Exercise 9 - Monitoring Processes - Script 8

Goal:

The goal of this exercise is to create a shell script that monitors a list of processes specified in a configuration file. If any of the process are not running, this script starts those processes.

Scenario:

Saturday at 3:24 AM you were called because a web application that is hosted on one of your servers was unavailable. You crawled out of bed, opened up your laptop, logged into the system and found that the new application that you deployed last week wasn't running. You immediately started it so that the site would be available again. Crisis averted.

Being the awesome sysadmin you are, you find it hard to just restart a service without determining why it went down in the first place. When you examined the log files you discovered OOM (out of memory) error messages. You send a quick email to the development team stating that you suspect the application has a memory leak and you'd be happy to help them research it further on Monday.

After meeting with the development team on Monday it becomes clear that they will not have a fix for this problem anytime soon. You enjoy sleeping through the night, so you decide to write a script that simply restarts the service when it goes down.

Shell Script Requirements:

You think about what the shell script must do and how you would like it operate. You come up with the following list.

The script:

- Is named "`watchdog.sh`".
- Enforces that it be executed without superuser (root) privileges. Exits with an exit status of 1 if not executed with superuser privileges.
- Uses a configuration file named "watchdog.conf.HOSTNAME" located in /vagrant. This allows you to create one configuration file for each host. Each line in the file contains the name of the process to monitor and the command to start the process when it fails. Here's the format:
    - PROCESS_NAME COMMAND_TO_START_PROCESS_IF_IT_FAILS
    - Examples:
        - sshd systemctl start sshd
        - rsyslogd systemctl start rsyslog

- Exits with an exit status of 1 if the configuration file cannot be found or read.
- Checks to see if at least one process is running with the exact name listed in the configuration file. If it is not, it execute the provided command to start the process. If any processes are restarted, make the exit status of the script 2.
- Creates a log file named "watchdog.log.HOSTNAME" located in /vagrant. This allows you to have one log file per server.

Start the Virtual Machines and Log into admin01

In a previous exercise you created a vagrant project called multinet. Use the VMs created in that project for this exercise.

First, start a command line session on your local machine. Next, move into the working folder you created for this course.

```
cd shellclass
```

Change into the multinet directory, start the virtual machines with "vagrant up", and then connect to the admon01 VM with "vagrant ssh admin01".

```
vagrant up
vagrant ssh admin01
```

Navigate to the `/vagrant` Directory

```
cd /vagrant
```

Write the Shell Script

At this point, you can either create the script inside the virtual machine using the `vim`, `nano`, or `emacs` text editors or you can create the file using your favorite text editor on your local operating system. (Atom from https://atom.io/ is a good choice.)

When creating your script, refer back to the shell script requirements. If you want or need more detailed steps to help you write your script, refer to the pseudocode at the end of this document. It was intentionally placed at the end of the document because I want to encourage you to write the script on your own. It's fine if you need the pseudocode. As you get more scripting practice, you'll be able to script without any additional aids.

Create the Configuration Files

Contents of `/vagrant/watchdog.conf.admin01:`
```
sshd systemctl start sshd
rsyslogd systemctl start rsyslog
```

Contents of `/vagrant/watchdog.conf.server01:`
```
sshd systemctl start sshd
rsyslogd systemctl start rsyslog
httpd systemctl start httpd
```

Contents of `/vagrant/watchdog.conf.server02:`
```
sshd systemctl start sshd
rsyslogd systemctl start rsyslog
httpd systemctl start httpd
```

Test Your Script

Once you've finished writing the script, test it by:

- Executing it without super user privileges.
- Making sure it creates a log file.
- Stopping a service and making sure the script restarts it.
- Scheduling the script to run via cron and stopping a service to make sure the script restarts it.

Remember that the first time you execute the script you'll need to make sure it has executable permissions.

```
chmod 755 watchdog.sh
```

Here is an example run of the script. (Portions typed are in bold.)

```
./watchdog.sh
Please run with sudo or as root.
echo ${?}
1
```

Execute the script with root privileges and make sure a log file is created.

```
sudo ./watchdog.sh
cat watchdog.log.admin01
Apr 11 10:22:26 Checking service: sshd
Apr 11 10:22:26 sshd running as PID(s): 2971 2969 954
Apr 11 10:22:26 Checking service: rsyslogd
Apr 11 10:22:26 rsyslogd running as PID(s): 3053
```

