**Scenario: Automating Backups with a Bash Script**

**Problem:**

You need to:

1. Back up a directory (/var/log/app/) to an AWS S3 bucket daily.
2. Ensure old backups are deleted after 7 days to save storage costs.
3. Log success or failure for monitoring.

**Solution: A Robust Bash Script**

Here’s the script:

bash

Copy code

#!/bin/bash

# Define variables

BACKUP\_SOURCE="/var/log/app/"

BACKUP\_DEST="/backup/"

S3\_BUCKET="s3://my-app-backups/"

LOG\_FILE="/var/log/backup\_script.log"

DATE=$(date +%F)

# Create backup directory if it doesn't exist

if [ ! -d "$BACKUP\_DEST" ]; then

mkdir -p "$BACKUP\_DEST"

fi

# Backup file name

BACKUP\_FILE="${BACKUP\_DEST}app\_backup\_${DATE}.tar.gz"

# Log function

log\_message() {

echo "$(date +'%Y-%m-%d %H:%M:%S') - $1" >> "$LOG\_FILE"

}

# Step 1: Create the backup

log\_message "Starting backup process."

tar -czf "$BACKUP\_FILE" "$BACKUP\_SOURCE" 2>>"$LOG\_FILE"

if [ $? -ne 0 ]; then

log\_message "Backup creation failed!"

exit 1

fi

log\_message "Backup created successfully: $BACKUP\_FILE"

# Step 2: Upload to S3

aws s3 cp "$BACKUP\_FILE" "$S3\_BUCKET" 2>>"$LOG\_FILE"

if [ $? -ne 0 ]; then

log\_message "S3 upload failed!"

exit 1

fi

log\_message "Backup uploaded to S3: $S3\_BUCKET"

# Step 3: Cleanup old backups

find "$BACKUP\_DEST" -type f -mtime +7 -exec rm {} \; 2>>"$LOG\_FILE"

if [ $? -ne 0 ]; then

log\_message "Failed to clean up old backups!"

exit 1

fi

log\_message "Old backups cleaned up."

log\_message "Backup process completed successfully."

exit 0

**Detailed Explanation**

1. **Variable Definitions:**
   * BACKUP\_SOURCE: Directory to back up.
   * BACKUP\_DEST: Local directory to temporarily store backups.
   * S3\_BUCKET: Target AWS S3 bucket for storing backups.
   * LOG\_FILE: File to store logs for monitoring.
2. **Error Handling:**
   * Every critical step checks for errors using if [ $? -ne 0 ]; then.
   * Logs failures for troubleshooting.
3. **Idempotency:**
   * Ensures the backup directory exists before proceeding (mkdir -p).
   * Deletes backups older than 7 days with find.
4. **Logging:**
   * Logs each action with timestamps for easy debugging.

**Cron Job to Automate the Script**

Schedule this script to run daily at 2 AM:

bash

Copy code

crontab -e

0 2 \* \* \* /path/to/backup\_script.sh

**Mock Interview Questions**

1. **Q:** How does find identify files older than 7 days?  
   **A:** The -mtime +7 flag finds files modified more than 7 days ago.
2. **Q:** How do you make AWS credentials secure when running a script?  
   **A:** Use:
   * AWS IAM roles if running on EC2.
   * AWS CLI aws configure with access keys and secret keys stored securely.
   * Use environment variables instead of hardcoding credentials.
3. **Q:** What happens if aws s3 cp fails in this script?  
   **A:** The script logs the failure and exits with an error code using exit 1.
4. **Q:** How would you test this script locally without uploading to S3?  
   **A:** Comment out the aws s3 cp line and check the backup file in the local directory.

**Customizations for a DevOps Role**

1. **Versioning Backups:**
   * Use S3 versioning to retain previous backups in case of accidental deletion:

bash

Copy code

aws s3api put-bucket-versioning --bucket my-app-backups --versioning-configuration Status=Enabled

1. **Monitoring with CloudWatch:**
   * Send logs to AWS CloudWatch for centralized monitoring:

bash

Copy code

aws logs create-log-group --log-group-name "/backup/logs"

aws logs create-log-stream --log-group-name "/backup/logs" --log-stream-name "daily-backup"

aws logs put-log-events --log-group-name "/backup/logs" --log-stream-name "daily-backup" --log-events file://log\_events.json

1. **Dockerize the Script:**
   * Wrap the script in a Docker container for portability.