Analyze and store logs

Linux #redhat #logs

Log Monitoring

Another and most important way of system administration is log monitor.

- Think of it as if you have your personal physician or a doctor, every time you go to him, he or she has a chart on you with the record of your medical history, which tells the doctor what are the problems you had in the past, whether you had any surgery, whether you're allergic to any medicine. So that is the way doctors keep logs on your physical health.
- Just like that systems also have to keep logs and generate logs and record everything that goes on with the system.
- In a Linux system, the primary log directory is `/var/log`
  - All the logs that are generated in Linux machines are mainly in `/var/log` directory unless specified or changed in the configuration file of an application to change the log location.

Log Directory = `/var/log`

- `boot` - Log when your system boots up or reboots.
  - Records everything that goes on to your system
    - How it cleans up the memory
    - What are the process starting up
    - If it is having any issues, it will log everything in there.
- `chronyd` = `NTP` - (The newer version of NTP), it has its own log.
- `cron`
- `maillog`
- `secure`
- `messages` - One of the important logs to monitor system activities.
  - Everything that is hardware wise, application wise, process wise, or anything that has todo with the system it goes into `/var/log/messages`
- `httpd` - An Apache application log.

Boot logs

Example using `ll` command:

```
[user@localhost log]$ ll
```

- Now the current directory location is `/var/log`
- Remember the `ll` command has the same effect as the `ls -ltr` except for the sorting order, instead of ordering from newest to older, it sorts in alphabetical order of the name of the file.

Output:

```
total 4608
drwxr-xr-x. 2 root   root      4096 Jun  3 13:27 anaconda
```

```
drwx------. 2 root   root        23 Jun  3 13:37 audit
-rw-------. 1 root   root      2099 Jun 14 12:21 boot.log
-rw-------. 1 root   root     16674 Jun  6 11:48 boot.log-20240606
-rw-------. 1 root   root     16676 Jun  9 15:55 boot.log-20240609
-rw-------. 1 root   root     16619 Jun 10 11:22 boot.log-20240610
-rw-------. 1 root   root     32992 Jun 11 11:10 boot.log-20240611
-rw-------. 1 root   root     16675 Jun 12 00:14 boot.log-20240612
-rw-------. 1 root   root     65723 Jun 13 10:45 boot.log-20240613
-rw-------. 1 root   root     16675 Jun 14 12:21 boot.log-20240614
-rw-rw----. 1 root   utmp      5760 Jun 14 13:42 btmp
drwxr-x---. 2 chrony chrony       6 Jan 23 08:33 chrony
-rw-------. 1 root   root     17215 Jun 14 13:24 cron
-rw-------. 1 root   root     13248 Jun  6 23:01 cron-20240609
drwxr-xr-x. 2 lp     sys         57 Jun  3 13:13 cups
-rw-r--r--. 1 root   root     30518 Jun 14 13:17 dnf.librepo.log
-rw-r--r--. 1 root   root     66505 Jun 14 13:17 dnf.log
-rw-r--r--. 1 root   root      2378 Jun 14 13:17 dnf.rpm.log
-rw-r-----. 1 root   root         0 Jun  3 13:37 firewalld
drwx--x--x. 2 root   gdm          6 Jan 18 09:08 gdm
-rw-r--r--. 1 root   root      1533 Jun 14 13:17 hawkey.log
-rw-r--r--. 1 root   root       900 Jun  6 21:04 hawkey.log-20240609
drwx------. 2 root   root         6 Feb 15 12:01 insights-client
-rw-------. 1 root   root     11826 Jun 14 12:22 kdump.log
-rw-rw-r--. 1 root   utmp    293168 Jun 14 13:43 lastlog
-rw-------. 1 root   root         0 Jun  9 15:55 maillog
-rw-------. 1 root   root         0 Jun  3 13:11 maillog-20240609
-rw-------. 1 root   root   2065905 Jun 14 13:43 messages
-rw-------. 1 root   root   2116408 Jun  6 22:33 messages-20240609
drwx------. 2 root   root         6 Jun  3 13:11 private
drwxr-xr-x. 2 root   root         6 Mar 26 09:19 qemu-ga
lrwxrwxrwx. 1 root   root        39 Jun  3 13:11 README -> ../../usr/share/doc/systemd/README.logs
drwxr-xr-x. 2 root   root       119 Jun  9 15:55 rhsm
drwx------. 3 root   root        17 Jun  3 13:11 samba
-rw-------. 1 root   root     40897 Jun 14 13:43 secure
-rw-------. 1 root   root     21941 Jun  6 13:15 secure-20240609
drwx------. 2 root   root         6 Aug 10  2021 speech-dispatcher
-rw-------. 1 root   root         0 Jun  9 15:55 spooler
-rw-------. 1 root   root         0 Jun  3 13:11 spooler-20240609
drwxr-x---. 2 sssd   sssd        55 Jun  9 15:55 sssd
-rw-------. 1 root   root         0 Jun  3 13:11 tallylog
drwxr-xr-x. 2 root   root        23 Jun  3 19:58 tuned
-rw-rw-r--. 1 root   utmp     32640 Jun 14 12:27 wtmp
```

