Purpose

The purpose of this exercise is to practice using cron to automatically run processes at predetermined times.

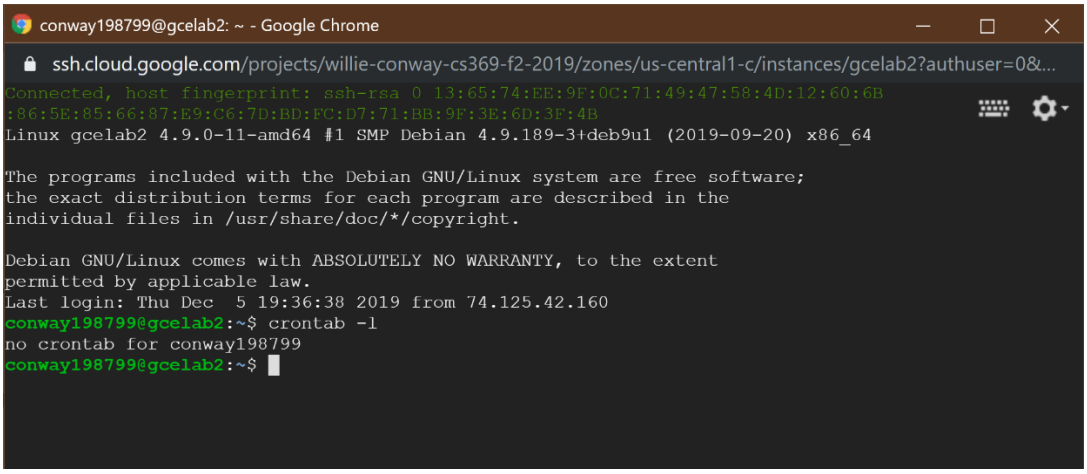
Preparation

A good site to read about cron and get several examples is A Beginner's Guide to Cron Jobs (Links to an external site.)

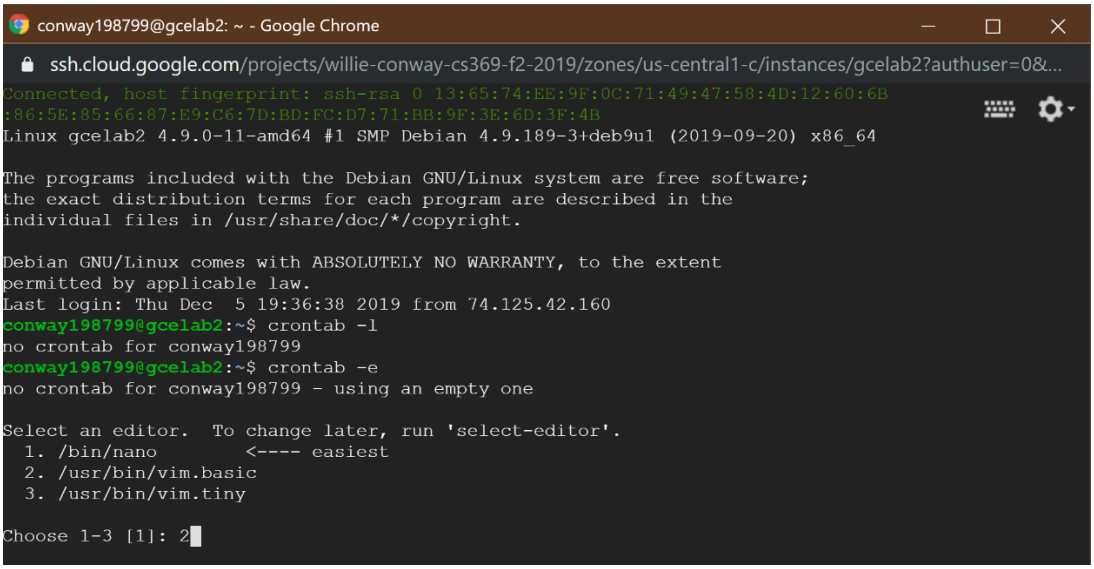
Assignment

Keep a Word document which will show your process throughout the lab. Include the contents of the crontab file and any other files that you modify.

Use the program crontab to create cron jobs that:

