—— —— —— —— —— —— —— —— —— —— —— —— —— —— —— ——

**Scenario**

—— —— —— —— —— —— —— —— —— —— —— —— —— —— —— ——

You are working on a critical project inside a dev/ directory located in your /home/username/Learning_directory/. These files must be backed up daily to prevent data loss. You want an automated solution that:

- Compresses the dev/ directory into a .tar.gz file.
- Stores the backup in ~/Learning_directory/.
- Optionally sends you an email notification if the backup is successfully created.
- Runs automatically every morning, using cron.

## 1. Prepare Your Environment

### Verify your current working directory

pwd

/home/tiagopaquete/Learning_directory

### Create and organize a test directory (optional if it doesn't exist)

mkdir -p ~/Learning_directory/dev

drwxr-xr-x. 2 root root  6 Apr  4 20:17 **dev**

touch ~/Learning_directory/dev/testfile1
touch ~/Learning_directory/dev/testfile2

-rw-r--r--. 1 root root  0 Apr  4 20:19 testfile1
-rw-r--r--. 1 root root  0 Apr  4 20:19 testfile2

(This simulates important files you don't want to lose.)

— — — — — — — — — — — — — — — — — — — — — — — —

## 2. Create the Backup Script

— — — — — — — — — — — — — — — — — — — — — — — —

**Create a file named mybackup**

```
nano mybackup
```

Or use any other text editor (e.g., vi).

**Add the script content**

```
#!/bin/bash

# myBackup: backup utility for dev directory

BACKUP_PATH="/home/username/Learning_directory/dev/"
HOME_PATH="/home/username/Learning_directory/"

DATE=`date +%d%m%Y`
BACKUP="backup_"
EXT=".tar"

FILE_NAME=$HOME_PATH$BACKUP$DATE$EXT

# Create a compressed tar archive of BACKUP_PATH and place it in FILE_NAME
tar cfz $FILE_NAME $BACKUP_PATH

# Check if backup file was created successfully
if test -f "$FILE_NAME"; then
   echo "Here's your daily backup@" | mail -A $FILE_NAME -s "Today's Backup"
example@gmail.com
else
   echo $DATE "There was a problem creating the backup file." >> $HOME_PATH/
error.log
fi
```

Replace "example@gmail.com" with your real email address and “username” with your user name

## 3. Adjust Script Permissions

### Make the script executable

```
-rw-r--r--. 1 root root 633 Apr  4 20:44 mybackup
```

```
chmod u=rwx mybackup
```

```
-rwxr--r--. 1 root root 633 Apr  4 20:44 mybackup
```

### Fine-tune permissions if desired

```
chmod go=rx mybackup    # Group + Others can read, execute
```

Adjust as needed. For a private script, you might limit read/execute to yourself only.

— — — — — — — — — — — — — — — — — — — — — — — —

## 4. Test the Backup Script

— — — — — — — — — — — — — — — — — — — — — — — —

**Run the script manually**

./mybackup

**Check for the backup file**

ls -l ~/Learning_directory

-rw-r--r--. 1 root root  574 Apr  4 20:49 **backup_04042025.tar**

**Verify optional email or error logs**

If successful, check your inbox.
If not, open error.log in ~/Learning_directory/ to see if there's a recorded problem.

—— —— —— —— —— —— —— —— —— —— —— —— —— —— —— ——

## 5. Automate with cron

—— —— —— —— —— —— —— —— —— —— —— —— —— —— —— ——

**Open the crontab editor**

```
crontab -e
```

**Schedule the script (example: every day at 2:00 AM)**

```
0 2 * * * /home/tiagopaquete/Learning_directory/mybackup
```

The syntax: Minute Hour Day-of-Month Month Day-of-Week command.

`0 2 * * *` → At 2:00 AM every day.

**Save and exit**

Your backup script is now set to run automatically each day at 2 AM.