- This is how the `/var/log` directory looks like.
- Audit is one of the files or directory that has all the audit information.
- Then we have boot.log.
  - If we try to read this file as a common user it will throw: `more: cannot open boot.log: Permission denied`, this when using `more` command.
  - This is because the file is owned by root and the group who owns that file is also root. Also we can clearly see that it has no permissions at the group and others levels.

Example using `more` command:

```
[root@localhost log]# more boot.log
```

- Note we are located at the `/var/log` directory.
- Note we are root user so we will be able to read this file.

Output:

```
[  OK  ] Finished Rotate log files.
[  OK  ] Started Authorization Manager.
         Starting Modem Manager...
         Starting firewalld - dynamic firewall daemon...
[  OK  ] Started Accounts Service.
...
```

- Coming up with OK messages while system is booting up.
- Every time there is an issue with the system booting up, it will come up with a message saying false, error or alert.
- Starting all the services and processes one by one.
- When you reboot your system, the `/var/log/boot.log` file gets overwritten.

chronyd logs

- Any type of changes that we make on the Chrony service, it actually generates the log and it actually logs that information into their logs.

cron logs

- Whenever you schedule a job or a process through a cron tab entry, it generates some kind of activity, and that activity or that record is logged into the `/var/log/cron` file.

Example using `more` command:

```
[root@localhost log]# more cron
```

- Note we are located at the `/var/log` directory.
- Note we are root user so we will be able to read this file.

Output:

```
...
Jun 10 15:01:01 localhost run-parts[4760]: (/etc/cron.hourly) starting 0anacron
Jun 10 15:01:01 localhost run-parts[4766]: (/etc/cron.hourly) finished 0anacron
Jun 10 15:01:01 localhost CROND[4756]: (root) CMDEND (run-parts /etc/cron.hourly)
Jun 10 16:01:01 localhost CROND[4919]: (root) CMD (run-parts /etc/cron.hourly)
Jun 10 16:01:01 localhost run-parts[4922]: (/etc/cron.hourly) starting 0anacron
```

- Whenever you open up a log file:
  - The first column is the month
  - Second is the date
  - Third is time
  - Fourth is the name of your host name
  - Fifth is the daemon, which is CROND and process ID associated with that daemon.
  - Sixth is the user who's running that
  - Seventh is the command or the entry that has been associated with that cron.

Example using `dmesg` command:

```
[root@localhost log]# dmesg
```

- When you run the `dmesg` command it gives you the information about the hardware.
- Apparently RHEL 9 does not contain the a file named "dmesg" in the `/var/log` directory. In RHEL 8 an alternative to the `dmesg` command used to be `cat dmesg` while in the log directory.

Output:

```
...
[   30.602238] NET: Registered PF_QIPCRTR protocol family
[   32.507505] Warning: Unmaintained driver is detected: ip_set
[   33.409177] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
[   33.945963] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[   51.232882] block dm-0: the capability attribute has been deprecated.
[   60.706257] rfkill: input handler disabled
[  287.981112] rfkill: input handler enabled
[  297.794382] rfkill: input handler disabled
```

Maillog logs

- Has all the information about your send mail daemon
- Every time you send an email out, or every time an email comes in, all that activity is being recorded in this log.
- In my case this file is empty as I have not settled up any email in this machine.
- You can also use `more maillog` or `cat maillog` while in the `/var/log` directory to check the contents of `maillog`.
- Whenever you are troubleshooting issues with send mail service, this is one of the logs you would have to look into to see why your server is not able to send emails.

Secure logs

- Records all your logging in, logging out activity.

