# CRON JOBS

A Cron Job is a scheduled task on a Unix-like operating system. It's named after "cron," a time-based job scheduler in Unix-like systems (including Linux). These jobs can be anything from simple scripts to complex system maintenance tasks, and they run automatically at predetermined times or intervals.

## HOW DO CRON JOBS WORK?

Cron Jobs are managed by the Cron Daemon, a background process that continuously checks a configuration file called the "crontab" for scheduled tasks. Each user on the system can have their own crontab, which lists the tasks they want to run. The crontab contains lines with timing information and the command to execute. A breakdown:

- The Cron Daemon runs as a background process on Unix-like operating systems.
- It periodically (usually every minute) checks the crontab files for scheduled tasks.
- For each entry in the crontab files, it compares the current time to the schedule specified in the entry.
- If the current time matches the schedule of a task, the Cron Daemon triggers the execution of that task.
- The task/command specified in the crontab entry is executed, and any output or errors are typically logged or emailed to the user who scheduled the task.
- The Cron Daemon continues to periodically check for other tasks that may need to run.

Basic format of a Crontab line is * * * * * command_*to*_execute, which simply means:

- The five asterisks represent the schedule: minute (0–59), hour (0–23), day of the month (1–31), month (1–12), and day of the week (0–6, where 0 is Sunday).
- The command_to_execute is the actual command or script that the cron job will run.

For example, to have a cron job run a task every midnight, the line will look like this: 0 0 * * * /home/user/log_cleanup.sh.

COMMON USE CASES

- Data Backups**:** Regularly back up databases or important files.
- Maintenance: Perform system maintenance tasks, like cleaning up temporary files.
- Reports: Generate and send reports at specific times.
- Scheduled Jobs: Execute scripts for updating cached data, sending notifications, or processing data.
- Periodic Tasks**:** Check for software updates or security scans.

**GUIDE TO A SCHEDULING A DAILY LOG CLEANUP WITH A CRON JOB..**

- Open a terminal on your Linux system

- Create a Bash script: vim  /home/$USER/cleanup_logs.sh
- Then add the following script:

- (#!/bin/bash
-
- # Define the log directory
- LOG_DIR="/var/log/myapp”
- #check if the log directory exists
- If [ -d “$LOG_DIR” ];
- then
- # Delete log files older than 7 days
- find “$LOG_DIR” -type f -name "*.log" -mtime +7 -delete
-
- # Optional: Log the cleanup action to a new file
- echo "Log cleanup completed on $(date)" >> /var/log/cleanup_logs.log)
- else
-  echo “Log directory $LOG_DIR does not exist” >> /var/log/cleanup_logs.log
- Fi

- Save and exit.
- Make the script executable: chmod +x /home/$USER/cleanup_logs.sh
- Test the script manually: /home/$USER/cleanup_logs.sh.. check if it deletes the intended files and creates /var/log/cleanup.log, if you don’t have sample logs , you can create a dummy file using:

- mkdir -p /var/log/myapp
- touch /var/log/myapp/test1.log
- touch -d "10 days ago" /var/log/myapp/test2.log

Schedule the cron Job

- open the crontab file: crontab –e
- add the cron job by adding this script to run at midnight: 0 0 * * * /home/$USER/cleanup_logs.sh
- save and exit
- verify the cronjob: crontab –l

Ensure permissions: The script and log directory must be accessible to the user running the cron job (your user account, unless specified otherwise). If you’re cleaning system logs (e.g., /var/log), you may need **root privileges**. In that case:

- move the script to a system directory: sudo mv /home/$USER/cleanup_logs.sh /usr/local/bin/cleanup_logs.sh
- edit the root crontab: sudo crontab –e and add: 0 0 * * * /usr/local/bin/cleanup_logs.sh

MONITOR AND TEST

- check the cleanup log: cat /var/log/cleanup.log
- check cron logs if needed..use: grep CRON /var/log/syslog.

**Explanation**:

- #!/bin/bash: Specifies this is a Bash script.
- LOG_DIR: The directory containing logs. Replace /var/log/myapp with your target directory (e.g., /var/log/apache2). Avoid using /var/log directly, as it may contain critical system logs.
- find: Searches for files:
    - -type f: Only regular files (not directories).
    - -name "*.log": Matches files ending in .log.
    - -mtime +7: Files older than 7 days.
    - -delete: Deletes matching files.
- The echo command logs the cleanup time to a file for tracking.
- The script logs its actions to /var/log/cleanu_logs.log for auditing
- The if [ -d “$LOG_DIR” ] check ensures the script doesn’t fail if the directory doesn’t exist.