## 6. Validation

**View the crontab**

crontab -l

```
$ crontab -l
# ┌───────────────────── Minute (0 - 59)
# │ ┌─────────────────── Hour (0 - 23)
# │ │ ┌───────────────── Day of the Month (1 - 31)
# │ │ │ ┌─────────────── Month (1 - 12)
# │ │ │ │ ┌───────────── Day of the Week (0 - 7) (Sunday is both 0 and 7)
# │ │ │ │ │
# * * * * * command_to_execute

# The backup script is now set to run automatically each day at 2 AM.
0 2 * * * /home/tiagopaquete/Learning_directory/mybackup

# Notes:
# - Use '*' as a wildcard for 'every' possible value.
# - The cron daemon uses the system's timezone and time settings.
# - All output (including errors) will be emailed to the user unless redirected.

#redirect the output to a file
0 2 * * * /home/tiagopaquete/Learning_directory/mybackup >> /var/log/cron_backup.log 2>&1

# To learn more:
# man 5 crontab   → explains field syntax
# man 8 cron      → explains the cron service
```

## 7. Optional Maintenance Commands

**Remove or change the crontab**

```
crontab -r   # Removes your entire crontab (use with caution)
```

**Edit again if you want to change frequency**

```
crontab -e
```

—— —— —— —— —— —— —— —— —— —— —— —— —— —— —— ——

**Script Breakdown**

—— —— —— —— —— —— —— —— —— —— —— —— —— —— —— ——

#!/bin/bash

**Shebang line**: Tells the system to execute this script using the Bash shell.

It must be the **first line** in the script.

#!/bin/bash is the full path to the Bash interpreter on most Linux systems (including CentOS 9).

# myBackup: backup utility for dev directory

A **comment line**. It's ignored during execution but useful for documentation. Describes the purpose of the script for future readers or yourself.

—— —— —— —— —— —— —— —— —— —— —— —— —— —— —— ——

**Define the Paths**

—— —— —— —— —— —— —— —— —— —— —— —— —— —— —— ——

BACKUP_PATH="/home/tiagopaquete/Learning_directory/dev/"
HOME_PATH="/home/tiagopaquete/Learning_directory/"

**BACKUP_PATH**: The **source directory** to back up.
**HOME_PATH**: The **destination directory** where the backup will be saved.

These are **hard-coded** absolute paths. You can make them dynamic later (e.g., based on $HOME).

—— —— —— —— —— —— —— —— —— —— —— —— —— —— —— ——

**Generate Dynamic Filename**

—— —— —— —— —— —— —— —— —— —— —— —— —— —— —— ——

DATE=`date +%d%m%Y`

Runs the date command with the format daymonthyear.

Example: On April 4, 2025 → 04042025.

The backticks `command` are command substitution, meaning the output of the command becomes the value of the variable.

**Tip:** You can also use DATE=$(date +%d%m%Y) — preferred in modern Bash scripts.

```
BACKUP="backup_"
EXT=".tar"
```

Defines the filename parts:

backup_: prefix
.tar: extension

```
FILE_NAME=$HOME_PATH$BACKUP$DATE$EXT
```

Combines everything to form the full **output filename path**.

Example: /home/tiagopaquete/Learning_directory/backup_04042025.tar

——————————————————————————————————————

**Create the Backup Archive**

——————————————————————————————————————

```
tar cfz $FILE_NAME $BACKUP_PATH
```

**tar** = archive tool (tape archive).
**c** = create a new archive
**f** = use the following filename
**z** = compress with gzip

$FILE_NAME is the name of the resulting archive
$BACKUP_PATH is the folder to include in the archive

This creates a compressed .tar.gz file of the dev/ directory.

——————————————————————————————————————

**Validate and Send Notification**

——————————————————————————————————————

```
if test -f "$FILE_NAME"; then
```

test -f checks if a **regular file** exists with the name $FILE_NAME.

The script now performs different actions **based on whether the backup was successful**.

— — — — — — — — — — — — — — — — — —

**If Backup Succeeded**

— — — — — — — — — — — — — — — — — —

```
    echo "Here's your daily backup@" | mail -A $FILE_NAME -s "Today's Backup" example@gmail.com
```

Sends an **email with the backup file as an attachment**.

**mail** is a command-line mail client (make sure it's installed, e.g., mailx).
**-A** attaches the file (if your version of mail supports it).
**-s** specifies the subject line.
example@gmail.com is the recipient (change this to your real email).

🔔 You'll get an email when the backup is successful, which is helpful for monitoring automated tasks.

— — — — — — — — — — — — — — — — — —

**If Backup Failed**

— — — — — — — — — — — — — — — — — —

```
else
    echo $DATE "There was a problem creating the backup file." >> $HOME_PATH/error.log
fi
```

else: fallback action if the backup file **was not created**.

Appends a line to error.log including the date and a message.

>> means "append to file" — it won't overwrite the existing file.