Example using `more` command:

```
[root@localhost log]# more secure
```

Output:

```
...
Jun  9 18:14:39 localhost useradd[5459]: new user: name=spiderman, UID=1001, GID=1001,
home=/home/spiderman, shell=/b
in/bash, from=/dev/pts/0
Jun  9 18:15:54 localhost usermod[5483]: add 'spiderman' to group 'superheros'
Jun  9 18:15:54 localhost usermod[5483]: add 'spiderman' to shadow group 'superheros'
Jun  9 19:12:41 localhost useradd[7419]: new user: name=ironman, UID=1002, GID=1002,
home=/home/ironman, shell=/bin/b
ash, from=/dev/pts/0
...
Jun 14 13:43:00 localhost su[3464]: pam_unix(su-l:session): session opened for user root(uid=0) by
```

```
mmarin(uid=1000)
Jun 14 13:43:51 localhost su[3464]: pam_unix(su-l:session): session closed for user root
Jun 14 13:51:50 localhost su[3514]: pam_unix(su-l:session): session opened for user root(uid=0) by
mmarin(uid=1000)
...
```

- When you do `more` on secure you'll see all the users that have been logged in, if they have failed logging in, from which machine they're logging in.

Example using `tail` command:

```
[root@localhost log]# tail -f secure
```

- `tail` - outputs the last part of the file
- `-f` is the option that will keep on sniffing the log and every time a new record is updated to the log, that `tail -f` will get the newest log at the bottom.
- Used to sniff the tail of the `/var/log/secure` log file.

Output:

```
Jun 14 13:43:00 localhost su[3464]: pam_unix(su-l:session): session opened for user root(uid=0) by
mmarin(uid=1000)
Jun 14 13:43:51 localhost su[3464]: pam_unix(su-l:session): session closed for user root
Jun 14 13:51:50 localhost su[3514]: pam_unix(su-l:session): session opened for user root(uid=0) by
mmarin(uid=1000)
```

- You will not get your prompt back until you do Shift + C to stop the process.
- Meaning it will keep sniffing the file.
- If a user attempts to log in to your machine it will look lik this:

Output:

```
Jun 14 14:33:03 localhost sshd[3652]: Accepted password for mmarin from 192.168.1.58 port 54926 ssh2
Jun 14 14:33:03 localhost sshd[3652]: pam_unix(sshd:session): session opened for user mmarin(uid=1000)
by mmarin(uid=0)
```

- In this example the user "mmarin" logged in from 192.168.1.58 machine.
- An SSH session is opened for mmarin.

Messages logs

- Every time there's an issue with your machine, the first thing an administrator does, they log into your machine and they will trace the logs in messages file.

Example using `more` command:

```
[root@localhost log]# more messages
```

Output:

```
...
Jun  9 15:55:15 localhost kernel:  Device   empty
Jun  9 15:55:15 localhost kernel: Movable zone start for each node
Jun  9 15:55:15 localhost kernel: Early memory node ranges
Jun  9 15:55:15 localhost kernel:  node   0: [mem 0x0000000000001000-0x000000000009efff]
Jun  9 15:55:15 localhost kernel:  node   0: [mem 0x0000000000100000-0x00000000dffeffff]
```

- This file has all the hardware information, all the software information, all the application information, all the processes information, everything is being logged into this log.

Example using `cat` command:

```
[root@localhost log]# cat messages | wc -l
```

- If you wanted to see how many lines this log file has you can run this command.
- The `wc` command is counting the lines, which is specified by the `-l` option, of the `/var/log/messages` file.

Output:

```
20753
```

Example using `grep` command:

```
[root@localhost log]# grep -i error
```

- If you want to see if there are any error messages in this file you can run this command.
- The `grep` command is being used to look for the word "error" in the `/var/log/messages` file.
- The `-i` option of the `grep` command specifies to ignore uppercase, lowercase from messages file.

Output:

```
...
Jun 14 12:26:37 localhost gnome-shell[2103]: libinput error: event3  - ImExPS/2 Generic Explorer
Mouse: client bug: event processing lagging behind by 38ms, your system is too slow
Jun 14 12:26:40 localhost gnome-shell[2103]: libinput error: event2  - AT Translated Set 2 keyboard:
client bug: event processing lagging behind by 166ms, your system is too slow
```

- You'll see it actually gripped every line inside of this log file that has a message called error.
- This way you could go and see what's going on with your system, what happened, why it failed, what time it failed.

Preserve journals

from How to configure your system to preserve system logs after a reboot

Keep system journals after a reboot

The process involves creating a storage location (probably in `/var/log`) and then editing the journald configuration to direct messages to that location.

