SCRIPTING

Scripting refers to writing small programs (called scripts) that automate tasks. Scripting languages are usually interpreted, not compiled, which makes them fast to write and execute. Popular scripting languages include:

- Bash (Linux/macOS shell)
- PowerShell (Windows automation)
- Python
- JavaScript

Scripting is widely used by system administrators, developers, DevOps engineers, data analysts, and cloud engineers.

Why Use Scripting?

- **Automation:** Replace repetitive manual tasks
- **Speed:** Quick to write and execute
- **Portability:** Scripts can be reused and shared easily
- **Flexibility:** Easily modified for different environments

Scripting in AWS: 5 Real-World Scenarios Using Internal and External Scripts

This documentation provides practical examples of scripting within AWS environments using both internal EC2-based scripts and external automation (such as AWS Lambda). Each scenario is designed for real-world applications in operations, automation, and security.

Internal Scripts (Built-in commands)

Steps:

Create script file: nano myscript.sh

Add shebang and commands:

#!/bin/bash

ls -la

df -h

Make executable: chmod +x myscript.sh

Execute: ./myscript.sh **or schedule via a cron job.**

In some scenarios like stopping the instance via a script created internally, you will need to attach an IAM role to the EC2 instance with permissions to carry out the specific operation.

External Scripts (Lambda Function)

Steps:

Create Lambda function in AWS Console

Write Lambda code (choose runtime for specific language):

Set IAM permissions for Lambda

Trigger Lambda from EC2:

aws lambda invoke --function-name MyFunction response.json

Or use EventBridge/CloudWatch to trigger automatically.

Scenario 1: Log Backup (Internal Bash Script)

Objective

Automatically back up system logs from an EC2 instance to a designated log file every day.

Script: /home/ec2-user/backup_logs.sh (refer to repo)

This is an internal bash script which creates a compressed backup of system log files.

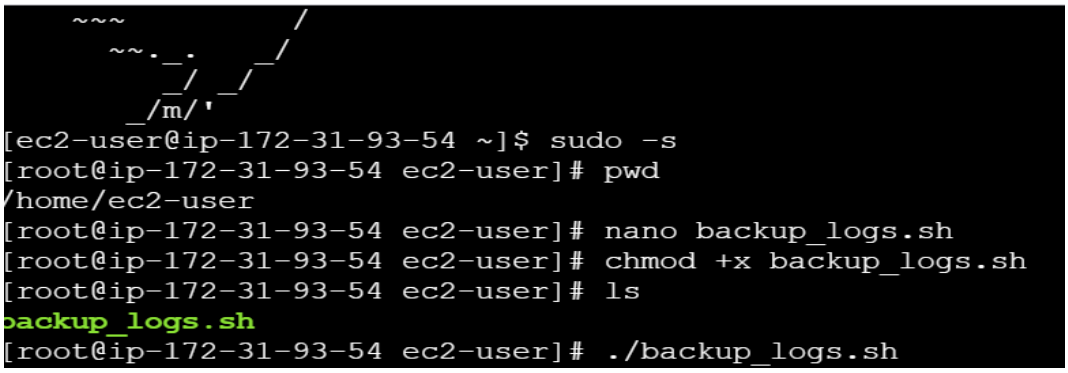
Cron Setup

0 1 * * * /home/ec2-user/backup_logs.sh (cron job to run daily at 1 AM)

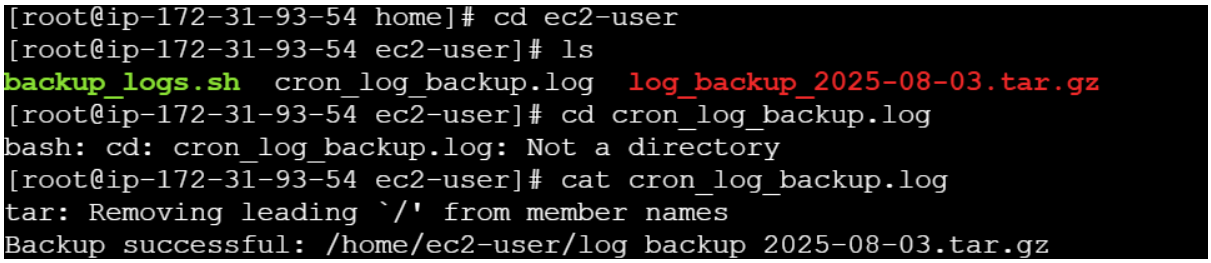
Purpose

- The script is designed to run automatically (likely via cron) and maintains a log of backup operations in /home/ec2-user/cron_log_backup.log.
- Ensures logs are compressed and stored for audit/compliance.

