Firefox/29.0

Accept-Encoding: gzip, deflate

Connection: Keep-Alive

Host: google.com

Cookie: datr=1265876274-[...]; locale=en_US; lsd=WW[...]; c_user=2101[...]

Here, the browser passes some meta information in the form of headers to the server along with the URL - "http://google.com". User-Agent header specifies the browser properties, Accept-Encoding headers specify the type of responses it will

accept. The connection header tells the server to keep open the TCP connection established here. The request also contains Cookies, which are meta information stored at the client end and contain previous browsing session information for the same website in the form of key-value pairs e.g. the login name of the user for Google.

Step 6. Server handles the incoming request

HTTP request made from browsers is handled by a special software running on the server - commonly known as web servers, e.g. Apache, IIS, etc. Web server passes on the request to the proper request handler - a program written to handle web services, e.g., PHP, ASP.NET, Ruby, Servlets, etc.

For example URL- http://edusagar.com/index.php is handled by a program written in PHP file - index.php. As soon as GET request for index.php is received, Apache(our webserver at edusagar.com) prepares the environment to execute the index.php file. Now, this php program will generate a response - in our case an HTML response. This response is then sent back to the browser according to HTTP guidelines.

Step 7. Browser receives the HTTP response

HTTP/1.1 200 OK

Cache-Control: private, no-store, no-cache, must-revalidate, post-check=0, pre-check=0

Expires: Thu, 19 Nov 2018 08:52:00 GMT

Pragma: no-cache

Content-Encoding: gzip

Content-Type: text/html; charset=utf-8

Connection: Keep-Alive

Content-length: 1215

Date: Fri, 30 May 2018 08:10:15 GMT

.....<some blob> .....

HTTP response starts with the returned status code from the server. Following is a very brief summary of what a status code denotes:

1xx indicates an informational message only

2xx indicates success of some kind

3xx redirects the client to another URL

4xx indicates an error on the client's part

5xx indicates an error on the server's part

Server sets various other headers to help the browser render the proper content. Content-Type tells the type of the content the browser has to show, Content-length tells the number of bytes of the response. Using the Content-Encoding header's value, browsers can decode the blob data present at the end of the response.

Step 8. Browsers displays the html content

Rendering of html content is also done in phases. The browser first renders the bare bone html structure, and then it sends multiple GET requests to fetch other hyper linked stuff, e.g. If the html response contains an image in the form of img tags such as , browser will send a HTTP GET request to the server to fetch the image following the complete set of steps which we have seen till now. But this isn't that bad as it looks. Static files like images, javascript, css files are all cached by the browser so that in the future it doesn't have to fetch them again.

Step 9. Client interaction with server

Once an html page is loaded, there are several ways a user can interact with the server. For example, he call fill out a login form to sign in to the website. This also follows all the steps listed above, the only difference is that the HTTP request this time would be a POST instead of GET and along with that request, the browser will send the form data to the server for processing (username and password in this case).

Once server authenticates the user, it will send the proper HTML content (may be the user's profile) back to the browser and thus the user will see that new web page after his login request is processed.

Step 10. AJAX queries

Another form of client interaction with the server is through AJAX(Asynchronous JavaScript And XML) requests. This is an asynchronous GET/POST request to which server can send a response back in a variety of ways - json, xml, html etc. AJAX requests doesn't hinder the current view of the webpage and work in the

background. Because of this, one can dynamically modify the content of a webpage by calling an AJAX request and updating the web elements using Javascript.

2.14 Linux Security

Requirements for this section:

CentOS 7

One of the myths about Linux is that it is secure, as it is not susceptible to viruses or other forms of malware. This is partially true, as Linux uses the foundations of the original UNIX operating system. Processes are separated and a normal user is restricted in what he or she can do on the system. Still, Linux is not perfectly secure by default. One of the reasons is the Linux distributions that package the GNU/Linux kernel and the related software. They have to choose between usability, performance, and security.