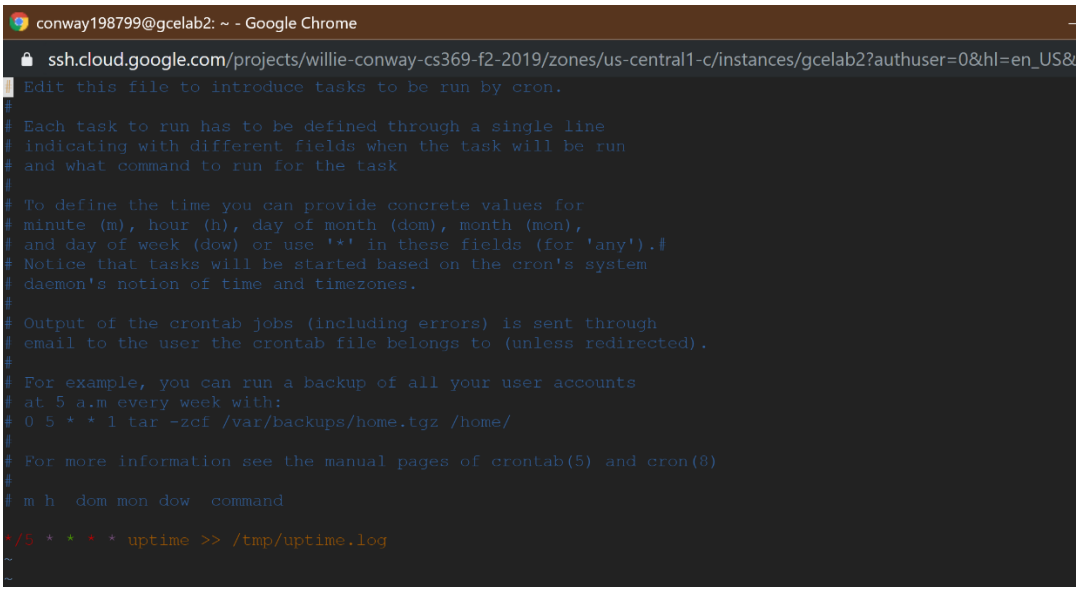


(The crontab command, found in Unix and Unix-like operating systems, is used to schedule commands to be executed periodically. Generally, crontab uses a daemon, crond , which runs constantly in the background and checks once a minute to see if any of the scheduled jobs need to be executed. The crontab -l command displays the current crontab on standard output of the user logged in cronjobs. As you can see user conway198799 has no cronjobs displayed.)

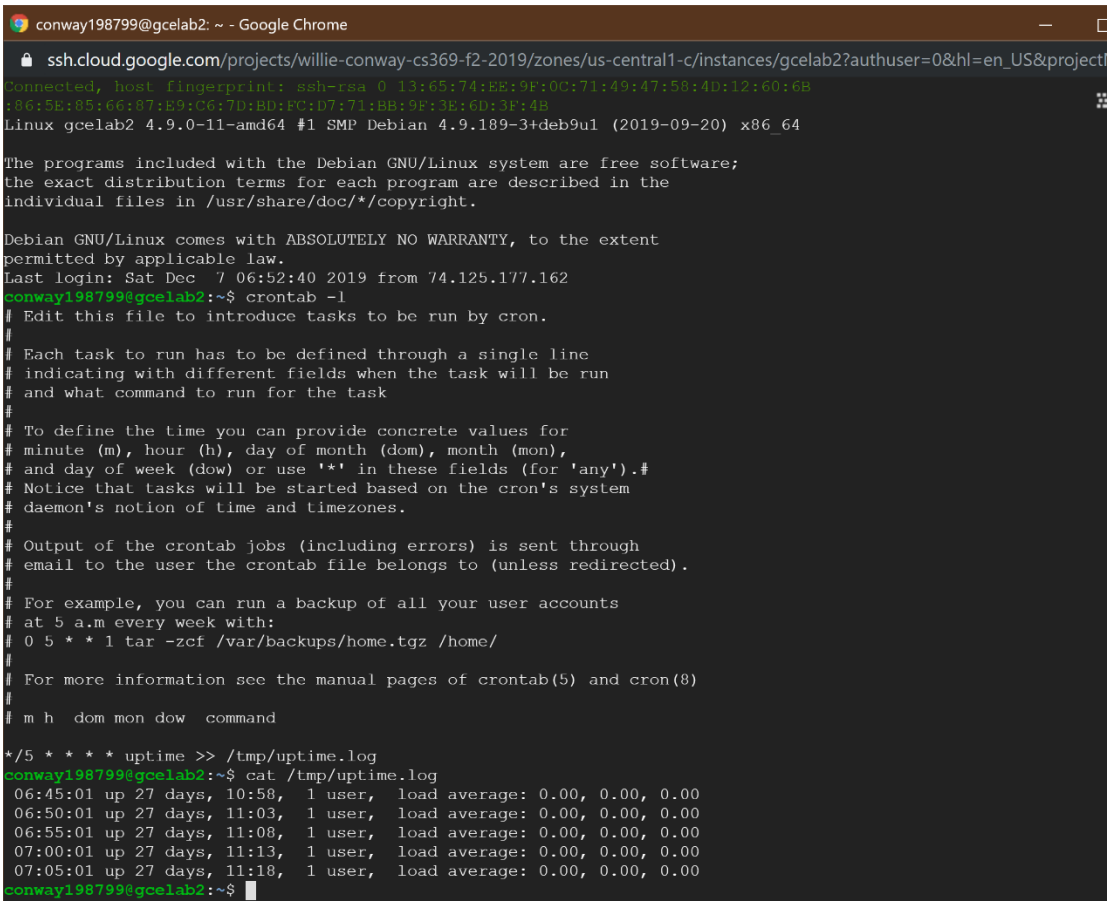


(The crontab -e command is used to edit the current crontab using the editor specified by the VISUAL or EDITOR environment variables. After you exit from the editor, the modified crontab will be installed automatically. I decided to choose vim.basic. This is option 2.)

