#### **Step 1: Create, Extract, Compress, and Manage tar Backup Archives**

Creating tar archives is something you must do everyday in your role at Credico Inc. In this section, you will extract and exclude specific files and directories to help speed up your workflow.

To get started, navigate to the ~/Projects directory where your downloaded TarDocs.tar archive file should be.

1. Extract the TarDocs.tar archive file into the current directory (~/Projects). Afterwards, list the directory's contents with ls to verify that you have extracted the archive properly.  
   * Note that because we want to preserve the directory structure of our archive, we do not have to specify a target directory to extract to.
   * Note that when you run ls you should see a new ~/Projects/TarDocs directory with five new subdirectories under TarDocs/.

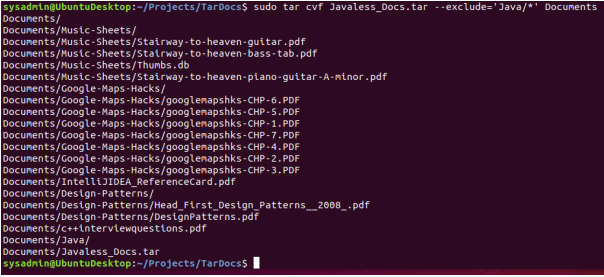
1. Command to extract the TarDocs.tar archive to the current directory:



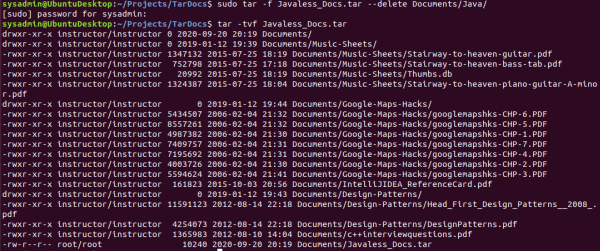
Verify that there is a Java subdirectory in the TarDocs/Documents folder by running: ls -l ~/Projects/TarDocs/Documents/.

1. Create a tar archive called Javaless\_Docs.tar that excludes the Java directory from the newly extracted TarDocs/Document/ directory.  
   * If you've executed this command properly, you should have a Javaless\_Docs.tar archive in the ~/Projects folder.

2. Command to create the Javaless\_Doc.tar archive from the TarDocs/ directory, while excluding the TarDocs/Documents/Java directory:



1. Verify that this new Javaless\_Docs.tar archive does not contain the Java subdirectory by using tar to list the contents of Javaless\_Docs.tar and then piping grep to search for Java.  
   3. Command to ensure Java/ is not in the new Javaless\_Docs.tar archive:



**Bonus**

* Create an incremental archive called logs\_backup.tar.gz that contains only changed files by examining the snapshot.file for the /var/log directory. You will need sudo for this command.

Command to create an incremental archive called logs\_backup\_tar.gz with only changed files to snapshot.file for the /var/log directory:



#### **Step 2: Create, Manage, and Automate Cron Jobs**

In response to a ransomware attack, you have been tasked with creating an archiving and backup scheme to mitigate against CryptoLocker malware. This attack would encrypt the entire server’s hard disk and can only be unlocked using a 256-bit digital key after a Bitcoin payment is delivered.

For this task, you'll need to create an archiving cron job using the following specifications:

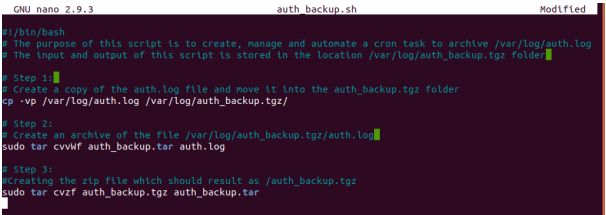
* This cron job should create an archive of the following file: /var/log/auth.log.
* The filename and location of the archive should be: /auth\_backup.tgz.
* The archiving process should be scheduled to run every Wednesday at 6 a.m.
* Use the correct archiving zip option to compress the archive using gzip.

1. To get started creating cron jobs, run the command crontab -e. Make sure that your cron job line includes the following:

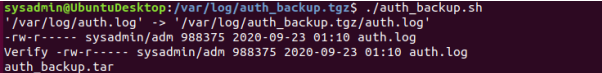
* + The schedule (minute, hour, etc.) in cron format. - **Hint:** Reference the helpful site [crontab.guru](https://crontab.guru/) as needed.
  + An archive (tar) command with three options.
  + The path to save the archive to
  + The path of the file to archive.

1 For this step, I have broken down each process. I didn’t want to mess up the auth.log file. I have created a copy with my auth\_backup.sh script, then actioned the tar command accordingly.

