Overview

We need to create two bash scripts so that we can use them if we need to make a secure backup of our system or we need to restore the backup to our system.

Steps:

- How to validate parameters passed to the script
- Backup function
- Restore function

How was I thinking while writing the code:

*****You should check that the script received 4 parameters from the user and validate the parameters************************

- The first thing I wanted to validate the parameters so I made this script:

The script on based on while true to check all params. But it is not useful to use it because we want to pass the params in one batch through a script

- We need to make sure that all params = 4 . I used $# to know how many parameters passed through if.
- I need to check if the value of $1 and $2 directory or not. I used if [-d $]
- I need to check if the value of $4 in num or not. Days must be numbers

****

****in the backup function****

(Your program should print a help message indicating how it should be used if it is invoked without any parameters.)

- If the script run without anything . I used echo to print notes to help how to input the values

(You should store all the parameters passed via the command line into bash variables.)

- Sorurce= $1
- Destintion= $2 etc

(take a snapshot of the full date and store it in a bash variable to be used later. You should replace any white space or colon with an underscore as we will use this variable to create directory names and file names; you can use sed to do that)

- I tried thid solution but it takes alot of params

now=$(date)

now2=${now// /_}

now3=${now2//:/_}

- So I used this :now=$(date "+%Y_%m_%d_%H_%M")

(You need to create a directory whose name should be equivalent to the date taken in point #4, under the backup directory provided as the second command-line parameter)

- I used : mkdir $dest/$now

(Your script should loop over all directories under the backup directory provided as the first user command-line parameter and check for the modification date to backup only modified files within the number of days specified by the fourth parameter)

- i tried this to backup the lasted modified files , it runs very well but we don't need all files in one tar

```
 tar -czvf $dest/$(date "+%a_%Y_%m_%d_%k_%M").tar.gz `find . -type f  -mtime -$days`
```

- I used for to loop over the dir then I used if to differentiate it's a dir of file:

```
for dir in *; do

if [ -d "$dir" ]; then

fi

 if [  -f "$dir" ]; then

          fi

   done
```

(You should create a tar.gz file using the tar command and the necessary switches under the created new backup directory. The filename should be <original directory name>_<date>.tgz. The “date” is the date acquired in point #4)

- In the loop I added the tar find line to archive all the modified fiels:

```
name=$(echo "$dir"| sed s/\ /_/g)_$now

find . -type f -mtime -$days | xargs tar -cvzf $dest/$now/$name.tar.gz
```

- And for files in the main dir: find . -type f -mtime -$days | xargs tar -cvzf $dest/$now/file_$now.tar.gz

(Within the loop, use the gnupg tool to encrypt the file created in point #7, using the provided key on the command line, in a new file with the same name followed by ".gpg".)

- This line to encrypt the tar:

```
echo "$key" | gpg -c --batch --yes --passphrase-fd 0 $dest/$now/file_$now.tar.gz
```

(you need to delete the original tar file and keep the encrypted one)

- `rm -rf $dest/$now/file_$now.tar.gz`

(After you are done with all the directories, you should enumerate all the files located directly under the backup main directory and group them into one tar.gz file and encrypt it in the same way.)

- Like the previous steps I made it like this :

```
find . -type f -name "*.gpg" | xargs tar -cvzf $now.tgz

echo "$key" | gpg -c --batch --yes --passphrase-fd 0 $now.tgz

 rm -rf $now.tgz
```

(After the backup is done, you should copy the backup into a remote server using scp.)

I used ssh to access the remote server using key pair

- `scp -i ~/main.pem $dest/$now.tgz.gpg ubuntu@ip-172-31-31-82:/home/ubuntu/data`

****

****in the restore function****

(Create a temp directory under the restore directory; command-line parameter #2.)

- mkdir $des/temp

(Loop over all the files in the backup directory; command-line parameter #1.)

- **I used for and if to loop**

```
Cd $sor

for file in *; do

    if [ -f "$file" ]; then

        if [[ $file == *.gpg ]]; then

        fi

    Fi

done
```

(Use gnupg tool to decrypt the files one by one and store the resulting file under the tempdirectory created in point #2. Use the decryption key provided via the command-line user input #3.)