Areas	Core	Resources	Services	Environment
System Hardening	Boot Process Containers Frameworks **Kernel** Service Manager Virtualization	Accounting **Authentication** Cryptography Logging Network Software Storage Time	Database Mail Middleware Monitoring Printing **Shell** **Web**	Forensics Incident Response Malware Risks Security Monitoring System Integrity
Security Auditing				
Compliance				

System hardening steps

1. Install security updates and patches
2. Use strong passwords

3. Bind processes to localhost
4. Implement a firewall
5. Keep things clean
6. Security configurations
7. Limit access
8. Monitor your systems
9. Create backups (and test!)
10. Perform system auditing

2.14.1Advantages of Using Sudo

The sudo utility can greatly enhance the security of your systems, and it can make an administrator's job much easier. With sudo, you can do the following:

- Assign certain users full administrative privileges, while assigning other users only the privileges they need to perform tasks that are directly related to their respective jobs.
- Allow users to perform administrative tasks by entering their own normal user passwords so that you don't have to distribute the root password to everybody and his brother.
- Make it harder for intruders to break into your systems. If you implement sudo and disable the root user account, would-be intruders won't know which account to attack because they won't know which one has admin privileges.

- Create sudo policies that you can deploy across an entire enterprise network even if that network has a mix of Unix, BSD, and Linux machines.
- Improve your auditing capabilities because you'll be able to see what users are doing with their admin privileges.

Lab: Assigning sudo privileges

Type below command in the console:

```
$sudo useradd user1
$sudo ueradd user2
$sudo useradd user3
$sudo passwd user1
$sudo passwd user2
$sudo passwd user3
```

Type visudo

```
$sudo visudo
```

Add the following lines to the end of the file, using tabs to separate the columns:

user1 ALL=(ALL) ALL

user2 ALL=(ALL) /usr/bin/systemctl status sshd

user3 ALL=(ALL) STORAGE

Save the file and exit visudo.

First, log in to user1's account and verify that he has full sudo privileges by running several root-level commands:

```
$su - user1
 $sudo su -
 $exit
 $sudo systemctl status sshd
 $sudo fdisk -l
 $exit
```

Log in as user2 and try to run some root-level commands.

```
 $su - user2
 $sudo su -
 $sudo systemctl status sshd
 $sudo systemctl restart sshd
 $sudo fdisk -l
 $exit
```

Finally, log in as user3, and run the same set of commands that you ran for user2.

2.14.2Enforcing Strong Password Criteria

- Make passwords of a certain minimum length
- Make passwords that consist of a combination of uppercase letters, lowercase letters, numbers, and special characters

- Ensure that passwords don't contain any words that are found in the dictionary or that are based on the users' own personal data
- Force users to change their passwords on a regular basis

Lab:

Open the `/etc/security/pwquality.conf` file in your preferred text editor. Remove the comment symbol from in front of the minlen line and change the value to 19. It should now look like this:

minlen = 19

Create a user account for newuser and attempt to assign the passwords, welcome, Welcome1, and Welcome1test. Note the change in each warning message.

In the `pwquality.conf` file, comment out the minlen line. Uncomment the minclass line and the maxclassrepeat line. Change the maxclassrepeat value to 5. The lines should now look like:

minclass = 3
maxclassrepeat = 5

Save the file and exit the text editor.

Try assigning various passwords that don't meet the complexity criteria that you've set to newuser account and view the results

Setting and enforcing password and account expiration

To get started, take a look at the expiry data for your own account.

```
$ chage -l newuser
```

Sample output:

[sudo] password for newuser:
Last password change : Dec 13, 2018
Password expires : never
Password inactive : never
Account expires : never
Minimum number of days between password change : 0
Maximum number of days between password change : 99999
Number of days of warning before password expires : 7
newuser@packt:~$

Edit `/etc/login.defs` file and verify the below three relevant lines to enforce strong password policy:

PASS_MAX_DAYS 99999
PASS_MIN_DAYS 0
PASS_WARN_AGE 7

2.14.3Enabling Firewall Through iptables

Type `$sudo iptables -L` to view the current rules

Create the rules that you need for a basic firewall, allowing for Secure Shell access but denying everything else:

Lab:

```
sudo iptables -A INPUT -m conntrack
                                        --ctstate
ESTABLISHED,RELATED
                      -j ACCEPT

sudo iptables -A INPUT -p tcp --dport ssh -j ACCEPT

sudo iptables -A INPUT -j DROP
```

Type `$sudo iptables –L` to view the latest rules

2.14.4Creating Encrypted Backups Using gpg and tar

Type following command:

```
$gpg --gen-key
```

Create some dummy files in your home directory, so that you'll have something to back up:

```
$touch {file1.txt,file2.txt,file3.txt,file4.txt}
```

Create a backup directory at the root level of the filesystem. Change ownership of the directory to your own account, and set the permissions so that only you can access it:

```
      sudo mkdir /backup
        sudo chown your_username: /backup
        sudo chmod 700 /backup
```


