Shell Scripting

A diagram of a shell

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A close-up of a computer script

Description automatically generated

# Executing / running a Program in Shell:

A screen shot of a computer code

Description automatically generated

What is logrotate in linux

**logrotate** is a utility in Linux that helps manage and rotate log files to prevent them from growing too large and consuming excessive disk space. It is commonly used on systems that generate a large number of log files, such as web servers, databases, or application servers.

The primary purpose of log rotation is to:

1. **Manage Log Size**: Logs can grow rapidly, especially on busy servers, and may occupy a significant amount of disk space. Log rotation ensures that logs are periodically truncated or archived, preventing them from filling up the entire disk.
2. **Maintain Log History**: Rotated logs are typically archived and saved with a timestamp or sequence number. This allows system administrators to maintain a history of log data, which can be useful for troubleshooting, auditing, and analysis.
3. **Improve Performance**: Managing smaller log files is more efficient and faster for processes that need to read and write log data.

The **logrotate** utility is usually configured through a configuration file located in the **/etc/logrotate.d/** directory or included in the main **/etc/logrotate.conf** file. The configuration file specifies which log files should be rotated, the frequency of rotation, the compression options, and how many archived log files to retain.

The log rotation process is often executed as a cron job, scheduled to run at specific intervals (e.g., daily, weekly, or monthly). When **logrotate** runs, it checks the criteria specified in the configuration files, rotates the log files that meet those criteria, and may optionally compress or archive the rotated logs.

To manually trigger **logrotate**, you can use the command **logrotate -f /etc/logrotate.conf** or **logrotate -f /etc/logrotate.d/<config\_file>**.

Always make sure to understand the log rotation settings and the potential impact before configuring or modifying **logrotate** for your system. Improper configuration could result in the loss of important log data or cause other issues.

When a log file is consuming more Space in etc

If the **sysco.log** file in the **/etc/logs** directory is consuming more space and you want to manage its size with log rotation, you can follow the same logrotate process as described before. Here's a specific logrotate script for the **sysco.log** file:

1. Create a new logrotate configuration file in the **/etc/logrotate.d/** directory for the **sysco.log**:

bashCopy code

sudo nano /etc/logrotate.d/sysco\_log

1. Add the following content to the file:

plaintextCopy code

/etc/logs/sysco.log { su root root daily rotate 7 missingok notifempty compress delaycompress create 640 root root }

The options in this configuration file are similar to the previous one, with the main difference being that it targets the specific **sysco.log** file.

1. Save the file (Ctrl + O in nano) and exit (Ctrl + X in nano).
2. To manually test the log rotation for **sysco.log**, you can use the following command:

bashCopy code

sudo logrotate -v -f /etc/logrotate.d/sysco\_log

The **-v** option adds verbosity to the output, and the **-f** option forces log rotation even if it's not scheduled yet.

This logrotate script will help manage the size of the **sysco.log** file by keeping up to 7 rotated log files. Adjust the **rotate** value as per your requirements to retain a suitable number of log file archives. Remember to be cautious when rotating logs in critical directories like **/etc**, as some logs may be essential for the system's operation. Always verify the configuration before implementing it in a production environment.



tell me using cron job

ChatGPT

Sure! To automate the log rotation using a cron job, you'll need to create a cron job that runs the logrotate command at a specific interval. Here's how you can set up a cron job for log rotation:

1. Open the crontab for editing:

bashCopy code

crontab -e

1. If it's the first time you're editing the crontab, you may be asked to choose an editor. Select your preferred editor (e.g., nano, vim).
2. Add a new line at the end of the crontab file to schedule the log rotation:

plaintextCopy code

0 0 \* \* \* /usr/sbin/logrotate -f /etc/logrotate.d/sysco\_log

In this example:

* **0 0 \* \* \*** specifies the schedule for the cron job. The five asterisks represent the minute, hour, day of the month, month, and day of the week, respectively. In this case, it runs the logrotate command every day at midnight (00:00).
* **/usr/sbin/logrotate** is the path to the logrotate binary. Use **which logrotate** command to find the correct path if it's different on your system.
* **-f /etc/logrotate.d/sysco\_log** specifies the logrotate configuration file to use for log rotation. Make sure to adjust the path and filename based on your logrotate configuration.

1. Save the crontab file and exit the editor.

The cron job will now run the logrotate command specified in the crontab at the scheduled time. This will automatically rotate the **sysco.log** file and keep up to 7 rotated log files as per the configuration provided in **/etc/logrotate.d/sysco\_log**.

Remember to exercise caution when managing logs in critical directories like **/etc**. Make sure you have tested the log rotation and verified the configuration before setting it up as a cron job in a production environment.



The logrotate which you provided doesnt have a shebang command in it ? So its not required in logroate ?

ChatGPT

You are correct. In logrotate configuration files, a shebang command (e.g., **#!/bin/bash**, **#!/bin/sh**) is not required because these files are not executed as scripts but are used as configuration files for the **logrotate** utility.

The **logrotate** utility itself is a separate program that runs outside the context of a shell script. It reads the configuration files in **/etc/logrotate.d/** and **/etc/logrotate.conf** to determine how to rotate and manage log files.

The lack of a shebang is intentional, as **logrotate** does not interpret the configuration files as shell scripts. Instead, it reads the instructions and settings specified in these files to perform log rotation tasks based on the defined criteria.

So, when you create a logrotate configuration file in **/etc/logrotate.d/** (e.g., **/etc/logrotate.d/sysco\_log** as shown earlier), you don't need a shebang line in it. The file should contain only the relevant log rotation directives and settings for the specific log file you want to manage. **logrotate** will take care of reading and executing these directives accordingly.

Top of Form

In my linux system , the Jboss service is getting stopped always . tell me a shell script , so that it should restart the service after 5 mins once after the service goes down

Create a shell script, let's call it **restart\_jboss.sh**, with the following content:

#!/bin/bash

# Function to check if JBoss is running

is\_jboss\_running() {

if systemctl is-active jboss.service >/dev/null 2>&1; then

return 0 # JBoss is running

else

return 1 # JBoss is not running

fi

}

# Restart JBoss service and wait for 5 minutes

restart\_jboss() {

systemctl restart jboss.service

echo "JBoss restarted at $(date)"

sleep 300 # Wait for 5 minutes (300 seconds) before checking again

}

# Main loop to monitor and restart JBoss

while true; do

if is\_jboss\_running; then

echo "JBoss is running. Waiting..."

else

restart\_jboss

fi

done

1. Save the file and make it executable:

bashCopy code

chmod +x restart\_jboss.sh

1. Next, you can set up a cron job to run the script periodically. Open the crontab for editing:

bashCopy code

crontab -e

1. Add a new line at the end of the crontab to schedule the script to run every minute:

plaintextCopy code

\* \* \* \* \* /path/to/restart\_jboss.sh >/dev/null 2>&1

Replace **/path/to/restart\_jboss.sh** with the actual path to your **restart\_jboss.sh** script.

This cron job will execute the **restart\_jboss.sh** script every minute. The script will check if the JBoss service is running, and if it's not, it will restart the service and wait for 5 minutes before checking again. By running the script every minute, it ensures that if the JBoss service goes down, it will be restarted within 5 minutes.

Please keep in mind that running the script every minute might generate a lot of output in the system logs (e.g., syslog or cron logs). If you prefer a less frequent check, you can adjust the cron schedule accordingly (e.g., every 5 minutes, every 10 minutes, etc.).