Backup Automation Script Report

Objective

As a Site Reliability Engineer, I was tasked with automating the process of taking backups of specific directories and saving them in an AWS S3 bucket. The process should run every Friday after working hours, with email notifications sent at every critical stage of the process, including backup creation, and successful or failed S3 uploads.

Script Overview

The bash script I developed automates the backup process and includes the following key steps:

1. **Creating the Source Directory**: Ensures the source directory exists or creates it if it doesn't.
2. **Creating Sub Files**: Simulates file creation within the source directory for backup purposes.
3. **Creating a Backup**: Compresses the source directory into a tar.gz file.
4. **Syncing Backup to S3**: Uploads the created backup file to an S3 bucket.
5. **Sending Email Notifications**: Sends an email at each critical stage of the process.

Backup Automation Script Report

Objective

As a Site Reliability Engineer, I was tasked with automating the process of taking backups of specific directories and saving them in an AWS S3 bucket. The process should run every Friday after working hours, with email notifications sent at every critical stage of the process, including backup creation, and successful or failed S3 uploads.

Script Overview

The bash script I developed automates the backup process and includes the following key steps:

1. **Creating the Source Directory**: Ensures the source directory exists or creates it if it doesn't.
2. **Creating Sub Files**: Simulates file creation within the source directory for backup purposes.
3. **Creating a Backup**: Compresses the source directory into a tar.gz file.
4. **Syncing Backup to S3**: Uploads the created backup file to an S3 bucket.
5. **Sending Email Notifications**: Sends an email at each critical stage of the process.

Script Details

Here is the detailed script used:

bash

```
#!/bin/bash

# Variables
SOURCE_DIR="/root/backupdata"
BACKUP_DIR="/root"
DATE=$(date +%Y-%m-%d)
BACKUP_FILE="$BACKUP_DIR/backup_$DATE.tar.gz"
KEYPEM_DIR="/root/Downloads/project2.pem"
BACKUP_SERVER_NAME="project2"
BACKUP_SERVER_IP="54.71.32.150"
BACKUP_SERVER_DIR="s3://project02/"
ADMIN_EMAIL="kayboateng360@gmail.com"

send_email() {
    local subject=$1
    local message=$2
    echo "$message" | mutt -s "$subject" "$ADMIN_EMAIL"
}

# Create source directory (if not exists)
mkdir -p "$SOURCE_DIR"
if [ $? -eq 0 ]; then
    send_email "Backup Process: Create Source Directory" "I
successfully created the source directory."
else
    send_email "Backup Process: Create Source Directory" "I failed to
create the source directory."
    exit 1
fi

# Touch files (if needed)
touch "$SOURCE_DIR/file1.txt" "$SOURCE_DIR/file1.pdf"
"$SOURCE_DIR/file1.docx"
if [ $? -eq 0 ]; then
    send_email "Backup Process: Create Sub Files" "I successfully
created subfiles in the source directory."
```

```
else
    send_email "Backup Process: Create Sub Files" "I failed to create
subfiles in the source directory."
    exit 1
fi

# Sleep for a while (if needed)
sleep 10s

# Create backup
tar -czvf "$BACKUP_FILE" "$SOURCE_DIR"
if [ $? -eq 0 ]; then
    send_email "Backup Process: Create Backup" "I successfully created
a zipped backup file."
    sleep 10s

    # Sync backup to S3
    aws s3 cp "$BACKUP_FILE" "$BACKUP_SERVER_DIR"
    if [ $? -eq 0 ]; then
        send_email "Backup Process: Send Backup" "I successfully sent
the backup to S3 bucket $BACKUP_SERVER_NAME."
    else
        send_email "Backup Process: Send Backup" "I failed to send the
backup to S3 bucket $BACKUP_SERVER_NAME."
    fi
else
    send_email "Backup Process: Create Backup" "I failed to create the
backup file."
    exit 1
fi
```

Cron Job Setup

To automate the script execution every Friday at 14:15 PM, I configured a cron job as follows:

cron

```
15 14 * * 5 /root/backupdata/backup.sh
```

This configuration ensures that the script runs at the specified time each week. The cron job line specifies:

- `15` - minute (15th minute)
- `14` - hour (14 PM)
- `*` - day of the month (any)
- `*` - month (any)
- `5` - day of the week (Friday)

Dependencies

- **mutt**: For sending email notifications.
- **aws-cli**: For interacting with AWS S3.

Both dependencies must be installed and configured correctly on the system where the script runs.

Permissions

The script must have execute permissions, which can be set using:

bash
```
chmod +x /root/backupdata/backup.sh
```

Conclusion

The script effectively automates the backup process, ensuring that backups are taken regularly and stored securely in an AWS S3 bucket. The inclusion of email notifications provides transparency and monitoring at every crucial stage, allowing for quick response in case of any issues. This automation not only saves time but also enhances the reliability and consistency of the backup process.

The IP address or hostname of the EC2 instance for verification;

```
54.71.32.150
```