11 – System/Log monitoring commands and System Information/Maintenance Commands

top

- The top command has been around a long time and is very useful for viewing details of running processes and quickly identifying issues such as memory hogs. Its default view is shown below.

top - 11:56:28 up 1 day, 13:37, 1 user, load average: 0.09, 0.04, 0.03
Tasks: 292 total, 3 running, 225 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.1 us, 0.2 sy, 0.0 ni, 99.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 16387132 total, 10854648 free, 1859036 used, 3673448 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 14176540 avail Mem
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
17270 alan 20 0 3930764 247288 98992 R 0.7 1.5 5:58.22 gnome-shell
20496 alan 20 0 816144 45416 29844 S 0.5 0.3 0:22.16 gnome-terminal-
21110 alan 20 0 41940 3988 3188 R 0.1 0.0 0:00.17 top
1 root 20 0 225564 9416 6768 S 0.0 0.1 0:10.72 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.01 kthreadd
4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/0:0H
6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 mm_percpu_wq
7 root 20 0 0 0 0 S 0.0 0.0 0:00.08 ksoftirqd/0

- The update interval can be changed by typing the letter **s** followed by the number of seconds the user prefers for updates.
- To make it easier to monitor required processes, user can call top and pass the PID(s) using the **-p** option.
- **top -p20881 -p20882 -p20895 -p20896**

Tasks: 4 total, 0 running, 4 sleeping, 0 stopped, 0 zombie
%Cpu(s): 2.8 us, 1.3 sy, 0.0 ni, 95.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 16387132 total, 10856008 free, 1857648 used, 3673476 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 14177928 avail Mem
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
20881 alan 20 0 12016 348 0 S 0.0 0.0 0:00.00 nginx
20882 alan 20 0 12460 1644 932 S 0.0 0.0 0:00.00 nginx
20895 alan 20 0 12016 352 0 S 0.0 0.0 0:00.00 nginx

20896 alan 20 0 12460 1628 912 S 0.0 0.0 0:00.00 nginx

df

- The **df** command (short for disk free), is used to display information related to file systems about total space and available space.
- **Syntax :** df [OPTION]... [FILE]...
- If no file name is given, it displays the space available on all currently mounted file systems.

Example 1: df

Portion of output:

Filesystem 1K-blocks Used Available Use% Mounted on

udev 3996816 0 3996816 0% /dev

tmpfs 804624 10020 794604 2% /run

/dev/sda9 68117056 18036160 46597712 28% /

Example 2: df -h /home/tmapp

- Use -h option to display size in power of 1024

Output:

Filesystem Size Used Avail Use% Mounted on /dev/sda10 76G 65G 7.0G 91% /home

Example 3: df -H /home/tmapp

- Use -H option to display sizes in power of 1000

Output:

Filesystem Size Used Avail Use% Mounted on

/dev/sda10 81G 70G 7.5G 91% /home

Observe the size section of two command with -h and -H option for difference.

Example 4: df -T /home/tmapp

- Use -T option to display file type.

Output:

Filesystem Type 1K-blocks Used Available Use% Mounted on

/dev/sda10 ext4 78873504 67528128 7315732

iostat

- The **iostat** command in Linux is used for monitoring system input/output statistics for devices and partitions.
- It monitors system input/output by observing the time the devices are active in relation to their average transfer rates.
- The iostat produce reports may be used to change the system configuration to raised balance the input/output between the physical disks.