Let's stop the rsyslogd process and see if our script restarts it. (NOTE: the process name is "rsyslogd", but the systemctl service name does not include the trailing "d": "rsyslog".)

```
sudo systemctl stop rsyslog
sudo ./watchdog.sh
echo ${?}
2
cat watchdog.log.admin01
Apr 11 10:22:26 Checking service: sshd
Apr 11 10:22:26 sshd running as PID(s): 2971 2969 954
Apr 11 10:22:26 Checking service: rsyslogd
Apr 11 10:22:26 rsyslogd running as PID(s): 3053
Apr 11 10:25:22 Checking service: sshd
Apr 11 10:25:22 sshd running as PID(s): 2971 2969 954
Apr 11 10:25:22 Checking service: rsyslogd
Apr 11 10:25:22 Restarting rsyslogd with command: systemctl start rsyslog
```

Add the following crontab entry to root's crontab on all the servers so that the script runs every minute:

```
* * * * * /vagrant/watchdog.sh &>> /var/tmp/watchdog.cronlog
```

(HINT: run "`sudo crontab -e`" or "`su -`" followed by "`crontab -e`".)

Running this script every minute is fine for testing. However, consider running it every 5 minutes in a production. This way if it takes a service a couple of minutes to start, the script will not execute again while it's still running. Also, 5 minutes is a "reasonable" amount of time for a service check. Here's the time specification to use for every 5 minutes: "`*/5 * * * *`"

Make sure the script is executing properly via cron.

```
cat /vagrant/watchdog.log.admin01
Apr 11 10:26:26 Checking service: sshd
Apr 11 10:26:26 sshd running as PID(s): 2971 2969 954
Apr 11 10:26:26 Checking service: rsyslogd
cat /var/tmp/watchdog.cronlog
```

Make sure the script restarts a downed service properly when running via cron.

```
sudo systemctl stop rsyslog
tail -f /vagrant/watchdog.log.admin01
Apr 11 12:14:01 Checking service: sshd
Apr 11 12:14:01 sshd running as PID(s): 5219 5217 954
Apr 11 12:14:01 Checking service: rsyslogd
Apr 11 12:14:01 Restarting rsyslogd with command: systemctl start rsyslog
cat /var/tmp/watchdog.cronlog
```

Make sure the script is executing properly on the other servers.

```
ssh server01
sudo systemctl stop httpd
tail -f /vagrant/watchdog.log.server01
Apr 11 12:16:01 Checking service: sshd
Apr 11 12:16:01 sshd running as PID(s): 4394 4392 953
Apr 11 12:16:01 Checking service: rsyslogd
Apr 11 12:16:01 rsyslogd running as PID(s): 954
Apr 11 12:16:01 Checking service: httpd
Apr 11 12:16:01 Restarting httpd with command: systemctl start httpd
cat /var/tmp/watchdog.cronlog
exit
```

```
ssh server02
sudo systemctl stop httpd
tail -f /vagrant/watchdog.log.server02
Apr 11 12:18:02 Checking service: sshd
Apr 11 12:18:02 sshd running as PID(s): 4394 4392 953
Apr 11 12:18:02 Checking service: rsyslogd
Apr 11 12:18:02 rsyslogd running as PID(s): 954
Apr 11 12:18:02 Checking service: httpd
Apr 11 12:18:02 Restarting httpd with command: systemctl start httpd
cat /var/tmp/watchdog.cronlog
```

Reference Material:

Vagrantfile for multinet

Here are the contents of the `shellclass/multinet/Vagrantfile` file with all the comments removed.

```
Vagrant.configure(2) do |config|
  config.vm.box = "jasonc/centos7"

  config.vm.define "admin01" do |admin01|
    admin01.vm.hostname = "admin01"
    admin01.vm.network "private_network", ip: "10.9.8.10"
  end

  config.vm.define "server01" do |server01|
    server01.vm.hostname = "server01"
    server01.vm.network "private_network", ip: "10.9.8.11"
  end

  config.vm.define "server02" do |server02|
    server02.vm.hostname = "server02"
    server02.vm.network "private_network", ip: "10.9.8.12"
  end

end
```

[This space intentionally left blank. Instructions continue on the following page.]

Pseudocode

You can use the following pseudocode to help you with the logic and flow of your script.

```
# The location of the configuration file.

# The location of the log file.

# Include path to the pidof command

  # Restarts a given service if it is not running.

  # Requires SERVICE_NAME and START_SERVICE_COMMAND as arguments.

  # Returns 0 if SERVICE_NAME us running, 1 if SERVICE_NAME was restarted

# Make sure the script is being executed with superuser privileges.

# Make sure the config file exists.

# Expect the best, prepare for the worst.

# Read the configuration file, line by line.
```