Create an encrypted backup file of your own home directory. Compression is optional, but we'll go ahead and use xz for the best compression.

```
cd /home
sudo tar cJvf - your_username/ | gpg -c >
/backup/your_username_backup.tar.xz.gpg
```

Now, let's say that either your home directory got deleted, or that you accidentally deleted some important files from your own home directory. Extract and decrypt the original home directory within the /backupdirectory:

```
cd /backup
sudo gpg -d your_username.tar.xz.gpg | tar xvJ
ls -la your_username/
```

2.14.5Protecting Network Ports

Each network daemon that's running on your system has a specific network port or set of network ports assigned to it, on which it will listen. The /etc/services file contains the list of common daemons and their associated network ports, but it doesn't prevent someone from configuring a daemon to listen on some non-standard port. So, without some mechanism to prevent it, some sneaky intruder could potentially plant some sort of malware that would cause a daemon to listen on a non-standard port, possibly listening for commands from its master.

SELinux protects against this sort of malicious activity by only allowing daemons to listen on certain ports. Use semanage to look at the list of allowed ports:

Use semanage to look at the list of allowed ports:

Lab:

Requirement – A sample webserver running on port 80

Type the blow command and review the output:

```
$ sudo semanage port –l
```

Sample Output:

[sudo] password for user:

SELinux Port Type Proto Port Number
afs3_callback_port_t tcp 7001
afs3_callback_port_t udp 7001
afs_bos_port_t udp 7007
afs_fs_port_t tcp 2040
afs_fs_port_t udp 7000, 7005
afs_ka_port_t udp 7004;

Type the below command to print the ports used by http:

```
[user@localhost ~]$ sudo semanage port -l | grep 'http'
```

Sample output:

[sudo] password for user:

http_cache_port_t tcp 8080, 8118, 8123, 10001-10010

http_cache_port_t udp 3130

Let's go into the `/etc/httpd/conf/httpd.conf` file and look at the ports on which webserver is currently listening. Search for Listen, and you'll see the following line:

Listen 80

Edit `/etc/httpd/conf/httpd.conf` file and change Listen 80 to a port number that SELinux doesn't allow. For example, port 92:

After saving the file, restart Apache to read in the new configuration:

```
$sudo systemctl restart httpd
```

Sample output:

Job for httpd.service failed because the control process exited with error code. See "systemctl status httpd.service" and "journalctl -xe" for details.

Review log in /var/log/messages file

Sample output:

Nov 29 16:39:21 localhost python: SELinux is preventing /usr/sbin/httpd from name_bind access on the tcp_socket port 82.#012#012***** Plugin bind_ports (99.5 confidence) suggests **********#012#012If you want to allow /usr/sbin/httpd to bind to network port 92#012Then you need to modify the port type.#012Do#012# semanage port -a -t PORT_TYPE -p tcp 92#012 where PORT_TYPE is one of the following: http_cache_port_t, http_port_t,

jboss_management_port_t, jboss_messaging_port_t, ntop_port_t, puppet_port_t.#012#012***** Plugin catchall (1.49 confidence) suggests **********#012#012If you believe that httpd should be allowed name_bind access on the port 92 tcp_socket by default.#012Then you should report this as a bug.#012You can generate a local policy module to allow this access.#012Do#012allow this access for now by executing:#012# ausearch -c 'httpd' --raw | audit2allow -M my-httpd#012# semodule -i my-httpd.pp#012

Allow port 92 in selinux

```
$sudo semanage port -a 92 -t http_port_t -p tcp

$ sudo systemctl restart httpd
$ sudo systemctl status httpd
```

Sample output:

● httpd.service - The Apache HTTP Server
Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
Active: active (running) since Wed 2018-12-14 20:09:51 IST; 7s ago
Docs: man:httpd(8)

Release Notes

B. TECH CSE with Specialization in DevOps

Semester Four -Year 02

Release Components.
Facilitator Guide, Facilitator Course Presentations, Student Guide, Mock exams and relevant lab guide.

Current Release Version.
1.0.0

Current Release Date.
6 Feb 2019

Course Description.
Xebia, has been recognized as a leader in DevOps by Gartner and Forrester and this course is created by Xebia to equip students with set of practices, methodologies and tools that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals while automating the process of software delivery and infrastructure changes.

The language used in this course is US English. Our sources of reference for grammar, syntax, and mechanics are from The Chicago Manual of Style, The American Heritage Dictionary, and the Microsoft Manual of Style for Technical Publications.

Bugs reported	Not applicable for version 1.0.0
Next planned release	Version 2.0.0 Sept. 2019