Demo: The pictures below shows how file was created and executed.



Check if log file is created and shows that backup was successful



Unzip the backup file to see log file.

```
[root@ip-172-31-93-54 ec2-user]# tar -tzf log_backup_2025-08-03.tar.gz
var/log/
var/log/dnf.log
var/log/dnf.librepo.log
var/log/dnf.rpm.log
var/log/hawkey.log
var/log/lastlog
var/log/journal/
var/log/journal/ec214777fbd977aed7252c91aa42fb4f/
var/log/journal/ec214777fbd977aed7252c91aa42fb4f/system.journal
var/log/journal/ec214777fbd977aed7252c91aa42fb4f/user-1000.journal
var/log/README
var/log/tallylog
var/log/private/
var/log/wtmp
var/log/btmp
var/log/sssd/
var/log/chrony/
var/log/chrony/tracking.log
```

Scenario 2: Security Group Remediation (External Script using AWS Lambda)

Objective

Automatically revoke 0.0.0.0/0 access from specific ports like 22 (SSH), 25 (SMPT) and 3306 (MYSQL) if detected in any security group.

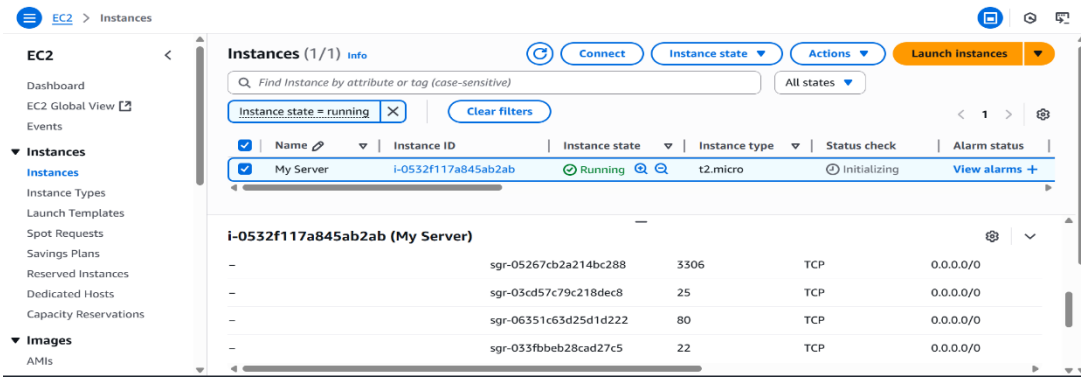
An example of Lambda Function (Python), EC2 instance and AWS SNS provisioned by Terraform can be found in the GitHub repo.

Trigger

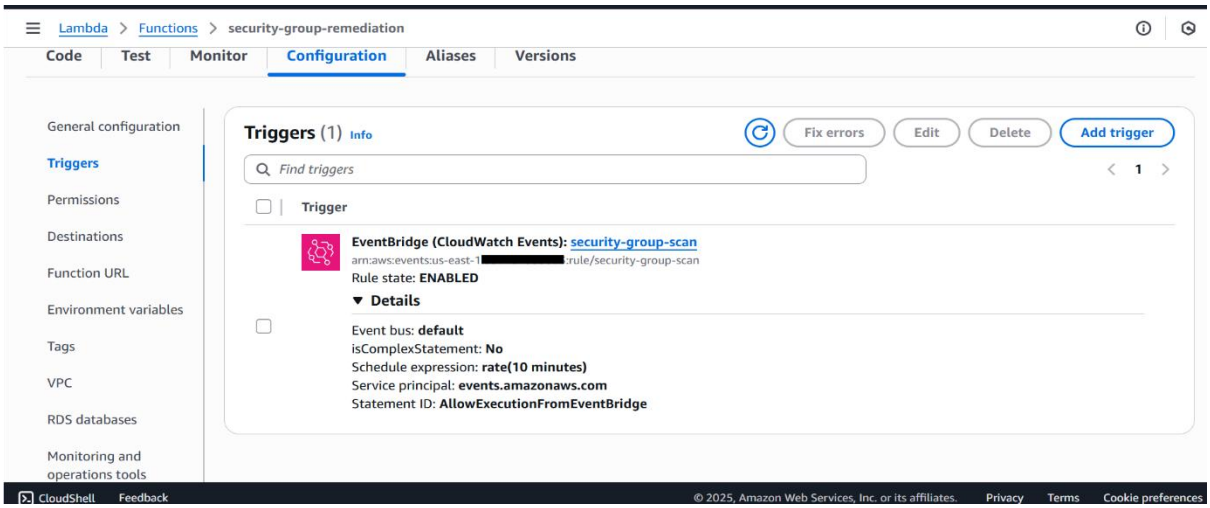
- Eventbridge Rule monitors security group every 10 minutes if there are any changes, if there are changes posing security risks, it triggers the lambda function to resolve it. An alert will also be sent via SNS by the function.