1. Append the output of the command uptime to a file. This should run every 5 minutes.

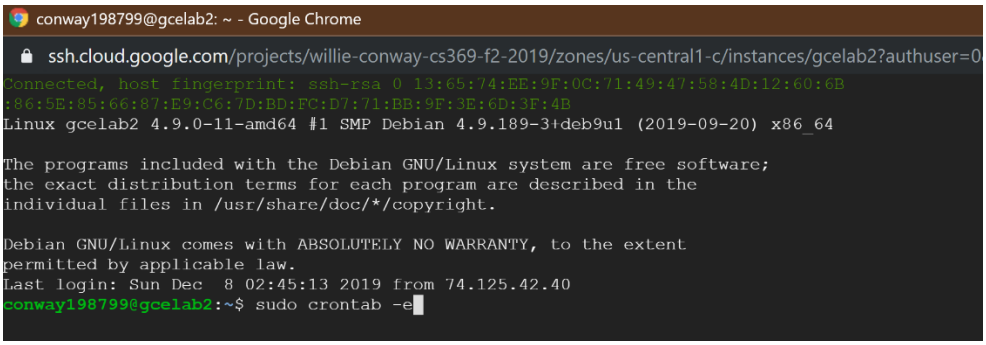


*(Appending the output of the command uptime to the file /tmp/uptime.log and running the cron job every five minutes using */5 **** uptime >> /tmp/uptime.log.)*

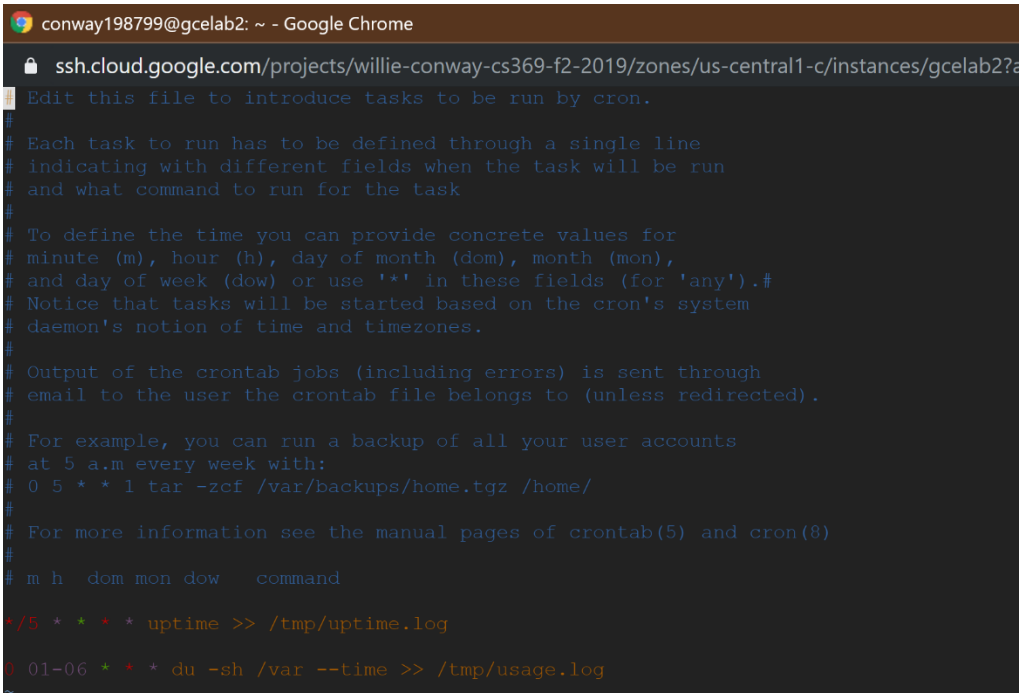


(Checking to see if cronjob posted with crontab -I and displaying the cronjob with cat /tmp/uptime.log.)

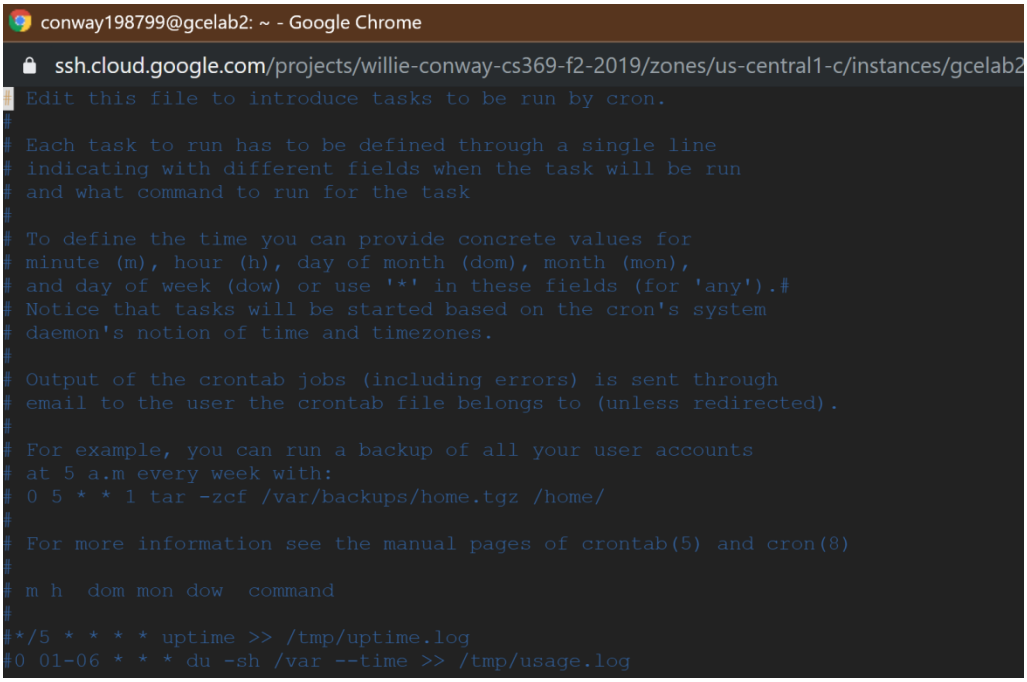
2. Appends the disk space used in /var to a different file. Include the date and time. This should run 6 times a day, at evenly spaced intervals. You can choose the times.



