CS312 :: Lab Week 8 :: Scheduling and Ansible

In this lab, we'll play around with process scheduling, Ansible, and a few surprises.

Supplies needed

- Personal laptop with VirtualBox installed
- Our **pfSense_Reference**, **CentOS_Reference**, and **Alpine_Reference** virtual machines, downloadable from our Canvas website.

Perform the Procedure and answer the Questions in order, as given below.

Procedure

In this Lab, we'll be doing the following:

1. Set up the Virtual Machines
2. Explore `at` and email
3. Explore `cron`
4. Scheduled Ansible deployment

Setting up the Virtual Machines

To get started, reset the three VMs specified above back to their original snapshots. If you have changed the VMs, but don't have any snapshots, go ahead and delete the VMs and reimport them: that will bring you back to a known good state. We recommend you take a snapshot now of those VMs, so that you can reset back to this original state at any time.

Explore `at` and email

All Operating Systems have the ability to schedule tasks, either as a one-time occurances or continuing events.

In Linux, one-time events are done with the `at` utility (as in, "do X at time Y"). Continuing events are scheduled with the `cron` utility (as in "chronos", Greek for "time"). Let's gain some basic experience with both.

1. Start up the pfSense_Reference VM and let it fully boot (all the way to the menu that gives you options).

2. Now, start up the CentOS_Reference VM. Make sure you have an IP address from the pfSense Router:

   ```
   $ ip addr
   ```

   Which should return an IP address in the 192.168.1.0/24 network. Ping an exteral IP address to verify you have an internet connection:

   ```
   $ ping www.google.com
   ```

3. First, we need to install at. This tool is often not installed by default, but it's very easy to get up and running. Enter this command to get at installed:

   ```
   $ sudo yum install at
   ```

4. In order to play with the `at` command, let's verify that it's daemon `atd` is up and running. Enter this command to get a list of all running processes, and filter it for the word "`atd`":

   ```
   $ ps -elf | grep "atd"
   ```

   What you should see is a line returned that has the last word as "`atd`" (you'll also likely see a line returned that has your grep command with the "atd" argument). The line that is the actual process entry is the `atd` daemon. If you see it, then we can continue. If you don't see it, you'll likely need to restart your VM, or reset it back to the original appliance state.

5. In order to use `at`, we need to feed it via stdin. Thus, we use `echo` to send our timed command as text into `at`. Enter this command to `ping` www.oregonstate.edu three times at one minute in the future from whenever you run it at:

   ```
   $ echo "ping -c 3 www.oregonstate.edu" | at now + 1 minute
   ```

   Linux will respond with a line about the scheduling. Note that `at` cannot resolve to times smaller than a minute: if you want to specify seconds, you'd have to add `sleep` with a semi-colon to your command. Go ahead and enter this in:

   ```
   $ echo "sleep 10; ping -c 4 www.orst.edu" | at now + 1 minute
   ```

   You could also use time terms like "`1145am June 12`" or "`tomorrow`": `at` is pretty flexible. Now, just wait!

6. Every few seconds or so, hit Enter on the bash prompt. You'll see that a new prompt is simply given, like normal. In about a minute, you'll hit Enter and see a line that says you have mail; this will happen for each job that completes:

   ```
   You have new mail in /var/spool/mail/centosuser
   ```

   Linux only checks for execution of `at` jobs just before it reprompts, so that's why we had to be entering commands (even if we just hit Enter) to get things to show up. Let's go read the mail! Enter this command to open up the mail:

   ```
   $ vi /var/spool/mail/centosuser
   ```

   This is a very primitive email system: all of your email is contained in this one file. As you run other at jobs, you'll see mail pile up, simply appended to this one file.

   To read this email in a way that makes sense, we're going to need to install an email client. I'm not even kidding.

7. Welcome to 1995's `mutt` email client! Enter this command to install it:

   ```
   $ sudo yum install mutt
   ```

8. Once it's installed, fire it up:

   ```
   $ mutt
   ```

Once it starts, it'll ask you if you want to create a Mail directory: just hit "n" for no. Next, you'll be presented with a list of email. Use the arrow keys to move around on any email in here, and hit Enter to open them up. Inside an email, hit "i" to get back to the inbox, and from the inbox, hit "d" to delete an email. This is how we used to read email. Hit "q" from the inbox when you're done reminiscing of a simpler time. Remember that if you get stuck, you can always hit CTRL+C to get out.