- Decrypted by this line:
- echo "$key" | gpg --batch --yes --passphrase-fd 0 --decrypt --output "$des/temp/alaa.tgz" $file

(Loop over the files stored in the temp directory and extract them one by one under the restore directory; command-line parameter #2.)

- We can loop here without for using find
- find $des/temp/ -type f -iname "*.tgz" -exec tar -xvf {} +

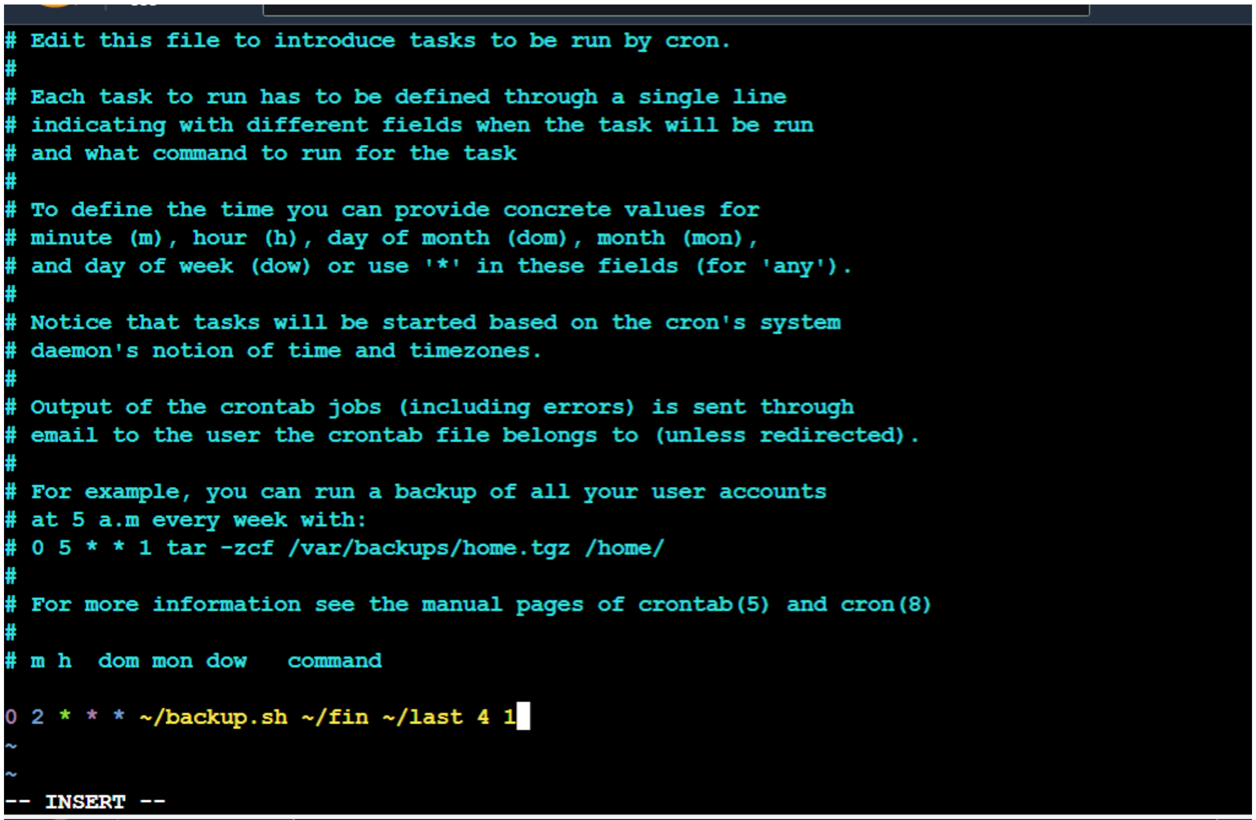
****

"you should amend the cron configuration files such that the backup script is executed every day:

1. tap crontab –e
2. Add this line :