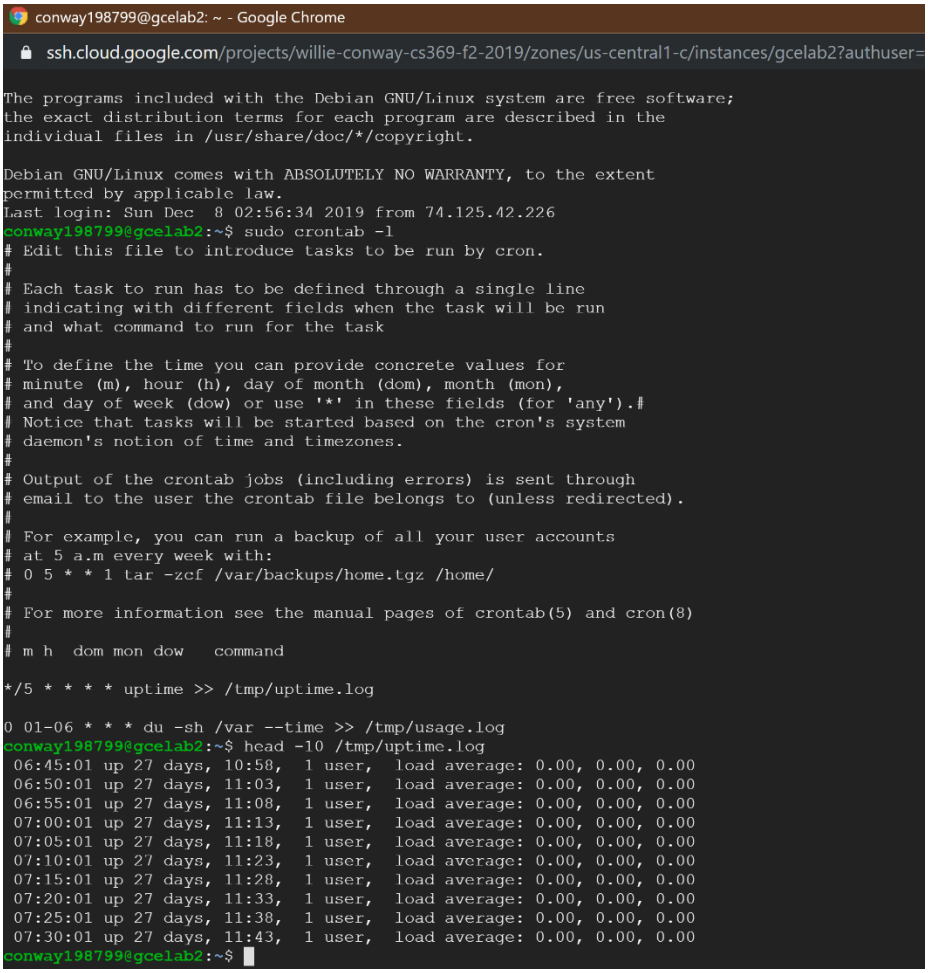
(Since I figure this job may take a sudo command, I decided to change into roots crontab using sudo crontab -e. i*f you are putting the script from one of the cron directories (/etc/cron. *) then you don't need to use sudo as that is running as root. If you are using crontab, then you will want to use root's crontab. This will run it as root, and also not need sudo.)*



