Linux

mthree

Introduction to Linux

Module 6: Understanding the health of your server

ALUMNI
BY MTHREE CONSULTING

Introduction to Linux

In this module, we go through several tools you can use in Linux to understand the health of your server.

Healthy infrastructure is vital to your application running properly, and being able to diagnose issues with your server quickly is key to incident resolution.

Seeing What Is Running On Your Server

mthree

While you may have monitoring running on your host, the ability to log on and see exactly what is running is the best way to understand the environment your application is running in. We can do this using the **ps** command.

ps	Shows processes connected to your current session; note that the ps process itself is also listed
ps -ef	This will list all processes owned by anyone on the server: a useful command that you may wish to pipe into a grep to search for something specific. The –f provides you with more columns of information
ps –f $$	Show the process info for our session
ps –ef f	All processes in tree view

What does a ps output look like?

[ec2-user@ip-172-31-40-21 ~]$ ps -ef | grep -v root

UID	PID	PPID	C	STIME	TTY	TIME	CMD
rpc	2324	1	0	Apr23	?	00:00:00	rpcbind
rpcuser	2345	1	0	Apr23	?	00:00:00	rpc.statd
dbus	2379	1	0	Apr23	?	00:00:00	dbus-daemon --system
ntp	2595	1	0	Apr23	?	00:00:24	ntpd -u ntp:ntp -p /var/run/ntpd.pid -g
smmsp	2625	1	0	Apr23	?	00:00:00	sendmail: Queue runner@01:00:00 for /var/spool/clientmqueue
ec2-user	29031	29029	0	15:45	?	00:00:00	sshd: ec2-user@pts/0
ec2-user	29032	29031	0	15:45	pts/0	00:00:00	-bash
ec2-user	29056	29032	0	15:45	pts/0	00:00:00	ps -ef

Process State Codes

mthree

Here are the different values that the s, stat, and state output specifiers (header "STAT" or "S") will display to describe the state of a process.

Add the –s option to your ps command to pull these up.

D uninterruptible sleep (usually IO)

R running or runnable (on run queue)

S interruptible sleep (waiting for an event to complete)

T stopped by job control signal

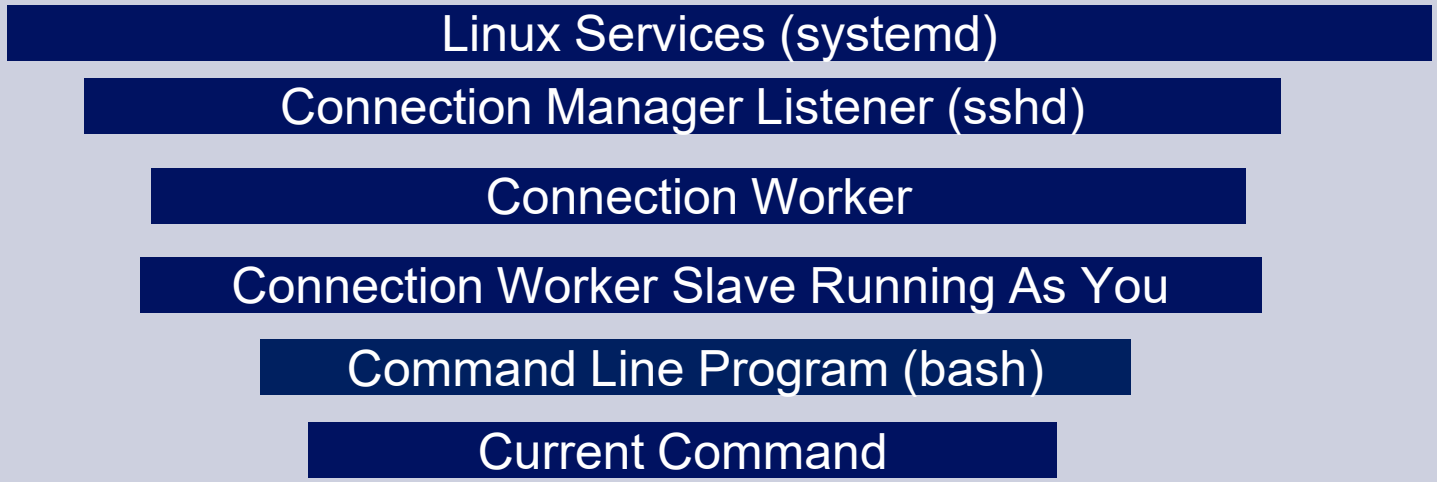
t stopped by debugger during the tracing