Demo: Images below shows results after applying the Terraform configuration.

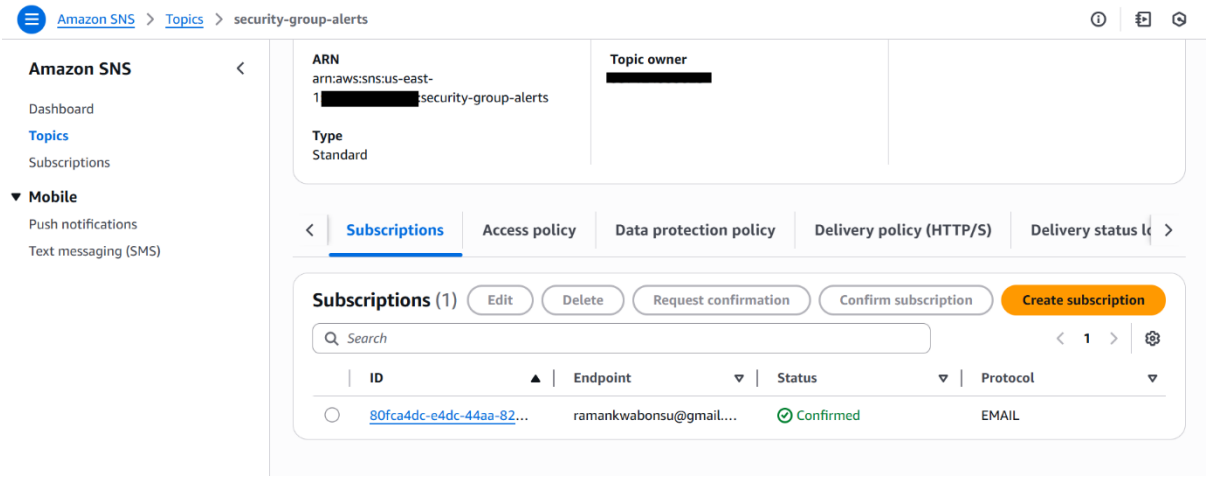
Instance is provisioned with various security groups listed.

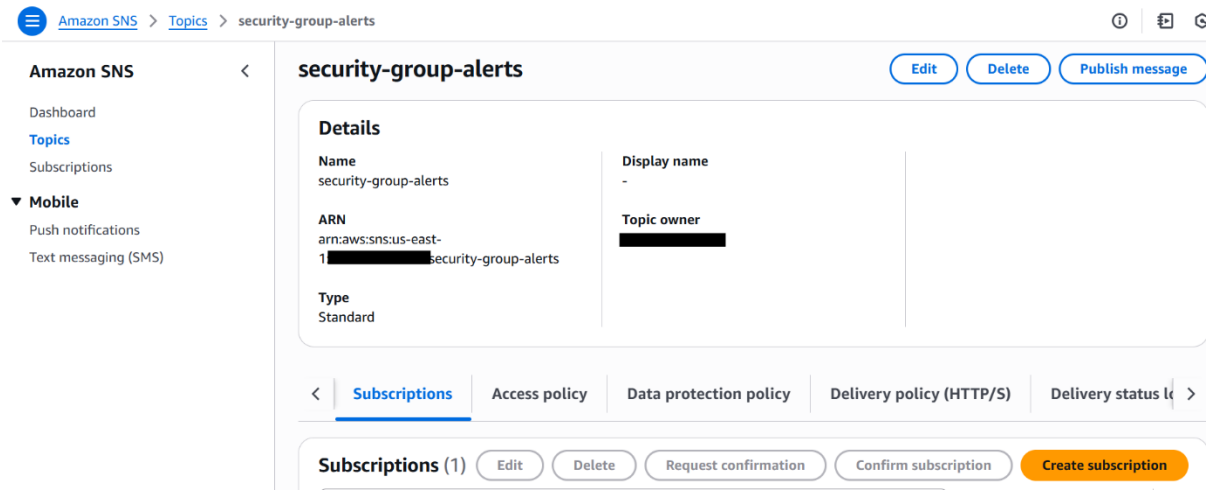


Lambda function Triggers shows an Eventbridge rule that runs every 10 minutes to check security groups.

