# Task 1: System Monitoring Setup

Objective: Configure a monitoring system to ensure the development environment’s health, performance, and capacity planning.

**Scenario:**

* The development server is reporting intermittent performance issues.
* New developers need visibility into system resource usage for their tasks.
* System metrics must be consistently tracked for effective capacity planning.

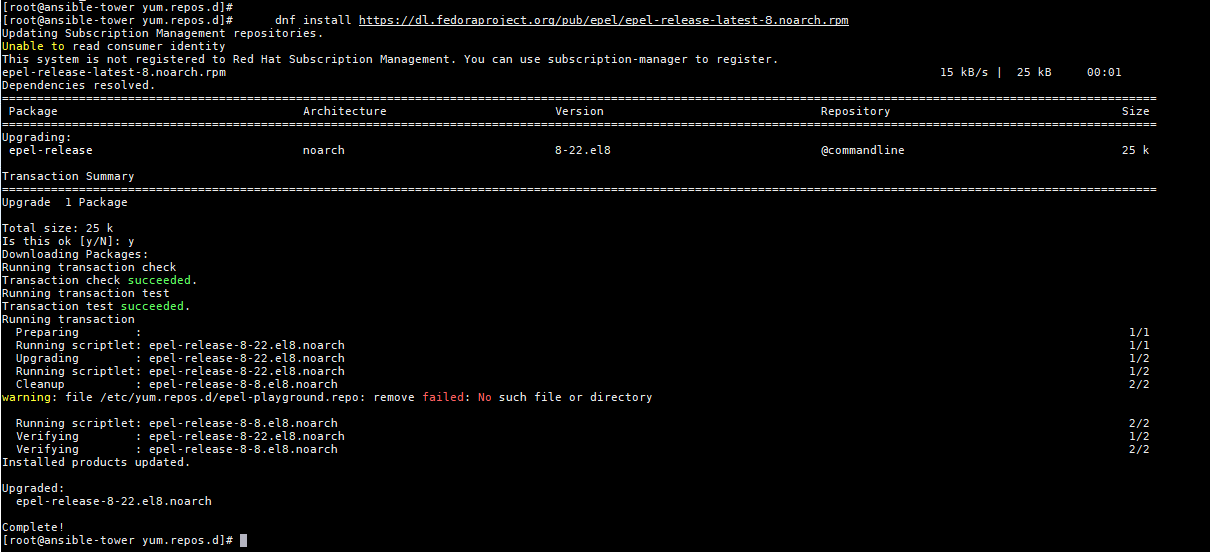
**Requirements:**

1. Install and configure monitoring tools (htop or nmon) to monitor CPU, memory, and process usage.
2. Set up disk usage monitoring to track storage availability using df and du.
3. Implement process monitoring to identify resource-intensive applications.
4. Create a basic reporting structure (e.g., save outputs to a log file for review).

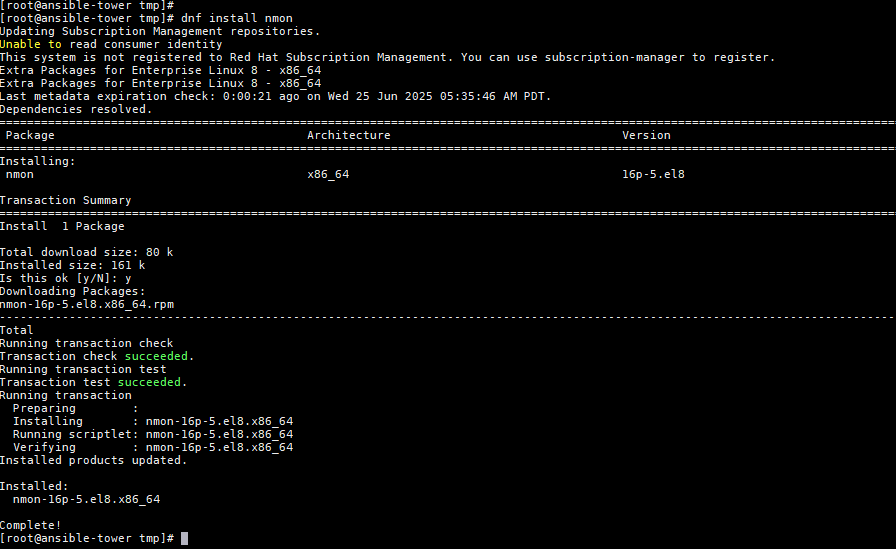
* Install nmon pkg on RHEL:

1. Enable EPEL repository:

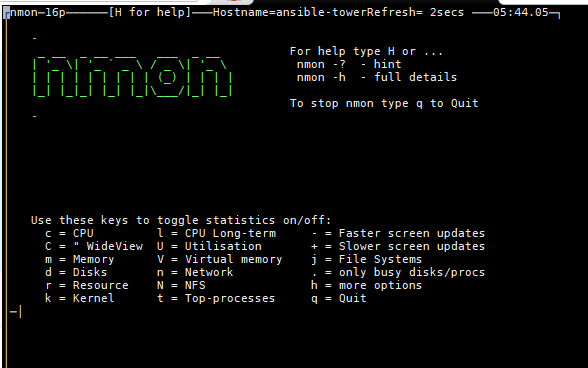
dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm

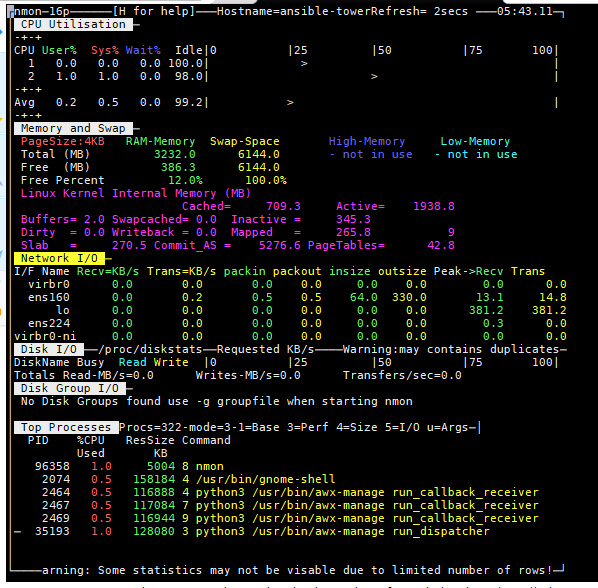


1. Install nmon package



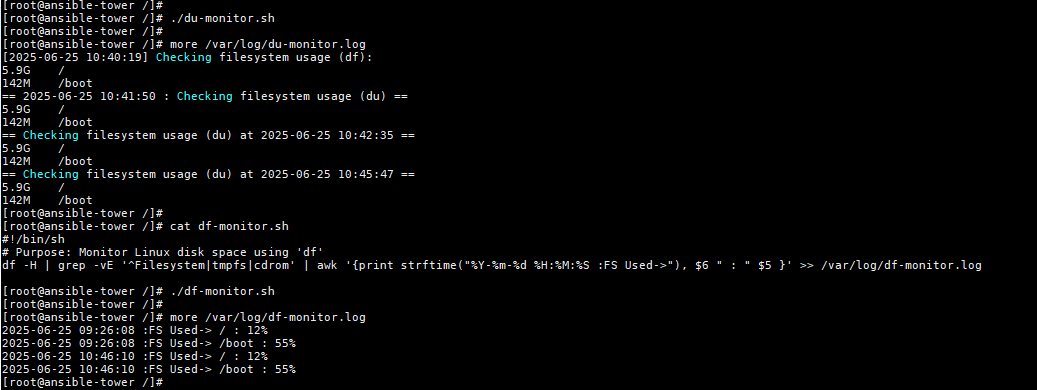
1. Performance monitoring using ‘NMON’



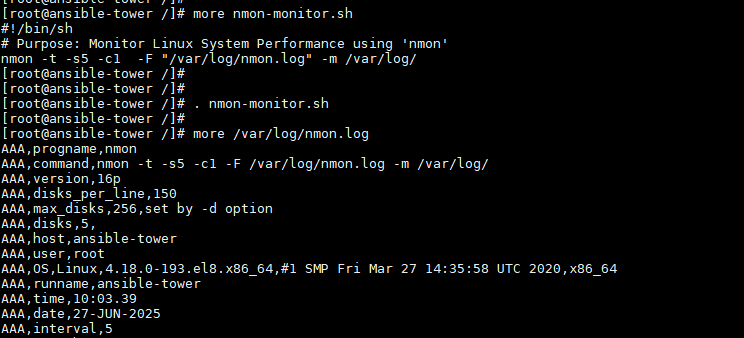


1. Test run of scripts:

* ‘du’ & ‘df’ commands



* ‘nmon’ script



# Task 2: User Management and Access Control

Objective: Set up user accounts and configure secure access controls for the new developers.

Scenario:

* Two new developers, Sarah and Mike, require system access.
* Each developer needs an isolated working directory to maintain security and confidentiality.
* Security policies must ensure proper password management and access restrictions.