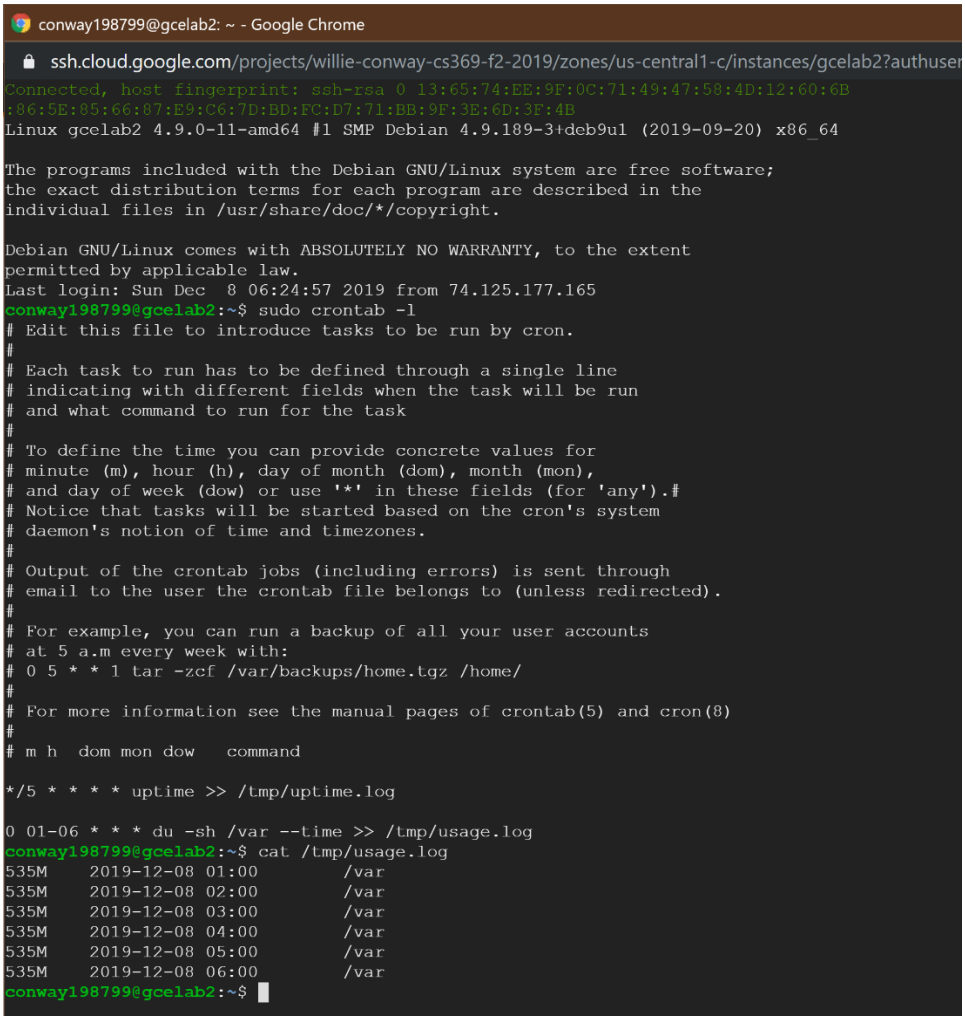
*(I decided to transfer my other cronjob over so that I could have a cronjob list. For the second cronjob I appended the disk space used in /var to a different file. Include the date and time. Plus, made sure the cronjob would run 6 times a day, at evenly spaced intervals. To do this I entered 0 01-6 * * * du -sh /var --time >> /tmp/usage.log. du command in Linux with examples. du command, short for disk usage, is used to estimate file space usage. The du command can be used to track the files and directories which are consuming excessive amount of space on hard disk drive.)*



(Since I decided to use the roots crontab to post my cronjobs, I decided to comment out the following jobs in my crontab file so there won't be ay interruptions.)

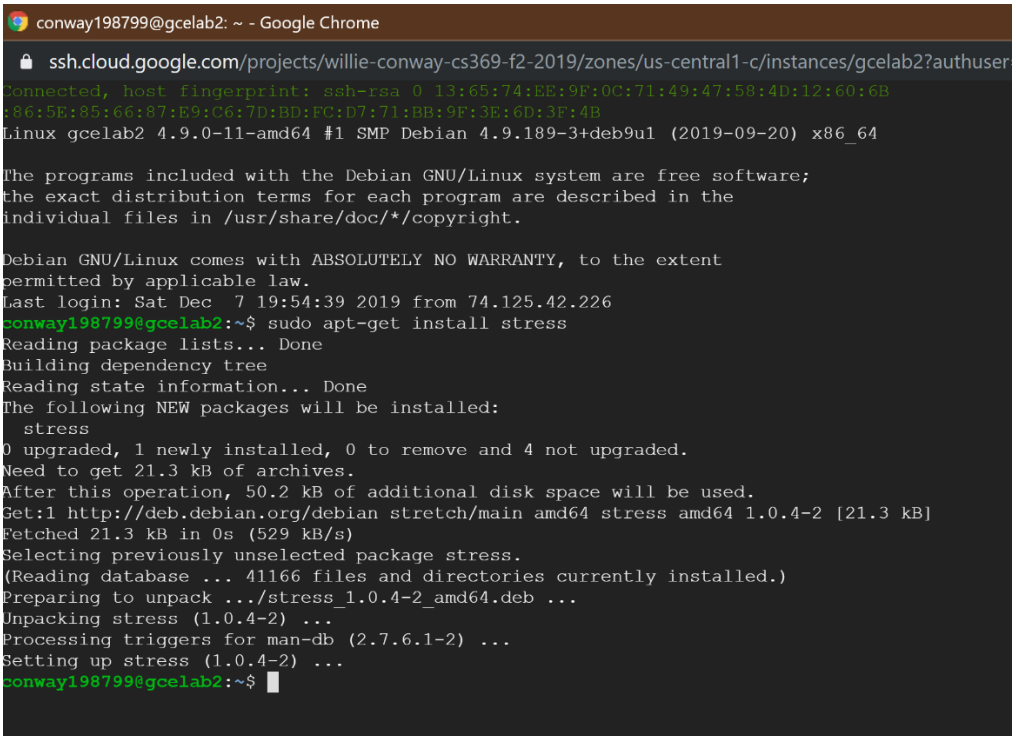


(Checking to see if cronjob posted with crontab -I and displaying the cronjob with cat /tmp/uptime.log.)