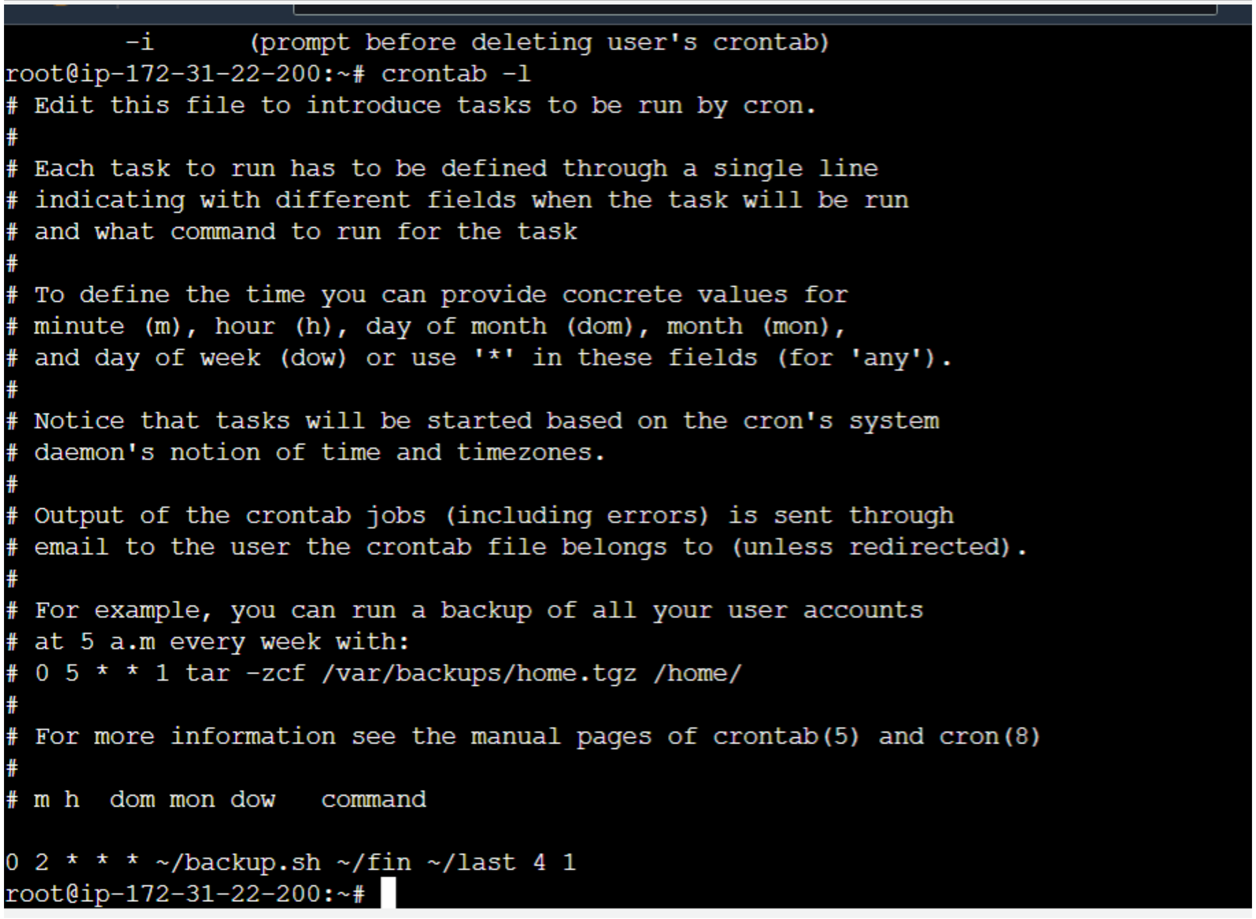
0 0 * * * /path/to/backup.sh <source_directory> <backup_directory> <encryption_key> <days>

0 0 * * * ~/backup.sh ~/fin ~/last 4 1



```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

0 2 * * * ~/backup.sh ~/fin ~/last 4 1
~
~
-- INSERT --
```



```
        -i       (prompt before deleting user's crontab)
root@ip-172-31-22-200:~# crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

0 2 * * * ~/backup.sh ~/fin ~/last 4 1
root@ip-172-31-22-200:~#
```