Initial script as follows:



Testing of script:



Comparing file size From /var/log/



Backup auth.log file in /var/log/auth\_backup.tgz/

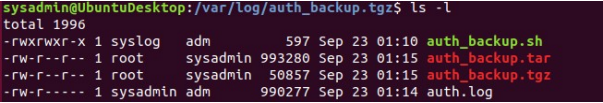


A small increment of data is observed between the log files which is understandable as the increase in the main system auth.log is constantly updating and the time stamp shows that there are 4 additional minutes of data captured. Hence an increase in the size of the main auth.log.





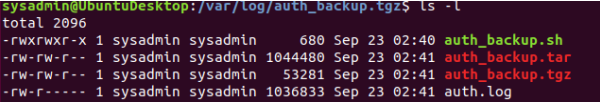
Checking if tar files are created:



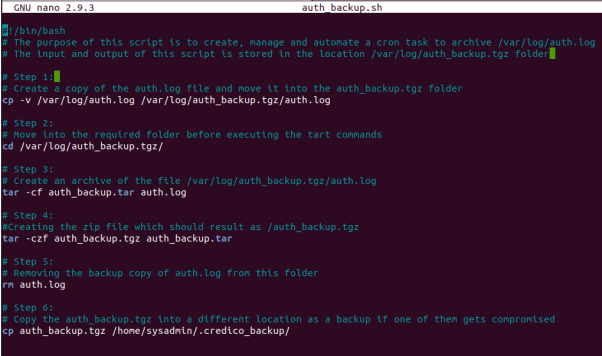
Now, automating the above steps via crontab to run every Wednesday at 6am. Before creating the final crontab, I ran some tests to see if the required outputs are generated. Set crontab to display at 2:41am on each Wednesday (as a test).



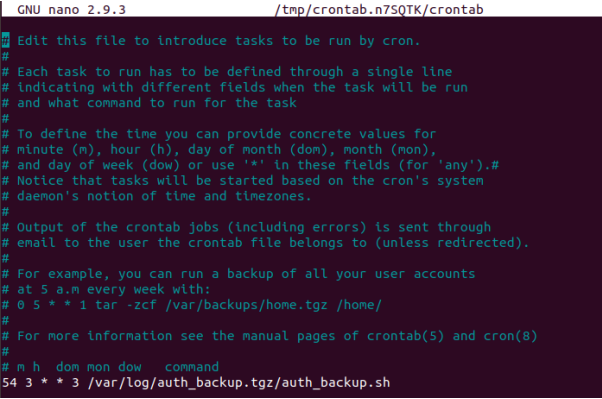
Output below:



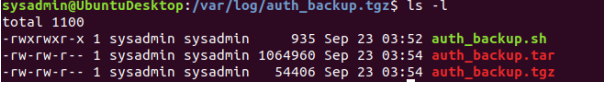
PS: I had some issues with the permissions. I have made the whole folder to carry forward sysadmin’s permissions and all commands were executed correctly. Final script Re-designing the auth\_backup.sh script to remove the auth.log after the creation of the backup to protect the integrity of the file from hackers and copy the auth\_backup.tgz in other location



Testing of script: Set crontab to trigger at 3:54 on every Wednesday (as a test)



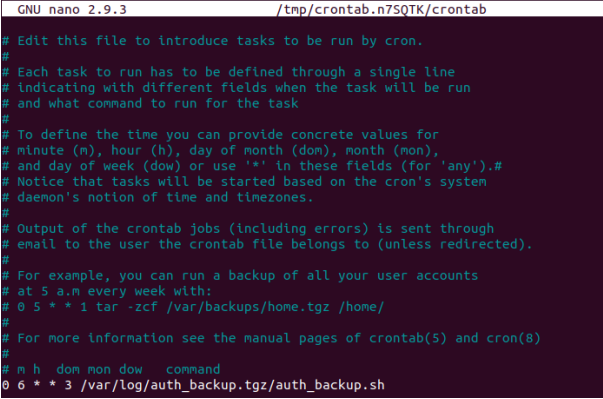
auth.log removed from the below directory.



auth\_backup.tgz copied into a hidden folder away from the directories where the logs are kept.

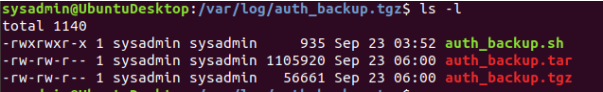


Final Crontab Automation: The final command for the crontab to run every Wednesday at 6am.



Test results:





It appears that the crontab task is working fine as displayed above.

#### **Step 3: Write Basic Bash Scripts**

Portions of the Gramm-Leach-Bliley Act require organizations to maintain a regular backup regimen for the safe and secure storage of financial data.