Find the email that relates to the jobs we kicked off. Open them up, and view the results.

QUESTION: Go answer Question 1 now.

Explore `cron`

Next, let's play around with the continuous event scheduling tool `cron`. It's just as old as `at` is, but is still used everyday to get things scheduled!

1. For our first trick, let's see if `cron` is already running:

   ```
   $ ps -elf | grep "cron"
   ```

   You should see that the `crond` daemon is already running, with parent process ID of 1.

2. Next, we need to understand how the `cron` system works. The `crond` daemon will check a special file, called the `crontab` ("chronos table"), every minute. If the current time matches any of the entries in the `crontab`, then the command is executed.

   In the crontab, each entry is one line. The format of the line is as follows

   ```
   a b c d e /path/to/script/or/command
   ```

   `a` is the minute that this command should be executed on (0 - 59)

   `b` is the hour that this command should be executed on (0 - 23)

   `c` is the day of the month that this command should be executed on (1 - 31)

   `d` is the month that this command should be executed on (1 - 12)

   `e` is the day of the week that this command should be executed on (0 - 7, 0 & 7 are both Sunday)

   A hyphen can be used to define a range, and multiple ranges or entries can be separated with a comma. Multiple commands can be separated with a semi-colon, like in bash. Importantly, you can also specify that the particular timing element doesn't matter, and the command for trigger for each one, with an asterisk (*).

   Here are some examples:

   ```
   * * * * * /bin/uptime; users  # Report uptime and logged in users
                                 # every minute

   0 12 1 * * /bin/uptime    # Report uptime at noon on the first day of
                             # each month
   ```

Let's set up an example. We want to tell Linux to run the uptime command every minute from 5pm to just before 10pm every weekday in May. To edit the crontab, we use the `crontab` command itself with the `-e` option, which in turn opens up `vi` for us to use. Enter this to start vi on the right file:

```
$ crontab -e
```

Now, enter in this line:

```
* 17-22 * 5 1-5 /bin/uptime
```

Save and exit the file with `:wq`. You do not have to restart `crond`, because it opens up the crontab once per minute to check the `crontab` file! Once this is running, and assuming that the date range specified above matches today, you should be getting an email every minute with an uptime report.

Open up `mutt` to see the inbox.

QUESTION: Go answer Question 2 now.

3. We need to remove the entry we made, so that the cron jobs stop. First, list the current `crontab` for our user:

   ```
   $ crontab -l
   ```

4. Open up the crontab:

   ```
   $ crontab -e
   ```

5. Delete our entry, and save and quit.

   QUESTION: Go answer Question 3 now.

Scheduled Ansible Deployment

This particular task is not given in detail as previous ones have been. Your task is is to follow the steps laid out in the Ansible lecture (using the same VMs as the rest of this Lab) and achieve the following:

1. Set up a pfSense_Router, which connects to the internet on interface 1, and manages a LAN on interface 2.
2. Set up a CentOS VM and Alpine VMs on the LAN created by the router VM.
3. Set up Ansible on the CentOS VM, provisioning the same web server as described in the lecture on the Alpine VM.
4. The CentOS VM shall deploy the apache web server to the Alpine VM as a scheduled task using cron every 15 seconds.
5. The deployed web server's HTML page needs to have the same `{{ template_run_date }}` variable defined, so that the page is visibly getting new data on the crontab schedule.

You'll be able to follow the steps in the Ansible lecture pretty closely to get this to work, and you'll want to have the 3 VMs reset to their base states to make things cleaner before you start. When you've got this functioning, answer the next question.

QUESTION: Go answer Question 4 now.

Questions

1) Get the TAs initials, showing that you are viewing the successful ping of www.orst.edu inside `mutt`. *(4 points)*
2) Get the TAs initials, showing that you are receiving emails by the minute inside `mutt` that contain uptime reports. *(4 points)*
3) Individual `crontab` entries are only processed once per minute. Write a `crontab` that will execute the `uptime` command every 10 seconds. Get the TAs initials, showing that you have this `crontab` created in your VM. Hint: This will require 6 different lines in the `crontab`. *(7 points)*
4) Get the TAs initials, showing that you can repeatedly execute: "`$ curl 192.168.1.XXX`" (which is targeting the Alpine VM), with no other commands needed, to display a web page that changes to a new timestamp every 15 seconds. *(25 points)*

You're done! To receive credit for this lab, you must turn in your answer sheet.