(Checking to see if cronjob posted with crontab -l and displaying the cronjob with cat /tmp/usage.log.)

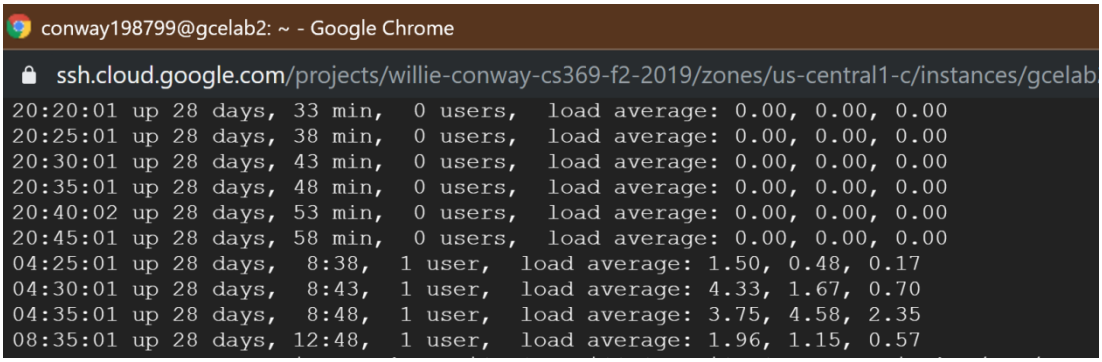
3. Searches through the file created in step 1 and outputs any lines where the 5-minute load average is above 2.00 to a third file. This program should run once a day between 1 and 3 am. (You should make sure that your system has a high 5-minute load average at least once so you can test your script. See the program stress for a nice tool that can stress test your system.)



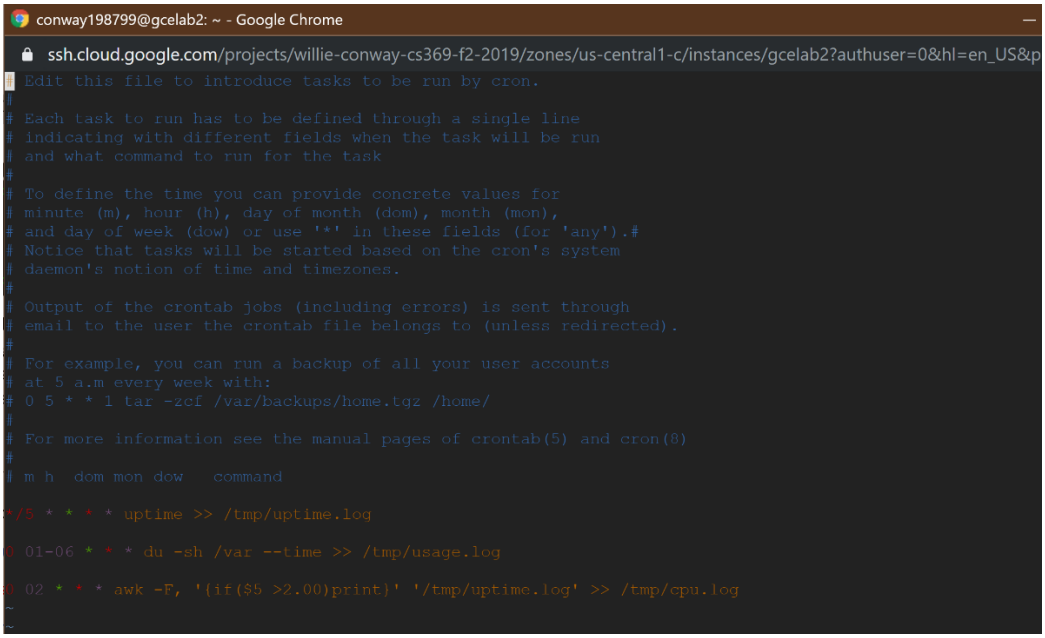
(Installing stress system using sudo apt-get install stress. stress – is a workload generator tool designed to subject your system to a configurable measure of CPU, memory, I/O and disk stress.)

```
conway198799@gcelab2:~$ sudo stress --vm 1 --timeout 60s
stress: info: [12284] dispatching hogs: 0 cpu, 0 io, 1 vm, 0 hdd
stress: info: [12284] successful run completed in 60s
conway198799@gcelab2:~$ uptime
 08:18:58 up 28 days, 12:32,  1 user,  load average: 0.49, 0.17, 0.06
conway198799@gcelab2:~$ sudo stress --vm 1 --timeout 20s
stress: info: [12323] dispatching hogs: 0 cpu, 0 io, 1 vm, 0 hdd
stress: info: [12323] successful run completed in 20s
conway198799@gcelab2:~$ uptime
 08:20:43 up 28 days, 12:34,  1 user,  load average: 0.32, 0.18, 0.08
conway198799@gcelab2:~$ sudo stress --cpu --timeout 20
stress: FAIL: [12342] (160) missing argument to option '--cpu'
conway198799@gcelab2:~$ sudo stress --cpu 8 --timeout 20s
stress: info: [12344] dispatching hogs: 8 cpu, 0 io, 0 vm, 0 hdd
stress: info: [12344] successful run completed in 20s
conway198799@gcelab2:~$ uptime
 08:22:24 up 28 days, 12:36,  1 user,  load average: 1.98, 0.63, 0.24
conway198799@gcelab2:~$ uptime
 08:23:45 up 28 days, 12:37,  1 user,  load average: 0.52, 0.48, 0.21
conway198799@gcelab2:~$ wat
```