Process Tree

pstree -shapu $$

Show children and parent process tree for the specified process. $$ is the process id of the current program (bash or your script).

```
[ec2-user@ip-172-31-83-60 ~]$ pstree -shapu $$
init,1
  └─sshd,2583
      └─sshd,2727
          └─sshd,2729,ec2-user
              └─bash,2730
                  ├─man,2811 cat
                  │   ├─less,2822 -s
                  │   ├─man,2817 cat
                  │   ├─man,2818 cat
                  │   ├─man,2819 cat
                  │   ├─man,2820 cat
                  │   └─man,2821 cat
                  └─pstree,23241 -shapu 2730
```



How Pipes Work


```
[ec2-user@ip-172-31-83-60 ~]$ ps -ef | egrep "$$|cmd"
ec2-user  2730  2729  0 20:06 pts/0    00:00:00 -bash
ec2-user  2811  2730  0 20:16 pts/0    00:00:00 man cat
ec2-user 23253  2730  0 21:03 pts/0    00:00:00 ps -ef
ec2-user 23254  2730  0 21:03 pts/0    00:00:00 grep -E --color=auto 2730|cmd

[ec2-user@ip-172-31-83-60 ~]$ ps -eo user,pid,ppid,stat,cmd|egrep "(CMD|$$)"
USER       PID  PPID STAT CMD
ec2-user  2730  2729 Ss   -bash
ec2-user  2811  2730 T    man cat
ec2-user 23255  2730 R+   ps -eo user,pid,ppid,stat,cmd
ec2-user 23256  2730 S+   grep -E --color=auto (CMD|2730)
[ec2-user@ip-172-31-83-60 ~]$
```

Programmatic ps

We can use some of our earlier commands to demo more on what we can do with ps

Command	Comments
ps -eo pid,stat	Shows everyone's processes, but only outputs the process id and status columns
ps -eo pid,stat | grep S	Shows processes that are sleeping (as we have filtered by just the S here)
ps -eo pid,stat | grep S | head	Here we are using head to just show us the first 10
ps -eo pid,stat | grep S | head -2	Here we are showing the first two

Kill

It happens sometimes that a process misbehaves and you cannot stop it gracefully. This is where the kill command comes in.

If an application is having an issue, it is best to check in with the developers of the application before running the kill command as they may want to run some diagnostics before the process is stopped. For example, they may want to be sure the process creates a core file when it crashes.

Command	Comments
kill pid	From your ps you will be able to find your process id: this is used as an argument to the kill command
kill $$	SIGTERM signal is handled by BASH and stops the current foreground process, but not BASH itself
kill –l	This will list all options that are available to you
kill -9 pid pid2 pid3	The -9 option will force kill any process running if kill on its own does not work: this is always a last resort. This example shows that you can kill multiple processes with one command.

Top

Another great command when looking at the health of the server is **top.** Top provides a dynamic view of the processes running on the server and will continually update your screen.

top

Give me the view of all the processes on the current server; to close the view you need to type q

top –u username

```
top - 18:49:12 up 7 days,  2:26,  1 user,  load average: 0.00, 0.00, 0.00
Tasks:  75 total,   1 running,  50 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1009148k total,   398620k used,   610528k free,    86260k buffers
Swap:        0k total,        0k used,        0k free,   187088k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
29031 ec2-user  20   0  117m 4160 3076 S  0.0  0.4   0:00.03 sshd
29032 ec2-user  20   0  112m 3444 2992 S  0.0  0.3   0:00.01 bash
29415 ec2-user  20   0 15356 2152 1872 R  0.0  0.2   0:00.00 top
```

Top continued

There are some other handy commands you can run while top is active that you may find useful

Command	Comments
Press z while top is running	This will highlight running process in top – can help with diagnosis
Press c while top is running	This will give you the absolute path of each of the processes running
Shift P while top is running	This will sort the results by CPU utilization – extremely useful when diagnosing performance issues on a server
Press k while top is running	This will allow you to kill a process id – it will prompt you for the pid once you have pressed k

Ping

As well as troubleshooting server issues, at some point you will need to troubleshoot network issues.

