Bash scripting for automation

Bash scripting is writing a series of commands in a text file (called a script) using the Bash shell. Bash stands for Bourne Again SHell, and it's the default shell in most Linux distributions, including Ubuntu.

Bash scripts help automate tasks like backups, installations, system monitoring, or even managing services.

1. Create a Script File:

Open a terminal and type:

hello.sh - Creates a file named "hello.sh"

2. Add Code:

Type these lines into the file:

#! /bin/bash - Tell the computer: "Use Bash to run this!"

echo "Hello, World!" - Prints "Hello, World!" on the screen

3. Make It Executable:

Save the file, then type:

chmod +x hello.sh - Gives permission to run the script

4. Run It: **./hello.sh**

```
#!/bin/bash
# Infinite for loop with break
i=0
for (( ; ; ))
do
        echo "Iteration: ${i}"
        (( i++ ))
        if [[ i -gt 10 ]]
        then
                break;
        fi
done
echo "Done!"
```

Automate Backups

Tools Used:

- Bash script
- **cron** for scheduling
- **tar** or **rsync** for backup

Example Workflow:

- Backup your /home/user/Documents to /home/user/Backups daily.

Script: backup.sh

Logging

Tools:

- Script output redirection (>> log.txt)
- System logs (/var/log/syslog, /var/log/auth.log)
- logger command (optional)

Monitoring

Tools:

- logwatch: Summarizes logs
- cron reports (via email)
- monit: Lightweight monitoring tool
- systemd for service status
- Optional: Nagios, Zabbix, or Prometheus for advanced setups

Backup	-	tar, rsync, bash script
Scheduling	-	cron
Logging	-	logger, log files
Monitoring	-	monit, systemd, logwatch
Alerts	-	mailx, or monitoring tools

Using crontab for shaduling

To use crontab for scheduling tasks in Ubuntu, you'll first need to ensure the cron service is installed and running. Then, you can edit your crontab file to add scheduled tasks. This file is a text file where you define the schedule and command for each task.

```
# ┌──────────── minute (0 - 59)
# │ ┌──────────── hour (0 - 23)
# │ │ ┌──────────── day of the month (1 - 31)
# │ │ │ ┌──────────── month (1 - 12)
# │ │ │ │ ┌──────────── day of the week (0 - 6) (Sunday to Saturday;
# │ │ │ │ │                                    7 is also Sunday on some systems)
# │ │ │ │ │
# │ │ │ │ │
# * * * * * command to execute
```

Installing Cron

Almost every Linux distribution has some form of cron installed by default. However, if you're using an Ubuntu machine on which cron isn't installed, you can install it using APT.

sudo apt install cron

Understanding How Cron Works

Cron jobs are recorded and managed in a special file known as a crontab. Each user profile on the system can have their own crontab where they can schedule jobs, which is stored under**/var/spool/cron/crontabs/.**

To schedule a job, open up your crontab for editing and add a task written in the form of a cron expression. The syntax for cron expressions can be broken down into two elements: the schedule and the command to run.

MIN (Minute)	Specifies the minute when the command will run	It ranges from 0 to 59.
HOUR	Denotes the hour of the day when the command is scheduled to execute.	It spans from 0 to 23.
DOM (Day of Month)	Specifies the day of the month for the task.	It ranges from 1 to 31.
MON (Month)	Indicates the month during which the command will be executed.	It varies from 1 to 12.
DOW (Day of Week)	Specifies the day of the week for the task.	It is represented by numbers from 0 to 7, where both 0 and 7 correspond to Sunday.
CMD (Command)	Represents the actual	

View your crontab:

crontab -l

Edit your crontab:

crontab -e

Remove your crontab:

crontab -r