You'll first need to set up multiple backup directories. Each directory will be dedicated to housing text files that you will create with different kinds of system information.

For example, the directory freemem will be used to store **free memory** system information files.

1. Using brace expansion, create the following four directories:  
   * ~/backups/freemem
   * ~/backups/diskuse
   * ~/backups/openlist
   * ~/backups/freedisk





1. **Note**: Remember that brace expansion uses the following format: ~/exampledirectory/{subdirectory1,subdirectory2,etc}

Now you will create a script that will execute various Linux tools to parse information about the system. Each of these tools should output results to a text file inside its respective system information directory.

* For example: cpu\_usage\_tool > ~/backups/cpuuse/cpu\_usage.txt

In the above example, the cpu\_usage\_tool command will output CPU usage information into a cpu\_usage.txt file.

To get started with setting up your script up in your home directory, do the following:

* Navigate to your home directory by running: cd ~/
* Run the command nano system.sh to open a new Nano window.

**Note**: If you're unsure how to get started, we included a system.sh starter file. Use that as a guide.

1. Edit the system.sh script file so that it that does the following:  
   * Prints the amount of free memory on the system and saves it to ~/backups/freemem/free\_mem.txt.
   * Prints disk usage and saves it to ~/backups/diskuse/disk\_usage.txt.
   * Lists all open files and saves it to ~/backups/openlist/open\_list.txt.
   * Prints file system disk space statistics and saves it to ~/backups/freedisk/free\_disk.txt.
2. **Note**: For the free memory, disk usage, and free disk commands, make sure you use the -h option to make the output human-readable.
3. Save this file and make sure to change or modify the system.sh file permissions so that it is executable.

You should now have an executable system.sh file within your home ~/ directory.

* Test the script with sudo ./system.sh.
* **Note**: If it appears, ignore the warning: lsof: WARNING: can't stat() fuse.gvfsd-fuse file system /run/user/1001/gvfs Output information may be incomplete.

**Optional**

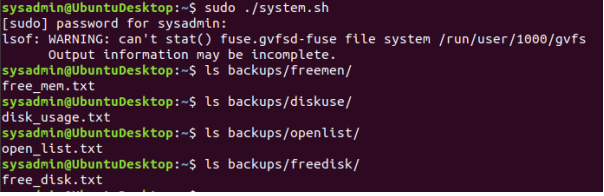
* Confirm the script ran properly by navigating to any of subdirectories in the ~/backup/ directory and using cat <filename> to view the contents of the backup files.

2. Paste your system.sh script edits below: 3. #!/bin/bash # INSTRUCTIONS: Edit the following placeholder command and output file paths # For example: cpu\_usage\_tool > ~/backups/cpuuse/cpu\_usage.txt # The cpu\_usage\_tool is the command and ~/backups/cpuuse/cpu\_usage.txt is the file path # In the above example, the `cpu\_usage\_tool` command will output CPU usage information into a `cpu\_usage.txt` file. # Do not forget to use the -h option for free memory, disk usage, and free disk space # Free memory output to a free\_mem.txt file free -h > ~/backups/freemen/free\_mem.txt # Disk usage output to a disk\_usage.txt file du -h > ~/backups/diskuse/disk\_usage.txt # List open files to a open\_list.txt file lsof > ~/backups/openlist/open\_list.txt # Free disk space to a free\_disk.txt file df -h > ~/backups/freedisk/free\_disk.txt

4. Command to make the system.sh script executable:



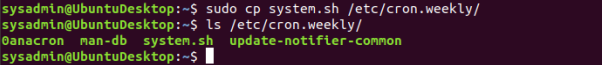
Optional Commands to test the script and confirm its execution:



**Bonus**

* Automate your script system.sh by adding it to the weekly system-wide cron directory.

Command to copy system to system-wide cron directory:



#### **Step 4. Manage Log File Sizes**

You realize that the spam messages are making the size of the log files unmanageable.

You’ve decided to implement log rotation in order to preserve log entries and keep log file sizes more manageable. You’ve also chosen to compress logs during rotation to preserve disk space and lower costs.

1. Run sudo nano /etc/logrotate.conf to edit the logrotate config file. You don't need to work out of any specific directory as you are using the full configuration file path.

Add your config file edits below:



1. Configure a log rotation scheme that backs up authentication messages to the /var/log/auth.log directory using the following settings:  
   * Rotates weekly.
   * Rotates only the seven most recent logs.
   * Does not rotate empty logs.
   * Delays compression.
   * Skips error messages for missing logs and continues to the next log.

1. Don't forget to surround your rotation rules with curly braces {}.