Create a storage directory

First, create a journal directory under the `/var/log` directory:

```
[server]$ sudo mkdir /var/log/journal
```

Edit the journald.conf file

Edit the file `/etc/systemd/journald.conf` and set the **Storage** parameter to **persistent** (it is set to **auto** by default):

```
[server]$ sudo vim /etc/systemd/journald.conf
[Journal]
Storage=persistent
[...]
```

Restart systemd-journald

Next, restart the **systemd-journald** service:

```
[server]$ systemctl restart systemd-journald
```

Reboot the server

Finally, reboot the server to confirm the persistence of entries by listing the `/var/log/journal` content. You should have output that looks similar to this:

```
[server]$ ls /var/log/journal
75ab164a278e48be9cf80d80716a8cd9
```

Maintaining accurate time

- Maintaining accurate system time is critical for log file analysis across multiple systems
  - The reason we maintain the accurate time is what if you're running for example a web server and that web server, behind it, you have multiple servers. That providing the service or the actual horsepower to your website. So something goes wrong, your website crashes, and now you have to review the logs one system to another. And if your systems do not have the accurate time, then you are gonna have very hard time finding out where and at what time the problem occurred.
- Also having accurate time on a system is requirement for sensitive applications such as database in production environment
- The network time protocol (NTP) is a standard way for machines to provide and synchronize the time to the NTP server

- An NTP server is a dedicated machine/computer which responds to clients for time synchronization
- `chronyd` is a NTP service used for time synchronization in the newer Linux versions.
  - Older versions had NTPD service.
- **Command to show system time/date**
  - `date`
    - Outputs: `Fri Jun 14 03:44:47 PM CDT 2024`
    - You can use the `date` command to change your time or date.
- **Command for time/date and NTP setting**
  - `timedatectl`
    - You can see your time and date
    - You can set your time and date
    - You can check your NTP setting.

Example using `timedatectl` command:

```
[user@localhost ~]$ timedatectl
```

Output:

```
               Local time: Fri 2024-06-14 15:46:57 CDT
           Universal time: Fri 2024-06-14 20:46:57 UTC
                 RTC time: Fri 2024-06-14 20:43:53
                Time zone: America/Ojinaga (CDT, -0500)
System clock synchronized: no
              NTP service: active
          RTC in local TZ: no
```

- It'll give you local universal RTC time,
- The time zone
- The system clock synchronization, this is what the NTP setting is, and it's telling no, meaning it is not synchronized with the NTP service, which is active.
- **To get help**
  - `timedatectl --help`
  - You can also run `man timedatectl`
- **To view the list of time zones**
  - `timedatectl list-timezones`

Example using `timedatectl` command:

```
[user@localhost ~]$ timedatectl list-timezones
```

- Command to see the list of time zone available to you.

Output:

```
...
America/Anguilla
America/Antigua
America/Araguaina
```

```
America/Argentina/Buenos_Aires
America/Argentina/Catamarca
```

- You will get all the list of every country within their city that you could set the time zone to.\
- This list is provided to you the first time when you are doing the Linux installation. So you could change it now if you want to.
- **To set a time zone**
    - `timedatectl set-timezone America/Chihuahua`
    - Replace the "America/Chihuahua" time zone example to your desired time zone
- **To set time or to set time and date**
    - `timedatectl set-time HH:MM:SS`
    - `timedate set-time '2024-06-14 20:15:50'`

Example using `timedatectl` command:

```
[user@localhost ~]$ timedatectl set-time 06:48:00
```

- Attempting to change the system time with the `timedatectl` command and `set-time` function.

Output:

```
Failed to set time: Automatic time synchronization is enabled
```

- When NTP is active for system clock synchronization this command won't be effective.
- **To enable NTP synchronization**
    - `timedatectl set-ntp true`
        - Note you cannot just sync to an NTP server if you do not have any NTP server running.

chronyd

- chronyd is a daemon that is the latest version that is used nowadays to replace an NTP daemon.
    - Why did they do it? Because it probably have most enhanced features in it.
- The purpose of chronyd is time synchronization
- The package name that you'll need to install for this program is called `chronyd`
- The configuration file is `/etc/chronyd.conf`
    - This is where you go in and specify the NTP server.
    - Now again, NTP is used to synchronize your system clock with an external clock or any other clock within your organization, that serves as an NTP server.
- Log file = `/var/log/chronyd`
    - All the activity is logged into this file.
- Service = `systemctl start/restart chronyd`
    - When you configure the `chronyd.conf` file then you have to start or restart a chronyd service.
- Program command = `chronyc.`
    - It will tell you which NTP clock you are synced with, what's the status of it.

Check if we have the package installed
Example using `rpm` command:

```
[root@localhost log]# rpm -qa | grep chrony
```

- Looking for the term "chrony" using the `grep` command and the `rpm` command to query all the packages we have installed.