Requirements:

1. Create user accounts for Sarah and Mike with secure passwords.
2. Set up dedicated directories:

Sarah: /home/Sarah/workspace

Mike: /home/mike/workspace

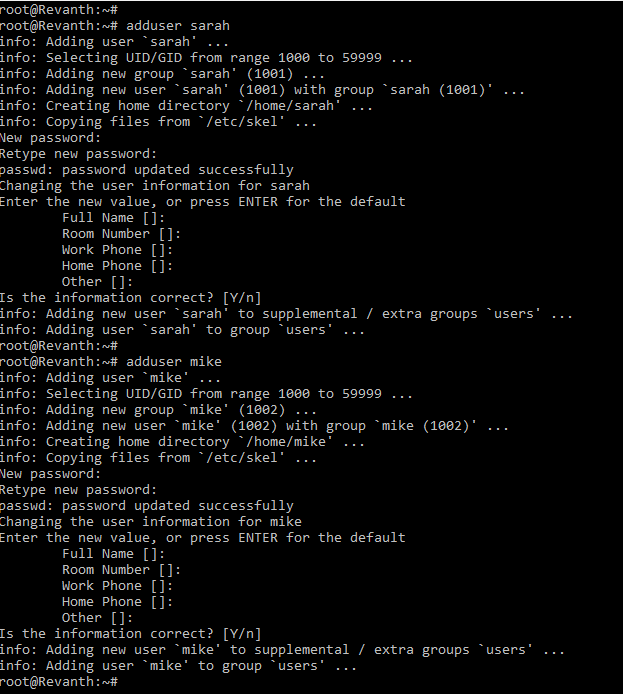
1. Ensure only the respective users can access their directories using appropriate permissions.
2. Implement a password policy to enforce expiration and complexity (e.g., passwords expire every 30 days).

**Solution:**

1. **Create User Accounts with Secure Passwords**

# adduser Sarah

# adduser mike



1. **Set Up Dedicated Workspace Directories**

mkdir -p /home/Sarah/workspace

mkdir -p /home/mike/workspace

1. **Assign ownership of each directory to the respective user:**

chown Sarah:Sarah /home/Sarah/workspace

chown mike:mike /home/mike/workspace

1. **Restrict Directory Access**

Set permissions so only the respective user can access their workspace:

# chmod 700 /home/Sarah/workspace

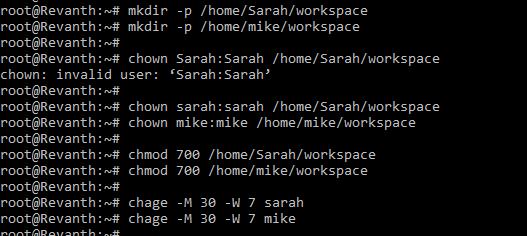
# chmod 700 /home/mike/workspace

1. **Enforce Password Expiration Policy**

Set passwords to **expire every 30 days** and warn users 7 days before expiration:

# chage -M 30 -W 7 sarah

# chage -M 30 -W 7 mike



1. **Enforce Password Complexity**
2. Install the PAM password quality module if not already present:

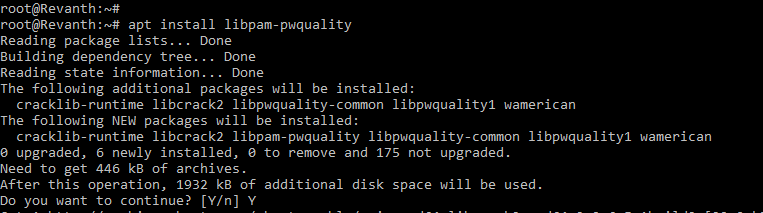
# apt install libpam-pwquality

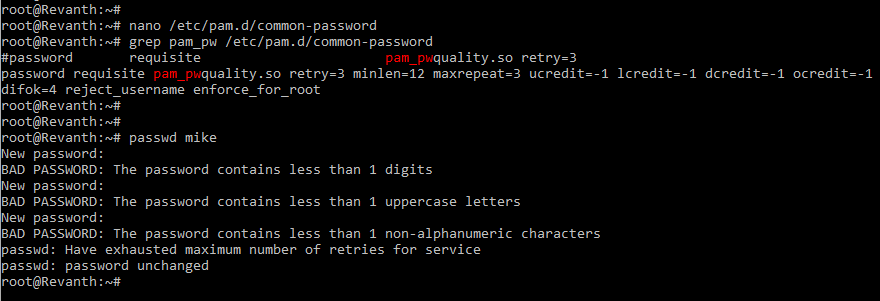
1. Edit /etc/pam.d/common-password to enforce complexity (at least 12 characters, upper/lowercase, digit, special character):