Ping is a useful tool to see if there are any network connectivity issues between your server and a destination. It works by sending an echo request across the network and waits for response. It also shows delays and any packet loss (which could be causing issues)

```
[ec2-user@ip-172-31-40-21 ~]$ ping google.com
PING google.com (172.217.12.238) 56(84) bytes of data.
64 bytes from iad30s15-in-f14.1e100.net (172.217.12.238): icmp_seq=1 ttl=51 time=1.10 ms
64 bytes from iad30s15-in-f14.1e100.net (172.217.12.238): icmp_seq=2 ttl=51 time=1.12 ms
64 bytes from iad30s15-in-f14.1e100.net (172.217.12.238): icmp_seq=3 ttl=51 time=1.14 ms
64 bytes from iad30s15-in-f14.1e100.net (172.217.12.238): icmp_seq=4 ttl=51 time=1.15 ms
64 bytes from iad30s15-in-f14.1e100.net (172.217.12.238): icmp_seq=5 ttl=51 time=1.11 ms
--- google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 1.100/1.129/1.159/0.036 ms
```

traceroute

The command **traceroute** will allow you to follow the path your traffic goes between your server and its destination. This can help you isolate potentially network issues on the route .

```
[ec2-user@ip-172-31-40-21 ~]$ traceroute google.com
traceroute to google.com (172.217.15.78), 30 hops max, 60 byte packets
 1  216.182.226.34 (216.182.226.34)  58.324 ms 216.182.231.114 (216.182.231.114)  21.645 ms 216.182.224.118 (216.182.224.118)  20.275 ms
 2  100.66.8.40 (100.66.8.40)  21.847 ms 100.66.13.58 (100.66.13.58)  13.896 ms 100.66.9.6 (100.66.9.6)  13.888 ms
 3  100.66.11.202 (100.66.11.202)  21.699 ms 100.66.11.144 (100.66.11.144)  11.849 ms 100.66.11.86 (100.66.11.86)  56.778 ms
 4  100.66.42.182 (100.66.42.182)  29.865 ms 100.66.43.130 (100.66.43.130)  21.849 ms 100.66.46.226 (100.66.46.226)  19.573 ms
 5  100.66.6.67 (100.66.6.67)  16.366 ms 100.66.7.131 (100.66.7.131)  26.932 ms 100.66.5.109 (100.66.5.109)  11.778 ms
 6  100.66.5.207 (100.66.5.207)  22.679 ms 100.65.12.177 (100.65.12.177)  0.302 ms 100.65.12.97 (100.65.12.97)  0.418 ms
 7  52.93.28.167 (52.93.28.167)  15.428 ms 52.93.28.173 (52.93.28.173)  0.824 ms 52.93.28.141 (52.93.28.141)  0.912 ms
 8  52.93.28.141 (52.93.28.141)  0.986 ms 52.93.28.139 (52.93.28.139)  1.077 ms  0.917 ms
 9  99.83.65.3 (99.83.65.3)  1.799 ms 100.100.4.10 (100.100.4.10)  1.062 ms 99.82.181.25 (99.82.181.25)  1.237 ms
10  108.170.246.33 (108.170.246.33)  2.284 ms * 108.170.246.65 (108.170.246.65)  1.424 ms
11  74.125.252.39 (74.125.252.39)  2.443 ms *  1.922 ms
12  216.239.48.176 (216.239.48.176)  3.222 ms 74.125.252.39 (74.125.252.39)  2.145 ms iad23s63-in-f14.1e100.net (172.217.15.78)  1.332 ms
[ec2-user@ip-172-31-40-21 ~]$
```

netstat

While you will have help from infrastructure support teams, it is good to understand the **netstat** command and how it can be useful

A socket in this context handles one end of a network data connection – they can be in connected state or waiting (often called listening) – example sample of output is below.

Command	Comments
netstat –a	This will provide you with a view of everything on the server – you will likely want to pipe this into a less so you can search the result

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:56335                 *:*                     LISTEN
tcp        0      0 *:sunrpc                *:*                     LISTEN
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
unix  2      [ ACC ]     STREAM     LISTENING     11165  /var/run/dbus/system_bus_socket
unix  2      [ ACC ]     STREAM     LISTENING     11694  /var/run/acpid.socket
```

ifconfig

The command **ifconfig** is a utility that allows you to configure, assign, add, delete, control, and query your network interfaces. You will likely not be able to run any commands except to view the status as here.

Eth0 in this is my ethernet card

Lo: is the loopback address

ifconfig –a will show you all of the network interfaces

```
[ec2-user@ip-172-31-40-21 ~]$ ifconfig
eth0      Link encap:Ethernet  HWaddr 0E:87:1C:FE:52:EF
          inet addr:172.31.40.21  Bcast:172.31.47.255  Mask:255.255.240.0
          inet6 addr: fe80::c87:1cff:fefe:52ef/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:9001  Metric:1
          RX packets:135616 errors:0 dropped:0 overruns:0 frame:0
          TX packets:128910 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:26265424 (25.0 MiB)  TX bytes:34146946 (32.5 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:140 (140.0 b)  TX bytes:140 (140.0 b)
```