Output:

```
chrony-4.5-1.el9.x86_64
```

- We just confirmed that we have the chrony package
- If you do not have the chrony package installed then run: `yum install chrony`

Checking our configuration file
Example using `vi` command:

```
[root@localhost log]# vi /etc/chrony.conf
```

File editor:

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (https://www.pool.ntp.org/join.html).
pool 2.rhel.pool.ntp.org iburst

# Use NTP servers from DHCP.
sourcedir /run/chrony-dhcp
...
```

- In this example machine there is no server defined with the `server` keyword.

Add a new server to the configuration file:
File editor:

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (https://www.pool.ntp.org/join.html).

server 8.8.8.8
pool 2.rhel.pool.ntp.org iburst
...
```

- Add a new line after the first two lines of comments.
- The Google server is the 8.8.8.8 server. (This is not an NTP server by google, this is just a DNS server)
  - Only works for practice and example purposes.

Check if the service is running
Example using `systemctl` command:

```
[root@localhost log]# systemctl status chronyd
```

Output:

```
● chronyd.service - NTP client/server
     Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; preset: enabled)
     Active: active (running) since Sat 2024-06-15 13:51:18 CST; 19min ago
       Docs: man:chronyd(8)
             man:chrony.conf(5)
    Process: 817 ExecStart=/usr/sbin/chronyd $OPTIONS (code=exited, status=0/SUCCESS)
   Main PID: 843 (chronyd)
      Tasks: 1 (limit: 65779)
     Memory: 4.0M
        CPU: 471ms
     CGroup: /system.slice/chronyd.service
             └─843 /usr/sbin/chronyd -F 2

Jun 15 13:51:18 localhost chronyd[843]: Frequency 27.072 +/- 0.169 ppm read from /var/lib/chrony/drift
...
```

- In this case the status of the chronyd service is active and running.
- It is a good practice to check the status of the `ntpd` as well by doing `systemctl status ntpd` and if it is running it is recommended to stop it because you do not want to tun two services that perform or that serves the same purpose.
  - If active, run `systemctl stop ntpd`, and verify the new status.
  - You will probably want to disable it to, then run `systemctl disable ntpd` so it does not start at boot time.
  - In my build of RHEL 9 the `ntpd` package is not included.
- If the chronyd service is active already and you just changed something in its configuration file you will have to stop the service and start it again, in order to apply its configuration changes. Or you can use the command `systemctl restart chronyd` to restart the service in the process reloading its configuration file.
  - Every time you modify a configuration file, you have to restart a service.

Start chronyd
Example using `systemctl` command:

```
[root@localhost log]# systemctl start chronyd
```

- Use this line to start the service in case the service appears inactive when checking the status.

Enable chronyd at boot time
Example using `systemctl` command:

```
[root@localhost log]# systemctl enable chronyd
```

chronyc

The program that comes whit this package it's called "chronyc" which it is belived that stands for "command".
Example using `chronyc` command:

```
[root@localhost log]# chronyc
```

- When you type this command and hit enter it will bring you into an interactive program mode of it's own chronyc.

Output:

```
chrony version 4.5
Copyright (C) 1997-2003, 2007, 2009-2023 Richard P. Curnow and others
chrony comes with ABSOLUTELY NO WARRANTY.  This is free software, and
you are welcome to redistribute it under certain conditions.  See the
GNU General Public License version 2 for details.

chronyc>
```

Example using `help` command:

```
chronyc> help
```

Output:

```
...
timeout <milliseconds>      Set initial response timeout
retries <retries>           Set maximum number of retries
keygen [<id> [<type> [<bits>]]]
                            Generate key for key file
exit|quit                   Leave the program
help                        Generate this help

chronyc>
```

- When you enter `help` you'll see there's so many commands that you could run within this program.

Example using `sources` command:

```
chronyc> sources
```

- This command will tell us which NTP server we are connecting to.

Output:

```
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^? dns.google                    0   6     0     -     +0ns[   +0ns] +/-    0ns
^- nu.binary.net                 2   6    17     0  +4017us[+4017us] +/-   62ms
^+ 66.85.78.80                   2   6    17     0  +2111us[+2111us] +/-   59ms
^- t1.time.bf1.yahoo.com         2   6    17     1  -4494us[-4494us] +/-   34ms
^* ns1.your-site.com             2   6     7     2   -859us[-2216us] +/-   43ms
```

- If you remember for the example above we picked 8.8.8.8, and this is the google server that we are syncing our time to. And of course this is not actually an NTP server, it is just picked for training purposes. Do not use a DNS server for NTP purposes.

Example using `quit` command:

```
chronyc>
```

- To exit out of `chronyc` just type `quit` in the command line.