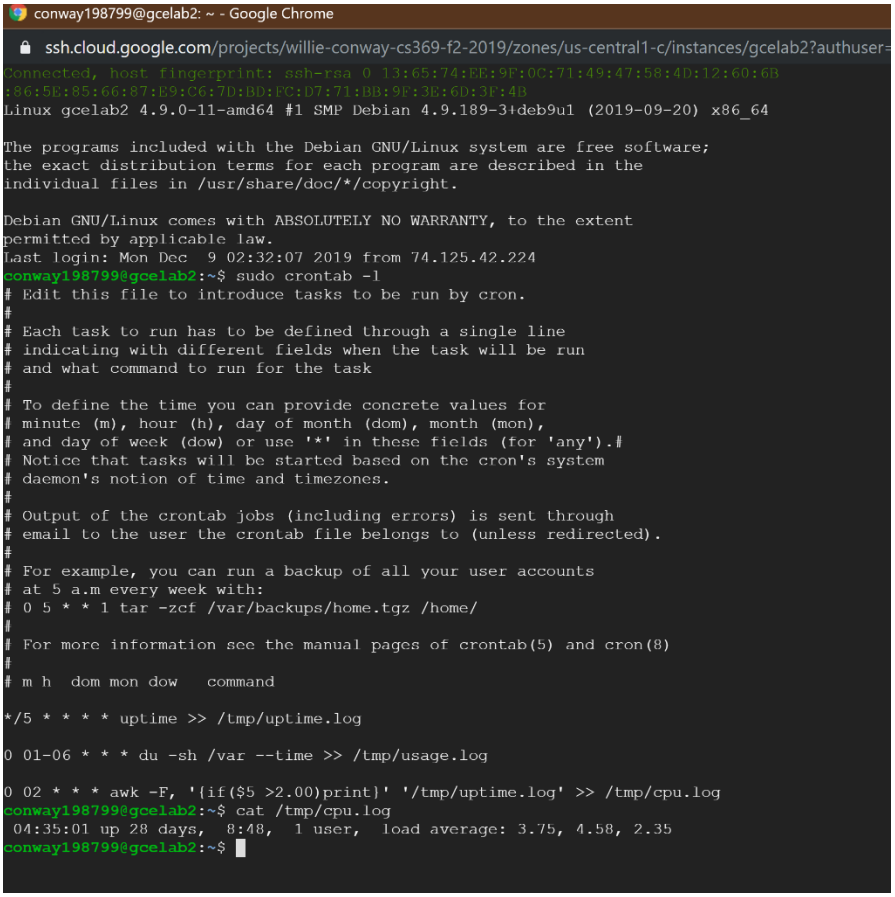
(Begin stress testing the system.)



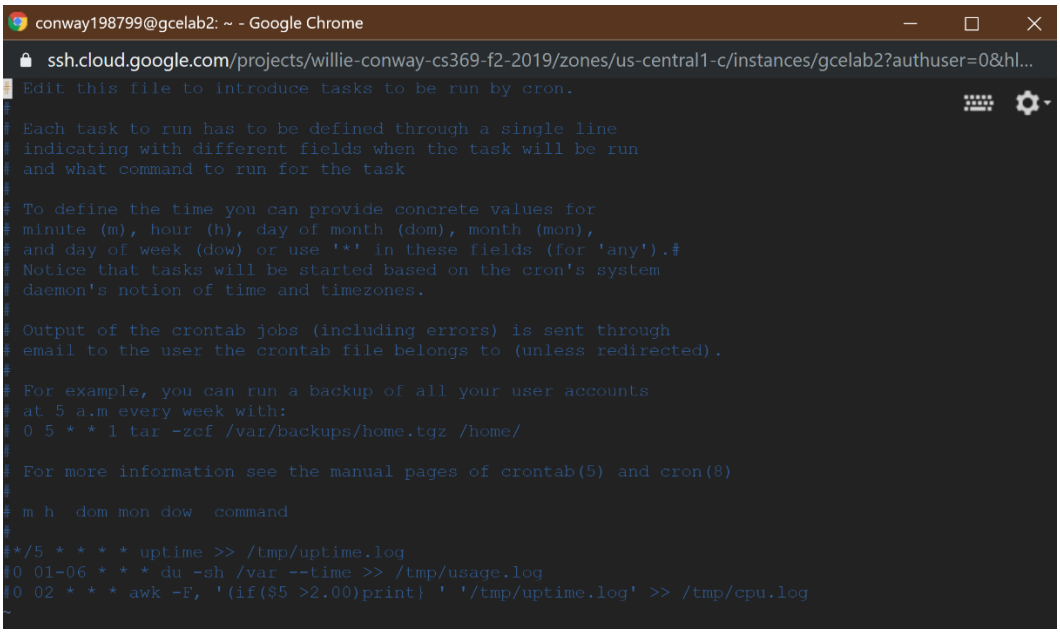
(Load averages columns after stress test.)