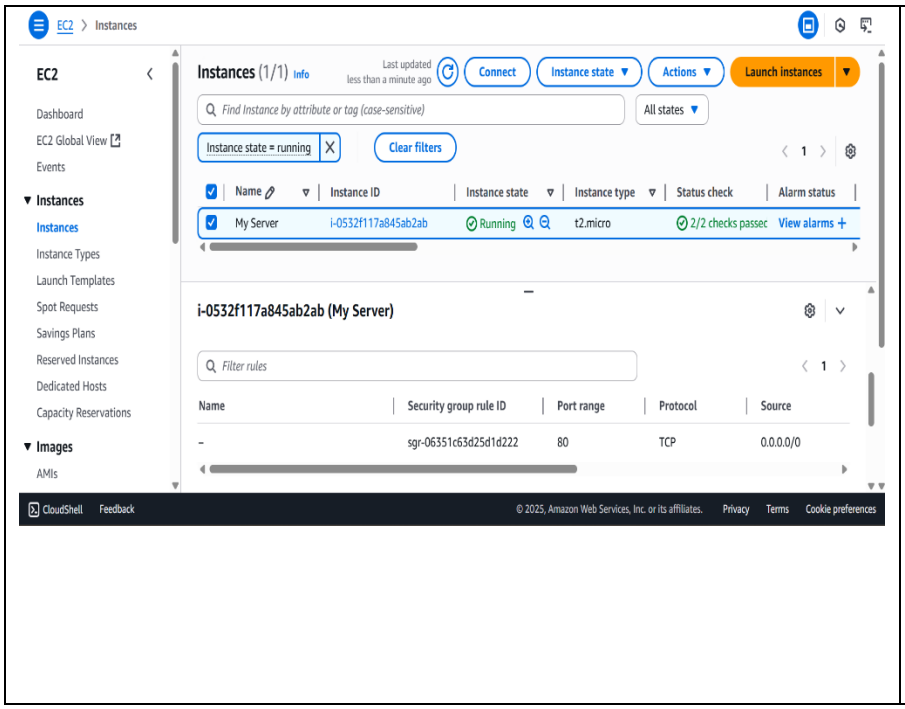


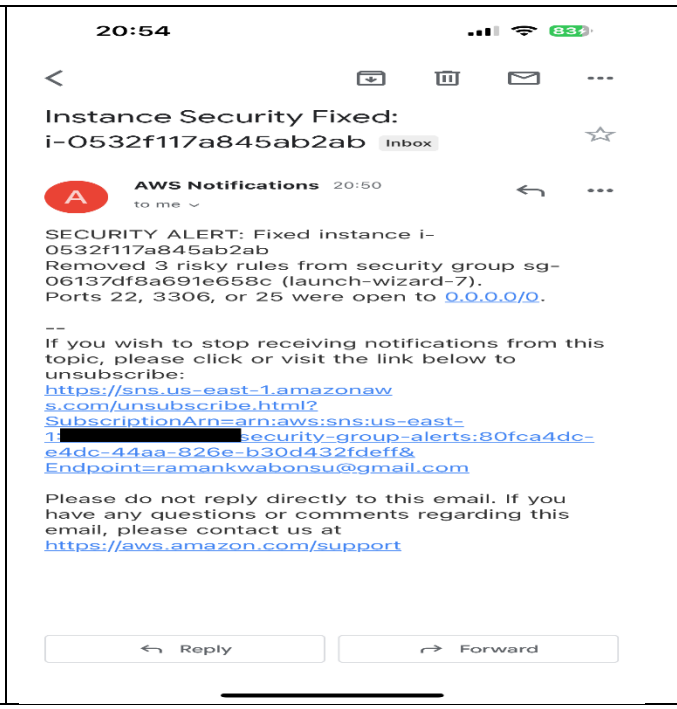
SNS is also provisioned with email confirmed and a topic called security-group-alerts subscribed to it.