Syntax: iostat

- **Options used:**
  - **-x:** This command shows more details statistics information. iostat command gives I/O devices report utilization as a result. So it's possible to extend the statistic result for a diagnose in depth with the -x option.
  - **-c:** This command show only the CPU statistic. It is possible to show the statistic information and report of our cpu with -c option.
  - **-d:** This command displays only the device report. It is possible to only show the status of the device utilization with the help of -d option. It will be going to list information for each connected device.
  - **-k:** This command captures the statistics in kilobytes or megabytes. By default, iostat measure the I/O system with the bytes unit.
  - **-c 2 2:** To show CPU only report with 2 seconds interval and 2 times reports.
- The **first section** contains CPU report:
  - **%user :** It shows the percentage of CPU being utilization that while executing at the user level.
  - **%nice :** It shows the percentage of CPU utilization that occurred while executing at the user level with a nice priority.
  - **%system :** It shows the percentage of CPU utilization that occurred while executing at the system (kernel) level.
  - **%iowait :** It shows the percentage of the time that the CPU or CPUs were idle during which the system had an outstanding disk I/O request.

- **%steal :** It shows the percentage of time being spent in involuntary wait by the virtual CPU or CPUs while the hypervisor was servicing by another virtual processor.
- **%idle :** It shows the percentage of time that the CPU or CPUs were idle and the system did not have an outstanding disk I/O request.

- The **second section** of the output contains device utilization report:

- **Device :** The device/partition name is listed in /dev directory.
- **tps :** The number of transfers per second that were issued to the device. Higher tps means the processor is busier.
- **Blk_read/s :** It shows the amount of data read from the device expressed in a number of blocks (kilobytes, megabytes) per second.
- **Blk_wrtn/s :** The amount of data written to the device expressed in a number of blocks (kilobytes, megabytes) per second.
- **Blk_read :** It shows the total number of blocks read.
- **Blk_wrtn :** It shows the total number of blocks written.

free

- **free** command is used to view memory consumption

- **free -m**

total used free shared buffers cached
Mem: 7976 6459 1517 0 865 2248
-/+ buffers/cache: 3344 4631
Swap: 1951 0 1951

- The m option displays all data in MBs.

- The total os 7976 MB is the total amount of RAM installed on the system, that is 8GB.

- The used column shows the amount of RAM that has been used by linux, in this case

around 6.4 GB. The second line tells that 4.6 GB is free. This is the free memory in first line added with the buffers and cached amount of memory (1517 + 865 + 2248 around 4631 MB= 4.6GB).

- Linux has the habit of caching lots of things for faster performance, so that memory can

be freed and used if needed.

- The last line is the swap memory, which in this case is lying entirely free.

cat /proc/cpuinfo

- The file **/proc/cpuinfo** displays what type of processor the user system is running

including the number of CPUs present.

- Portion of the Output:

cat /proc/cpuinfo
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 45
model name : Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz
stepping : 6
microcode : 1561
cpu MHz : 600.000
cache size : 20480 KB

cat /proc/meminfo

- On Linux, user can use the command cat /proc/meminfo to **determine how much memory the computer has**.

- This command displays the information stored in the meminfo file located in the /proc

directory.

- The total amount of memory will be displayed as MemTotal

- Know that the /proc file system does not contain **real files**.

  - They are rather **virtual files** that contain dynamic information about the kernel

and the system.

- **Portion of output is**

$ cat /proc/meminfo

MemTotal: 8167848 kB

MemFree: 1409696 kB

Buffers: 961452 kB

Cached: 2347236 kB

SwapCached: 0 kB

SwapTotal: 1998844 kB

SwapFree: 1998844 kB

...

- Check the values of MemTotal, MemFree, Buffers, Cached, SwapTotal, SwapFree. They indicate same values of memory usage as the free command.

Work on log directory:

/var/log

- It is essential that user know where the log files are located, and what is contained in them. Such files are usually in **/var/log**. Logging is controlled by the associated .**conf** file.
- Some log files are distribution specific and this directory can also contain applications such as samba, apache, lighttpd, mail etc.
- Generated log files will be important for system administration and troubleshooting.
- Learn and understand the content of various log files, which will help user when there is a crisis and user have to look though the log files to identify the issue.