Change like -> “password requisite pam\_pwquality.so retry=3”

To

“password requisite pam\_pwquality.so retry=3 minlen=12 maxrepeat=3 ucredit=-1 lcredit=-1 dcredit=-1 ocredit=-1 difok=4 reject\_username enforce\_for\_root”





# Task 3: Backup Configuration for Web Servers

**Objective:** Configure automated backups for Sarah’s Apache server and Mike’s Nginx server to ensure data integrity and recovery.

**Scenario:**

* Sarah is responsible for managing an Apache web server.
* Mike is responsible for managing a Nginx web server.
* Both servers require regular backups to a secure location for disaster recovery.

**Requirements:**

1. Sarah and Mike need to automate backups for their respective web server configurations and document roots:

* Sarah: Backup the Apache configuration (/etc/httpd/) and document root (/var/www/html/).
* Mike: Backup the Nginx configuration (/etc/nginx/) and document root (/usr/share/nginx/html/).

1. Schedule the backups to run every Tuesday at 12:00 AM using cron jobs.
2. Save the backups as compressed files in /backups/ with filenames including the server name and date (e.g., apache\_backup\_YYYY-MM-DD.tar.gz).
3. Verify the backup integrity after each run by listing the contents of the compressed file.

**Expected Output:**

* Cron job configurations for Sarah and Mike.
* Backup files are created in the /backups/ directory.
* A verification log showing the backup integrity.

**Deliverables:**

* Screenshots or terminal outputs showing successful completion of each task.
* A report summarizing the implementation steps and any challenges encountered.
* Backup files or verification logs for Task 3.

**Solution:**

1. **Create the /backups/ directory** (if it does not exist):

mkdir -p /backups/

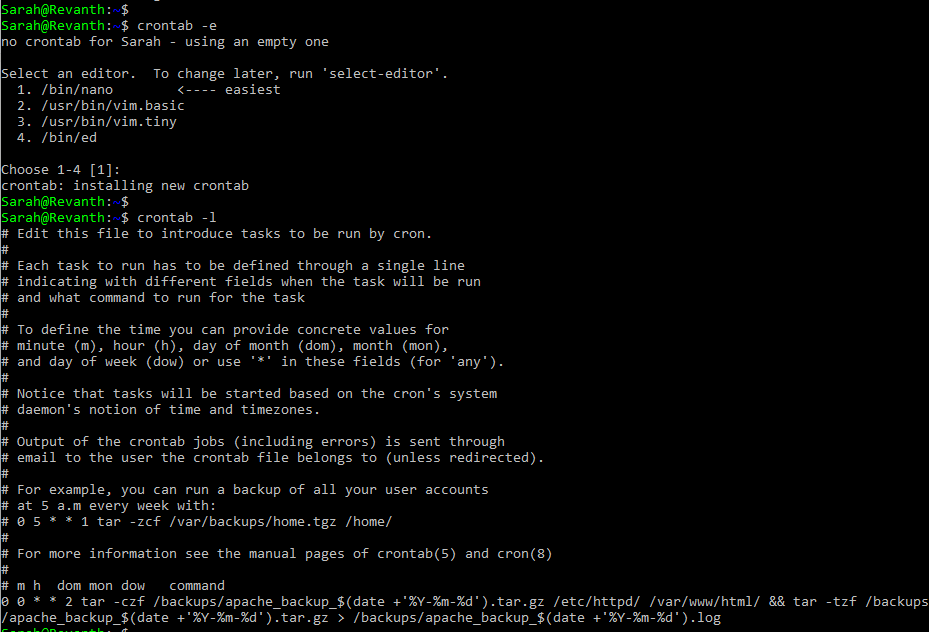
chown sarah:sarah /backups/ # for Sarah's server