After 10 minutes, security groups opened to the internet is removed with an email sent to alert the user.





Some external Lambda scripts can also be found here.

Scenario 3: HTTPD Service Health Checker (Internal bash Script).

Objective

Ensure that Apache (httpd) is always running. If it fails, restart the service.

Script: /home/ec2-user/install_httpd.sh (refer to repo)

Script: /home/ec2-user/health_check.sh (refer to repo)

Demo: Images below show how script is created and executed.

Install web server.

```
[root@ip-172-31-95-65 ec2-user]# nano install_httpd.sh
[root@ip-172-31-95-65 ec2-user]# chmod +x install_httpd.sh
[root@ip-172-31-95-65 ec2-user]# ls
install_httpd.sh
[root@ip-172-31-95-65 ec2-user]# ./install_httpd.sh
Amazon Linux 2023 Kernel Livepatch repository                    159 kB/s |  19 kB     00:00
Dependencies resolved.
Nothing to do.
Complete!
Last metadata expiration check: 0:00:01 ago on Sun Aug  3 20:27:00 2025.
Dependencies resolved.
================================================================================
 Package                 Architecture          Version                  Repository                Size
================================================================================
Installing:
 httpd                   x86_64                2.4.62-1.amzn2023        amazonlinux               48 k
Installing dependencies:
 apr                     x86_64                1.7.5-1.amzn2023.0.4     amazonlinux              129 k
 apr-util                x86_64                1.6.3-1.amzn2023.0.1     amazonlinux               98 k
```

Stop the web server, create and execute the health_check.sh. You can view message from the log file that is created from the script.

Cron Setup

*/5 * * * * /home/ec2-user/health_check.sh (cron job to run every 5 minutes)

Purpose

- This is a simple service watchdog script that:

1. Checks if httpd is running
2. If not running, logs the incident and automatically restarts it
3. Maintains a log of restart events in /var/log/health_check.log

Typically scheduled via cron to run at regular intervals for automated service recovery.

- Helps ensure high availability of web services.

Scenario 4: EC2 to S3 Sync Backup (Internal Bash Script).

Objective

Backup /var/www/html folder to S3 daily.

Script: /home/ec2-user/s3-sync.sh (refer to repo)

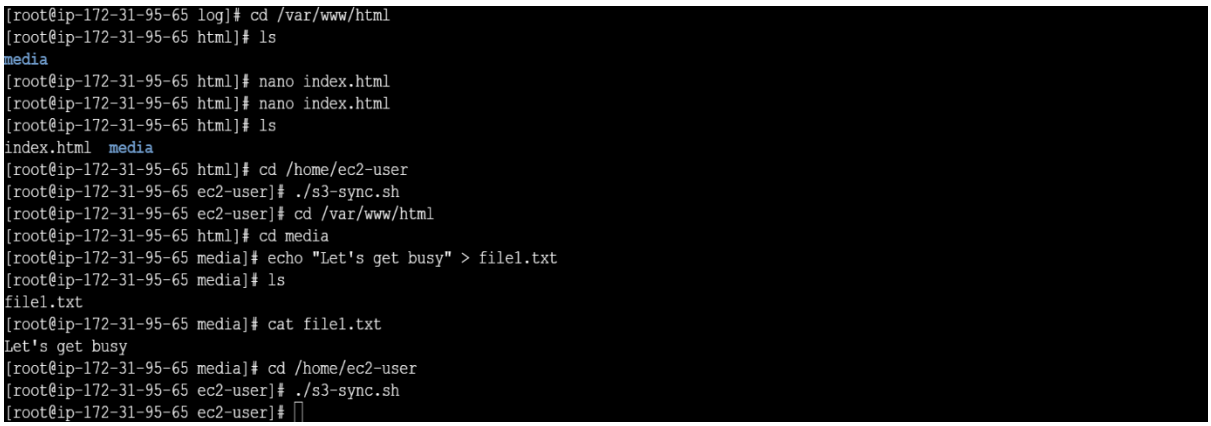
IAM Role Permission

Attach the following policy to the EC2 instance role:

```
{
  "Effect": "Allow",
  "Action": [
    "s3:PutObject",
    "s3:GetObject",
    "s3:ListBucket",
    "s3:DeleteObject"
  ],
  "Resource": [
    "arn:aws:s3:::my-bucket-name",
    "arn:aws:s3:::my-bucket-name/*"
  ]
}
```

