Linux script to periodically log enabled systemctl services, configured network IPs and routings, server established connections and iptables firewall rules

Author : admin

For those who are running some kind of server be it virtual or physical, where multiple people or many systemins have access, sometimes it could be quite a mess as someone due to miscommunication or whatever could change something on the configured Network Ethernet interfaces, or configured routing tables, or simply issue an update which might change the set of automatically set to run systemctl services due to update. Such changes on a Linux server Operating system often can remain unnoticed and could cause quite a harm. Even when the change is noticed the logical question occurs what was the previous network route on the server or what kind of network was configured on Ethernet interface **ethX** etc. Problems like the described where, pretty common in many public Private Clouds or VMWare / XEN based Hypervisors that host multiple Virtual machines, for that reason I've developed a small script which is pretty dumb on the first glimpse but mostly useful as it keeps historical records of such important information.

```
#!/bin/sh
# script to show configured services on system, configured IPs, netstat state and network routes
# Script to be used during CentOS and Redhat Enterprise Linux RPM package updates with yum

output_file=network_ip_routes_services_status;
```

```
ddate=$(date '+%Y-%m-%d_%H-%M-%S');
iptables=$(which iptables);
if [ ! -d /root/logs/ ]; then
mkdir /root/logs/;
fi

  echo "STARTED: $(date '+%Y-%m-%d_%H-%M-%S'):" | tee -a /root/logs/$output_file-$(hostname)-$ddate.log
echo -e "# systemctl list-unit-files\n" | tee -a /root/logs/$output_file-$(hostname)-$ddate.log
systemctl list-unit-files --type=service | grep enabled | tee -a /root/logs/$output_file-$(hostname)-$ddate.log
echo -e '# systemctl | grep ".service" | grep "running"\n' | tee -a /root/logs/$output_file-$(hostname)-$ddate.log
systemctl | grep ".service" | grep "running" | tee -a /root/logs/$output_file-$(hostname)-$ddate.log
echo -e "# netstat -tulpn\n" | tee -a /root/logs/$output_file-$(hostname)-$ddate.log
netstat -tulpn | tee -a /root/logs/$output_file-$(hostname)-$ddate.log
echo -e "# netstat -r\n" | tee -a /root/logs/$output_file-$(hostname)-$ddate.log
netstat -r | tee -a /root/logs/$output_file-$(hostname)-$ddate.log
echo -e "# ip a s\n" | tee -a /root/logs/$output_file-$(hostname)-$ddate.log
ip a s | tee -a /root/logs/$output_file-$(hostname)-$ddate.log
echo -e "# /sbin/route -n\n" | tee -a /root/logs/$output_file-$(hostname)-$ddate.log
/sbin/route -n | tee -a /root/logs/$output_file-$(hostname)-$ddate.log
echo -e "# $iptables -L -n\n" | tee -a /root/logs/$output_file-$(hostname)-$ddate.log
echo -e "# $iptables -t nat -L" | tee -a /root/logs/$output_file-$(hostname)-$ddate.log
$iptables -L -n | tee -a /root/logs/$output_file-$(hostname)-$ddate.log
$iptables -t nat -L | tee -a /root/logs/$output_file-$(hostname)-$ddate.log
echo "ENDED $(date '+%Y-%m-%d_%H-%M-%S'):" | tee -a /root/logs/$output_file-$(hostname)-$ddate.log
```

Script produces its logs inside **/root/logs/network_ip_routes_services_status*hostname*currentdate*.log,** put the script inside /root/ or wherever you like.

To keep an eye how network routing or ip configuration or firewall changed or there was a peak with the established connections towards daemons running on host (lets say requiring a machine upgrade), I've set the script to run as usually via cron job at the end of the predefined cron job tasks, like so:

```
# crontab -u root -e
# periodic dump and log network routing tables, netstat and systemctl list-unit-files
*/1 01 01,25 * * /root/show_running_services_netstat_ips_route1.sh 2>&1 >/dev/null
```

You can download a copy of [show_running_services_netstat_ips_route1.sh script here](show_running_services_netstat_ips_route1.sh).
The script is written without much of efficiency on mind, as you can see the with the multiple tee -a and for critical hosts it might be a good idea to rewrite it to use '>>' OPERAND instead, anyhows as most machines today are pretty powerful it doesn't really matter much.

Of course today such a script is quite archaic, as most big corporations are using much more complex monitoring software such as Zabbix, Prometheus or if some kind of Elastic Search is used Kibana etc. but for a basic needs and even for a double checking and comparing with other more advanced monitoring tools (in case if monitoring tools database gets damaged or temporary down until backupped), still I think such an oldschool simple monitoring script can be of use.

A good addition to that if you use a central logging server is to set another cron to periodically synchronize produced /root/logs/* to somewhere, here is how to do it with simple rsync (considering your host is configured to login with a user without password with ssh key authentication).

```
# HOSTNAME=$(hostname); rsync -axHv --ignore-existing -e 'ssh -p 22' /bashscripts/  -q -i
--out-format="%t %f %b" --log-file=/var/log/rsync_sync_jobs.log --info=progress2
root@BACKUP_SERVER_HOST:/$(HOSTNAME)-logs/
```

Once something strange occurs with the machine, like the machine needs to be rebuild

I would be glad to hear if some of my readers uses some useful script which I can adopt myself. Cheers :)