Few log files:

- **/var/log/messages** – Contains global system messages, including the messages that are logged during system startup. There are several things that are logged in /var/log/messages including mail, cron, daemon, kern, auth, etc.
- **/var/log/auth.log** – Contains system authorization information, including user logins and authentication machinsm that were used.
- **/var/log/boot.log** – Contains information that are logged when the system boots

- **/var/log/lastlog** – Displays the recent login information for all the users. This is not an ascii file. User should use lastlog command to view the content of this file.
- **/var/log/user.log** – Contains information about all user level logs
- **/var/log/btmp** – This file contains information about failed login attemps. Use the last command to view the btmp file. For example, “last -f /var/log/btmp | more”
- **/var/log/yum.log** – Contains information that are logged when a package is installed using yum
- **/var/log/cron** – Whenever cron daemon (or anacron) starts a cron job, it logs the information about the cron job in this file

System maintenance commands:

Shutdown

- “**Shutdown**” refers to the process of stopping and shutting down a computer or server.
- This involves cutting the power to the main components of the system using a controlled process.
- Applications are closed, active processes and protocols are saved to the hard drive, device drivers are removed, and user settings are saved in the process.
- Linux operating systems can easily be stopped, shut down, and restarted using the shutdown command and its various options.
- **Standard command for shutting down Linux**
  - **shutdown –h**
- Linux will shut down in under a minute. The “-h” option explicitly stands for the shutting down or powering off of a system.
  - **shutdown**
- User can usually produce the same results by just entering the shutdown command on its own.
- **Standard command for restarting Linux**

- **shutdown -r**

- Linux will be restarted in under a minute.
- The “-r” option stands for reboot or restart.

Rebooting

- Booting is **starting a computer's operating system**, so rebooting is to start it for a second or third time.
- Rebooting is usually necessary after a computer crashes, meaning it stops working because of a malfunction.
- **sudo reboot**
- **sudo systemctl reboot**
- **sudo shutdown -r**

halt

- This command in Linux is **used to instruct the hardware to stop all the CPU functions**.
- Basically, it reboots or stops the system.
- **halt [OPTION]**
- **Options used:**

- -f, –force It does not invoke shutdown
- -w, –wtmp-only It will not call shutdown or the reboot system call but writes the shutdown record to /var/log/wtmp file.
- -p, –poweroff To behave as poweroff

init

- **init** is parent of all Linux processes with PID or process ID of 1.
- It is the first process to start when a computer boots up and runs until the system shuts down.

- init stands for initialization.
- The role of init is to create processes from script stored in the file /etc/inittab which is a configuration file which is to be used by initialization system.
- It is the last step of the kernel boot sequence.
  - init script initializes the service. So, it responsible for initializing the system.
  - Init scripts are also called rc scripts (run command scripts)
- Run Levels is the state of init where a group of processes are defined to start at the startup of OS.
- Each runlevel has a certain number of services stopped or started. Conventionally seven runlevel exist numbers from zero to six.

Run Level	Mode	Action
0	Halt	Shuts down system
1	Single-User Mode	Does not configure network interfaces, start daemons, or allow non-root logins
2	Multi-User Mode	Does not configure network interfaces or start daemons.
3	Multi-User Mode with Networking	Starts the system normally.
4	Undefined	Not used/User-definable
5	X11	As runlevel 3 + display manager(X)
6	Reboot	Reboots the system

By default most of the LINUX based system boots to runlevel 3 or runlevel 5.

- Runlevels 2 and 4 are used for user defined runlevels

- runlevel 0 and 6 are used for halting and rebooting the system.

System update & repositories

Update the Repositories

- **sudo apt-get update**

- This command refreshes local list of software, making a note of any newer revisions and updates.

- If there's a newer version of the kernel, the command will find it and mark it for download and installation.

Run the upgrade

- **sudo apt-get dist-upgrade**

- The "dist-upgrade" switch asks Ubuntu to handle any **dependencies** intelligently.

