

Real Time Shell Scripts Example ::

1 . Take backup of a particular folder

Create two folders as the **source** and **destination**. Then, inside the source folder, create some files. This source folder will be used as a backup folder, and the destination folder where we will take the backup.

```
ubuntu@ip-172-31-19-13:~$ mkdir source
ubuntu@ip-172-31-19-13:~$ cd source/
ubuntu@ip-172-31-19-13:~/source$ touch file1.txt file2.txt file3.txt file4.txt
ubuntu@ip-172-31-19-13:~/source$ ls
file1.txt file2.txt file3.txt file4.txt
ubuntu@ip-172-31-19-13:~/source$ cd ..
ubuntu@ip-172-31-19-13:~$ pwd
/home/ubuntu
ubuntu@ip-172-31-19-13:~$
```

Create a script file as **backup.sh** and set the **source folder path** in the script [for me its **/home/ubuntu/source**] and also mention the **destination** [**/home/ubuntu/destination**] folder where the backup will be taken .

Explanation

- **SOURCE:** The directory to be backed up.
- **DESTINATION:** The directory where the backup will be stored.
- **DATE:** Captures the current date and time to create a unique backup folder.
- **mkdir -p \$DESTINATION/\$DATE:** Creates the backup directory if it does not exist.
- **cp -r \$SOURCE \$DESTINATION/\$DATE:** Copies the contents of the source directory to the backup directory.
- **echo "Backup completed on \$DATE":** Outputs a message indicating the completion of the backup.

```
#!/bin/bash

SOURCE="/home/ubuntu/source"

DESTINATION="/home/ubuntu/destination"

DATE=$(date +%Y-%m-%d_%H-%M-%S)

# Create backup directory and copy files
mkdir -p $DESTINATION/$DATE

cp -r $SOURCE $DESTINATION/$DATE

echo "Backup completed on $DATE"

~
~
~
~
~
~
~
~
```

Now give the **executable permission** to the **backup file** and execute the script as **/backup.sh** . Then check the destination folder, and you will see that one backup is there.

```
ubuntu@ip-172-31-19-13:~$ chmod +x backup.sh
ubuntu@ip-172-31-19-13:~$ ./backup.sh
Backup completed on 2024-05-23_13-44-09
ubuntu@ip-172-31-19-13:~$ ls
backup.sh  destination  source
ubuntu@ip-172-31-19-13:~$ cd destination/
ubuntu@ip-172-31-19-13:~/destination$ ls
2024-05-23_13-44-09
ubuntu@ip-172-31-19-13:~/destination$ cd 2024-05-23_13-44-09/
ubuntu@ip-172-31-19-13:~/destination/2024-05-23_13-44-09$ ls
source
ubuntu@ip-172-31-19-13:~/destination/2024-05-23_13-44-09$ cd source/
ubuntu@ip-172-31-19-13:~/destination/2024-05-23_13-44-09/source$ ls
file1.txt file2.txt file3.txt file4.txt
ubuntu@ip-172-31-19-13:~/destination/2024-05-23_13-44-09/source$
```

Another process to execute the script by using the **cron job** . Open the **crontab -e** file in editing mode.

```
ubuntu@ip-172-31-19-13:~$ crontab -e
no crontab for ubuntu - using an empty one

Select an editor. To change later, run 'select-editor'.
 1. /bin/nano      <---- easiest
 2. /usr/bin/vim.basic
 3. /usr/bin/vim.tiny
 4. /bin/ed

Choose 1-4 [1]: 1
crontab: installing new crontab
```

Now, you can set the cron job as per your requirements. If I want to execute the script **every minute**, the cron job will be as follows:

```
#
# 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
* * * * * /home/ubuntu/backup.sh
```

^G Help	^O Write Out	^W Where Is	^K Cut	^T Execute	^C Lo
^X Exit	^R Read File	^_\ Replace	^U Paste	^J Justify	^/ Go

After some time, if you check the **destination folder**, you will see another backup of the source file inside the date folder.

```
ubuntu@ip-172-31-19-13:~$ cd destination/  
ubuntu@ip-172-31-19-13:~/destination$ ls  
2024-05-23_13-44-09 2024-05-23_13-53-01  
ubuntu@ip-172-31-19-13:~/destination$ cd 2024-05-23_13-53-01/  
ubuntu@ip-172-31-19-13:~/destination/2024-05-23_13-53-01$ ls  
source  
ubuntu@ip-172-31-19-13:~/destination/2024-05-23_13-53-01$ cd source/  
ubuntu@ip-172-31-19-13:~/destination/2024-05-23_13-53-01/source$ ls  
file1.txt file2.txt file3.txt file4.txt  
ubuntu@ip-172-31-19-13:~/destination/2024-05-23_13-53-01/source$
```

2. To check whether a Service is running or not in your system .

Create a script file called as **Service-health-check.sh** and this is the script.

In my case, I have installed the nginx service to check its status.

sudo apt-get install nginx -y

service nginx status ;; manually check whether its in running state or not

Explanation

- **SERVICE:** The name of the service to check.
- **systemctl is-active --quiet \$SERVICE:** Checks if the service is running.
- **echo "\$SERVICE is running":** Prints a message if the service is running.
- **systemctl start \$SERVICE:** Starts the service if it is not running.

```
#!/bin/bash

SERVICE="nginx"    ## put any service name that you want to check

# Check if the service is running, if not, start it

if systemctl is-active --quiet $SERVICE; then

echo "$SERVICE is running"
[]
else

echo "$SERVICE is not running"

systemctl start $SERVICE

fi
~
~
~
~
~
~
```

Give the executable permission to the script file and execute it manually, or you can set the cron job as well.

```
ubuntu@ip-172-31-19-13:~$ ./service-health-check.sh
nginx is running
ubuntu@ip-172-31-19-13:~$ []
```