(Appending a cronjob that searches through the file created in step 1 and outputs any lines where the 5-minute load average is above 2.00 to a third file. This program runs once a day between 1 and 3 am. I posted my cronjob to rub a 0200 which is equal to 2 a.m in military time. Since the clock on our OS does use milliary time. (I made sure that the system has a high 5-minute load average at least once so I can test my script.)



(Checking to see if cronjob posted with crontab -l and displaying the cronjob with cat /tmp/cpu.log.)



(Tranfering the last cronjob over to my crontab and saving it as a comment.

At the end of your Word document, include the answers to the following questions:

1. The location of your crontab file

   (my crontab file)

   /var/spool/cron/crontabs/conway198799

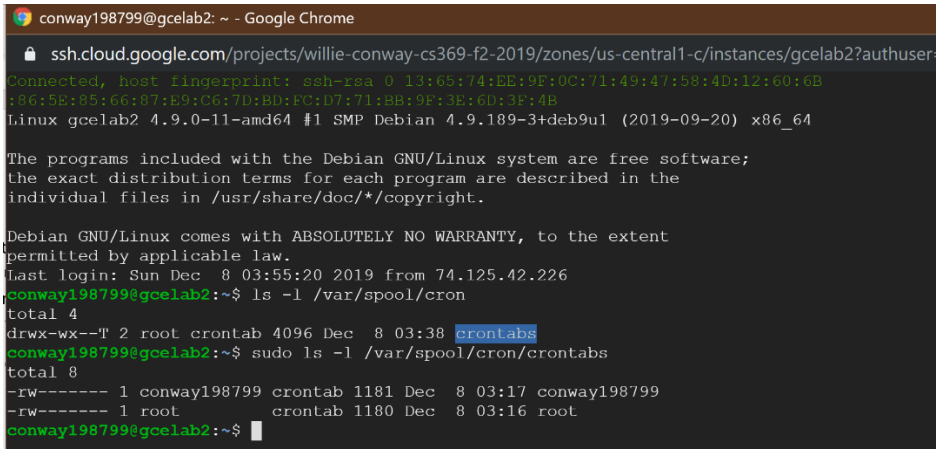
   (I edited my crontab file with comments of the 3 tasks due to not being able to use sudo commands in my crontab.)

   (root crontab file)

   /var/spool/cron/crontabs/root

   (I edited the root crontab with the 3 tasks due to being able to have privileges to use sudo commands and list all 3 tasks together.)

2. The permissions on the file and directory



   Use ls -l command. Each of the crontab files are in the /var/spool/cron/crontabs directory. The ls -l command list's information about the FILEs (the current directory by default) using a long listing format showing files/directories and their permissions.)

   The permissions on the /crontabs directory is: drwx-wx--T

   - The letter d indicates that the file is a directory, which is basically a special kind of file.
   - r refers to the read permission.
   - w refers to the write permission.
   - x refers to the execute permission.
   - A dash **(-)** indicates that the file is a regular file.

The permissions on each of the files are: -rw-------

- A dash **(-)** indicates that the file is a regular file.
- **r** - refers to the read permission.
- **w** - refers to the write permission.

The first trio of characters after the file types in a file list (rwx),(-rw) shows the permissions for the user, or file owner.

The next trio of characters (-wx),(---) shows the permissions for the group category.

The last trio of characters (---),(--T) shows the permissions for the final category, other. When the 'T' is present, it means that 'x' permission is has not been given on the directory/file. If you give 'x' permission, then a 't' will be shown. You can set the 'T'/'t' with a 4-character permission, rather than a 3 character.

Notice the specific order to the permissions in a trio: read, write, execute. A dash in place of a letter for a permission means that category doesn't have that permission.

3. In a sentence or two, what did you learn? *This was very difficult project ass I had to learn a lot about appending files and utilizing cron to create cronjobs. I had to acknowledge that Cron is a daemon, meaning that it works in the background to execute non-interactive tasks. Other than learning new commands to access the disk usage of a directory, I found task 3 quite challenging learning more about uptime and stressing my instance to get a standard load average. cron is a Linux utility which schedules a command or script on your server to run automatically at a specified time and date. A cron job is the scheduled task itself. Cron jobs can be very useful to automate repetitive tasks.*

4. In a sentence or two, what did you like about this project? *What I found interesting and liked about this project was how a simple task like cron can be used to handle daily or yearly task. This would be a handle system for a system or network administrator to use to make sure logs are always being conducted along with updates. It's easier to write the code and to manage its operation.*

5. In a sentence or two, what did you find confusing or would like to see done differently regarding this project? *The most thing I found confusing about this project was the third task. It took my 3 days of staying up to realize that I needed to use an if stament to process and search through the file. I'm no master at using the awk command. So, I had to encourage myself to learn. I tried so many other ways, but they all failed as I was looking for a specific load average above 2.00.*