  - That is, if a particular software package is **dependent** on another software package to run, this command will make sure that the second package is upgraded before upgrading the first one.

- This method is a safe way to upgrade Ubuntu Linux kernel.

- The kernel updates accessible through this utility have been tested and verified to work with version of Ubuntu.

Packaging Manager

- Packaging manager is the software used for managing, installing, updating, upgrading etc. of the packages of a system.

- Linux based systems or Linux systems have a lot of such packaging managers in which two are: **yum** and **rpm**.

yum

- Yum and RPM are both package managers for Linux systems.

- Yum stands for Yellowdog Updater Modified. They are packaging managers for RPMbased Linux systems.

- They are a high-level front end management package managers for Linux distributions that are RPM-based.
- It can sense and resolve dependencies.
- Yum can only install the packages available in its repository.
- Yum can also scan and upgrade the packages to the latest versions. It also entirely relies on online repositories.

rpm

- RPM stands for Redhat Packaging Manager.
- It can be considered one of the oldest packaging managers that do basic functions like uninstalling, updating, archiving the packages received by the Linux systems.
- It cannot sense and resolve dependencies on its own.
- It can install multiple packages with the condition that we give the correct file name with the .rpm extension.
- RPM does not depend on online repositories for any of its services and it cannot scan or upgrade itself or its packages to the latest versions.

12 – Domain Name Service (DNS)

Domain Name Service (DNS)

- **DNS** is an Internet service that maps IP addresses and fully qualified domain names (FQDN) to one another.
  - In this way, DNS alleviates the need to remember IP addresses.
- Computers that run DNS are called **name servers.**
- Ubuntu ships with **BIND** (Berkley Internet Naming Daemon), the most common program used for maintaining a name server on Linux.
  - BIND is the most common program used for maintaining a name server on Linux.

FTP server on LINUX and transfer files to demonstrate its working.

ftp

ftp is the user interface to the Internet standard File Transfer Protocol. The program allows a user to transfer files to and from a remote network site.

Examples below are to upload and download files using ftp service.

Prerequisites:

- Install **Ftp** (FTP-Client) and **vsftpd** (FTP-Server) using Software Manager.
- Change **/etc/vsftpd.conf** file to enable write permission (other settings can be done according to requirement.
  - **nano /etc/vsftpd.conf**
- Search for **# write_enable=YES (**around 30th line in that file)
- **Uncommnent** and Save and Exit
- Restart the Service: **systemctl restart vsftpd**

Example 1: Get the connection to ftp

- **ftp**

>open

192.168.21.198

- Enter user name and password for connection, this will take user to ftp utility and then issue ftp set of command
- For more command take the help by issuing? And to quit utility type **quit** command.

Example 2: Download the file from ftp server

- **get a.txt /home/cs/test.txt**
- will download the file **a.tx**t from **ftp server** to current **user** with file name **test.txt**

Example 3: Upload the file from server

- **put /home/cs/index.html test.html**
- will upload the file **index.html** from current (local **/home/cs**) user to remote ftp server with name **test.html**

Install and configure Apache web server and create virtual hosts:

- The Apache web server is the most popular way of serving web content on the internet.
- It accounts for more than half of all active websites on the internet and is extremely powerful and flexible.
- Apache breaks its functionality and components into individual units that can be customized and configured independently. The basic unit that describes an individual site or domain is called a virtual host.
- These designations allow the administrator to use one server to host multiple domains or sites off of a single interface or IP by using a matching mechanism. This is relevant to anyone looking to host more than one site off of a single server.
- Each domain that is configured will direct the visitor to a specific directory holding that site's information, never indicating that the same server is also responsible for other sites.
- This scheme is expandable without any software limit as long as server can handle the load.
- Install apache Server: **sudo apt-get install apache2**