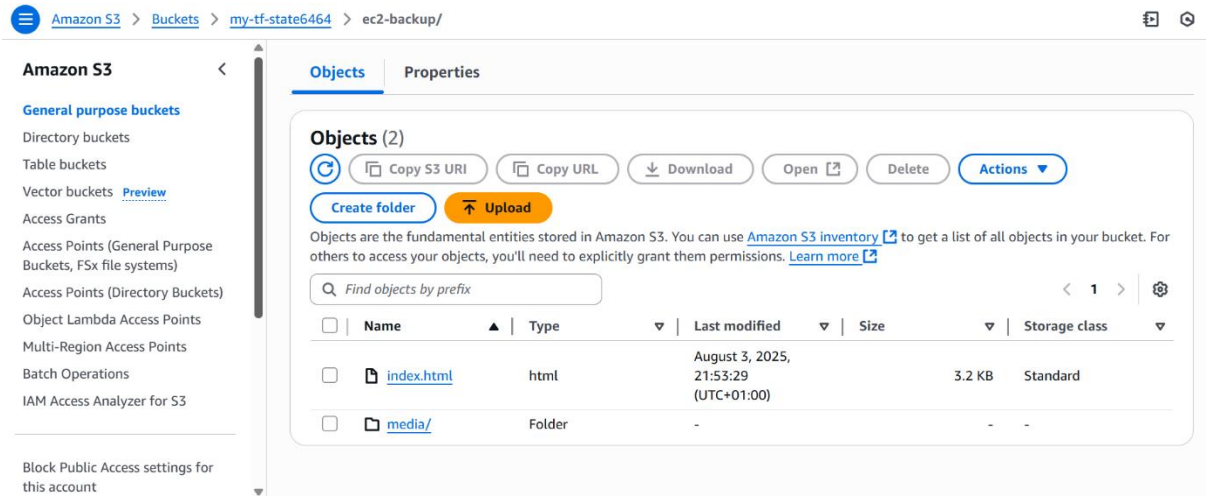
Demo: Images below show output when script is created and executed.

Create files in /var/www/html (default directory in httpd server to serve web files)



Create script file, make it executable and execute it. A log file below shows the backup operation was successful when script was executed.

Files being synced to S3 successfully.



Cron Setup

0 2 * * * /home/ec2-user/s3-sync.sh (cron job to run daily at 2 AM)

Purpose

This script creates an automated backup system that:

1. Mirrors local web files to S3
2. Removes deleted files from S3 to maintain exact sync
3. Logs all operations and results for monitoring
4. Typically runs via cron for regular backups

Requires AWS CLI configured with appropriate S3 permissions.

Scenario 5: Disk Usage Alert (Internal Bash Script)

Objective

Monitor disk usage and log warnings when it exceeds 10%.

Script: /home/ec2-user/check_disk.sh (refer to repo)

Demo: Images below shows warning message in log file created from the script when disk usage exceeded 10%.

Create script and make it executable

```
[root@ip-172-31-93-54 ec2-user]# nano check_disk.sh
[root@ip-172-31-93-54 ec2-user]# ls
backup_logs.sh  check_disk.sh  cron_log_backup.log  log_backup_2025-08-03.tar.gz
[root@ip-172-31-93-54 ec2-user]# chmod +x check_disk.sh
[root@ip-172-31-93-54 ec2-user]# ls
backup_logs.sh  check_disk.sh  cron_log_backup.log  log_backup_2025-08-03.tar.gz
[root@ip-172-31-93-54 ec2-user]# ./check disk.sh
```

```
[root@ip-172-31-93-54 ec2-user]# tail -f  /var/log/disk_alert.log
Sun Aug  3 00:49:40 UTC 2025: Disk usage warning on / - 20% used.
Sun Aug  3 00:49:40 UTC 2025: Disk usage warning on /boot/efi - 13% used.
```

Cron Setup

*/15 * * * * /home/ec2-user/check_disk.sh (cron job to run every 15 minutes)

Purpose

- Prevents EC2 failure due to full storage.

Final Thoughts

Scripting is a powerful tool for automation, applicable across IT, cloud, and development environments. These five scenarios show how scripting can automate crucial AWS tasks like backups, monitoring, security enforcement, and service reliability — both internally on EC2 and externally using AWS Lambda.