Matt Favazza

Server Monitoring Script

Before we decide what we need in our script, we need to figure out what we want to monitor first. I found this article which shows different tools to monitor the system

<https://www.cyberciti.biz/tips/top-linux-monitoring-tools.htmlhttps://www.cyberciti.biz/tips/top-linux-monitoring-tools.html>

This article has tools that we can use to monitor the server. I want to monitor the hardware so we can either use number 23, **nmon** or number 24 **glances**. In this case I will only use nmon. I’m choosing to only monitor the hardware because monitoring every process can create a lot of data, and by specifically choosing a few pieces of hardware well end up creating less data.

This script will be for the CentOS server and the Ubuntu Server.

Ubuntu Server

We need to install nmon first. **sudo apt-get install nmon**

**Touch HWMON.sh** will create our hardware monitoring script. Then we can edit it with any editor. I use nano, **nano HWMON.sh**. First and foremost, add the **#!/bin/bash** to make our script and create a header.

Text

Description automatically generated

Now we can add the two other commands to see who is logged in and the uptime of the machine.

Text

Description automatically generated

And a quick test, **bash HWMON.sh**

Text

Description automatically generated

Now here is where the script gets complicated. We need to run a nmon data collection, sort that data and then search for specific lines using **grep**.

Text

Description automatically generated

Here we run nmon and save the output to a file, which is sorted into a new file and the old file is then removed. This new file ‘nmonsort.csv’ is what we will process using **grep**. We need to use **sleep** to give the **sort** command time to work. Now we can use this data.

I have already ran nmon and saved the output using the same process above into a file called ‘test.csv’. Here we have a lot of information, roughly 1100 lines we need to now sort through. Using this csv document, we can find the data we want to display. But first we want to add whitespace to make this data more readable since everything is kind of messy when we **cat** the file.

Graphical user interface, text

Description automatically generated

So, when we use **grep** we will also use a **sed** command to add some space after every comma. Using the ‘test.csv’ as reference, all the way at the bottom of the file well see the data we want to extract.

Calendar

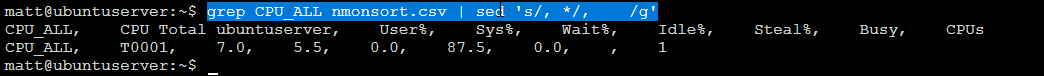
Description automatically generated

We will be grepping for the following:

CPU\_ALL, DISKBSIZE, DISKBUSY, DISKREAD, DISKWRITE, DISKXFER, JFSFILE, MEM, NET, NETPACKET, PROC, VM

Here we can search and add whitespace to the fields we want to see.

**grep CPU\_ALL nmonsort.csv | sed 's/, \*/, /g'**



We can now add this command for the rest of the data in the script.

Text

Description automatically generated

Finally we will want to kill the nmon process to save system resources. We can do that with the **kill** command, but that works off of using process ID (PID) numbers, so we need a loop with conditions to achieve this.

Text

Description automatically generated

Like I mentioned above, we need a loop to search for the PID for nmon because the process ID number can change whenever nmon is ran. So, we run **PS**, which searches processes and lists them out. Then we **grep** for nmon and print the first column using awk. Here is a breakdown of that.

Text

Description automatically generated

Now in the loop we can save the number into the variable ‘I’ and run the command kill. Again, we have to do this because kill runs off of Process ID numbers.

With the script complete, we can now automate this. First well move it into the **/usr/local/bin** directory.

Text

Description automatically generated

And make sure it has executable permissions.

Graphical user interface, text

Description automatically generated with medium confidence

Now we can add this to the cronjob lists. Type **crontab -e** to enter the editor.

Text

Description automatically generated

This will run the Hardware monitoring script at every hour at the half hour mark. 1:30, 2:30 etc. And outputs the results into HWMON.txt. I only have one > to ensure that the previous data is written over and not appended to the bottom of that text file.

CentOS Server

We do not need to rewrite the script for this server. We will simply upload it and install the required software.

Using SmarTTY to access the server, we will use the SCP feature to upload the script onto the server

Graphical user interface, application

Description automatically generated

Before we do anything we need to install nmon. **Sudo yum install nmon.** Now give it a quick test run to make sure it works properly on this system.

Text

Description automatically generated

All seems well, so now we can copy it into the **/usr/local/bin** directory

Text

Description automatically generated

Now we can make this a cronjob too. **Crontab -e** enters the editor and we can add the following to it

Text, logo

Description automatically generated

This will run every hour and output into a text file called HWMON.txt. Only one > is there to write over the previous data.

The script will output the HWMON.txt file in the home directory. Since we copied it from one directory to the /usr/local/bin we can still run it ourselves by typing **bash HWMON.sh** in the **SCRIPTS** directory on both servers where I have a copy of every script.

Resources that helped

<https://www.techrepublic.com/article/how-to-monitor-your-linux-servers-with-nmon/>

<https://www.toolbox.com/tech/operating-systems/question/nmon-daily-data-collection-cron-job-092809/>