Step 1: Creating the Directory Structure

- Make a directory structure that will hold the site data that we will be serving to visitors.
- Document **root** (the top-level directory that Apache looks at to find content to serve) will be set to individual directories under the **/var/www** directory.
- Create a directory here for virtual host.
- Within *this* directory, create a public_html folder that will hold our actual files. This gives some flexibility in hosting.
- **sudo mkdir -p /var/www/example.com/public_html**

Step 2: Granting Permissions

- Directory structure for host files owned by our root user. If user want regular user to be able to modify files in web directories, can change the ownership by doing this:
- **sudo chown -R $USER:$USER /var/www/example.com/public_html**
- By doing this, regular user now owns the **public_html** subdirectories where we will be storing our content.
- Also modify permissions a little bit to ensure that read access is permitted to the general web directory and all of the files and folders it contains so that pages can be served correctly:
- **sudo chmod -R 755 /var/www**

Step 3: Creating Demo Pages for Each Virtual Host

- Once directory structure in place then create some content to serve.
- Design a very simple Web page.
- nano /var/www/example.com/public_html/index.html
- In this file, create a simple HTML document that indicates the site it is connected to. The file looks like this:

```
<html>
<head>
<title>Welcome to Apache Server Experiment</title>
</head>
<body>
<h1>Success! The virtual host is working Fine!</h1>
</body>
```

</html>

- Save and close the file when finished.

Step 4: Creating New Virtual Host Files

- Virtual host files are the files that specify the actual configuration of our virtual hosts and dictate how the Apache web server will respond to various domain requests.
- Apache comes with a default virtual host file called 000-default.conf that can used as a jumping off point.
- Now copy it over to create a virtual host file for domains.
- The default Ubuntu configuration requires that each virtual host file end in .conf.

Creating the First Virtual Host File

- Start by copying the file for the domain:

sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/example.com.conf

- The file will look something like this (Comments have been removed here to make the file more approachable): After removing the comments the file contents look like (First Column).

sudo nano /etc/apache2/sites-available/example.com.conf

Before	**After**
<VirtualHost *:80> ServerAdmin webmaster@localhost DocumentRoot /var/www/html ErrorLog ${APACHE_LOG_DIR}/error.log CustomLog ${APACHE_LOG_DIR}/access.log combined </VirtualHost>	<VirtualHost *:80> ServerAdmin **admin@example.com** **ServerName example.com** **ServerAlias www.example.com** DocumentRoot**/var/www/example.com/public_html** ErrorLog ${APACHE_LOG_DIR}/error.log CustomLog ${APACHE_LOG_DIR}/access.log combined </VirtualHost>

- Add few contents to the above file. And also make few changes to the file such that it finally looks like (Second Column):

- Save and close the file.

Step 5: Enabling the New Virtual Host Files

- Now virtual host files have been created, Enable them. Apache includes some tools that allow us to do this.
- Ue the **a2ensite** tool to enable each of our sites like this:

sudo a2ensite example.com.conf

- Next, disable the default site defined in **000-default.conf**:

sudo a2dissite 000-default.conf

- When above task finished, Restart Apache to make these changes take effect:

sudo systemctl restart apache2

Step 6: Setting Up Local Hosts File

 If admin have not been using actual domain names to test this procedure and **have been using** some **example domains** instead, admin can at least test the functionality of this process by temporarily modifying the hosts file on local computer.

 edit local file with administrative privileges by typing:

sudo nano /etc/hosts

 For **example**, for the domains machine with ip address **192.168.21.7** is used then, add the following lines to the bottom of hosts file:

127.0.0.1 localhost

127.0.1.1 guest-desktop

192.168.21.7 example.com

 This will direct any requests for example.com on user computer and send them to server at **192.168.21.7** This is what is required if we are not actually the owners of these domains in order to test our virtual hosts.

 Save and close the file.

Testing working of Web Server

 Go to the other machines /etc/hosts and do the above settings (**step 6**) like

192.168.21.7

 Try **http://example.com**, this will display the web page of virtual host.