chown mike:mike /backups/ # for Mike's server



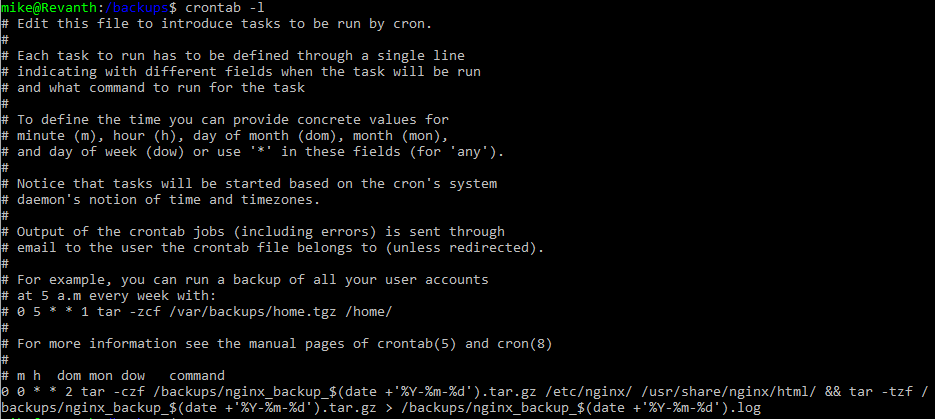
1. Add Cron Job Configurations for Automated Backups to each user's crontab: ‘crontab -e’
2. **Sarah (Apache Server):**

0 0 \* \* 2 tar -czf /backups/apache\_backup\_$(date +'%Y-%m-%d').tar.gz /etc/httpd/ /var/www/html/ && tar -tzf /backups/apache\_backup\_$(date +'%Y-%m-%d').tar.gz > /backups/apache\_backup\_$(date +'%Y-%m-%d').log



1. **Mike (Nginx Server):**

0 0 \* \* 2 tar -czf /backups/nginx\_backup\_$(date +'%Y-%m-%d').tar.gz /etc/nginx/ /usr/share/nginx/html/ && tar -tzf /backups/nginx\_backup\_$(date +'%Y-%m-%d').tar.gz > /backups/nginx\_backup\_$(date +'%Y-%m-%d').log



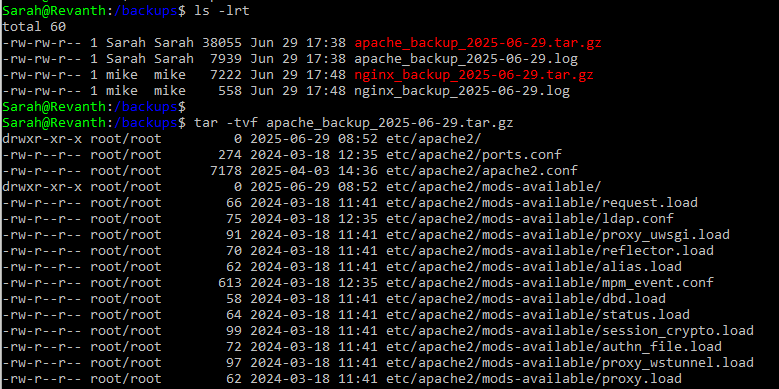
* **Runs every Tuesday at 12:00 AM**
* **Creates a compressed backup** of Apache config and document root
* **Verifies backup integrity** by listing archive contents into a log file

**Backup File Creation**

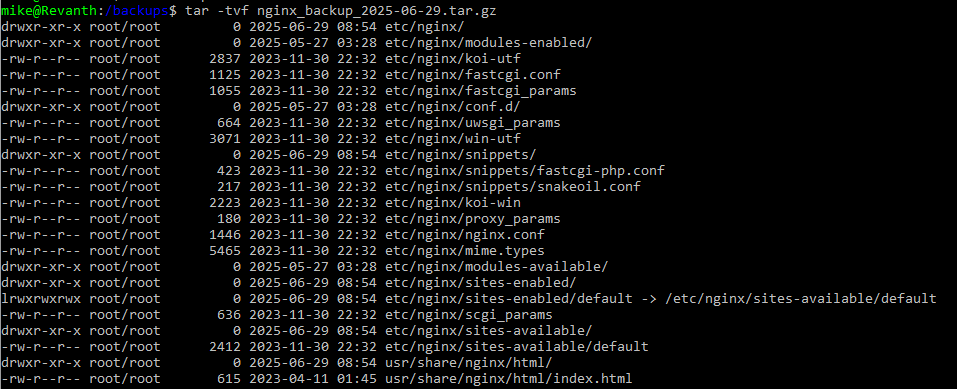
* **Backup files** are saved in /backups/ as:
  + apache\_backup\_YYYY-MM-DD.tar.gz
  + nginx\_backup\_YYYY-MM-DD.tar.gz
* **Verification logs** are saved as:
  + apache\_backup\_YYYY-MM-DD.log
  + nginx\_backup\_YYYY-MM-DD.log

**Verification of Backup Files:**

* **Sarah**

****

* **Mike**

****