Devops

AWS Resource Tracking using shell scripting

Here's the entire content formatted properly for your notes:

WHY MOVE TO CLOUD INFRASTRUCTURE?

Two Main Reasons:

1. Manageability:

- We don't want to create data centers
- We don't want to buy physical servers
- No hardware maintenance
- No need for system administrators to manage physical infrastructure
- Cloud provider handles all infrastructure management

2. Pay As You Use (Cost Efficiency):

Scenario: You have 100 developers in your organization. If we give all these 100 developers access to your AWS platform and everybody starts creating resources, we need to be cost effective.

Problem: We have to check if created resources are actually being used or not because we don't want to pay for something which we don't use.

Solution: As a DevOps engineer, we have to track the resource usage.

Multiple ways of tracking:

- Shell scripting (one of the ways)
- AWS Cost Explorer
- CloudWatch
- Third-party tools

PRACTICAL TASK: AWS RESOURCE TRACKING USING SHELL SCRIPT

Scenario: Organization: example.com Task: Track the usage of EC2, S3, Lambda, IAM Requirement: Send this info to manager everyday at 6 PM

Solution Approach:

1. Write a shell script to gather resource usage info
2. Save output to a file
3. Use cronjob to execute this task everyday at 6 PM
4. Use AWS CLI to get required output

STEP 1: INSTALL AND CONFIGURE AWS CLI

Check if AWS CLI is installed:

Command: aws

If not installed, you'll get an error.

Install AWS CLI:

Command: sudo snap install aws-cli --classic

Alternative installation methods:

- Ubuntu/Debian: sudo apt install awscli
- Using pip: pip install awscli
- Download from AWS website

Verify installation:

Command: aws --version

Output should show: aws-cli/2.x.x

Configure AWS CLI:

Command: aws configure

You will be prompted for:

1. AWS Access Key ID: Enter your access key
2. AWS Secret Access Key: Enter your secret key
3. Default region name: Enter your region (e.g., us-east-1, ap-south-1)
4. Default output format: json (recommended)

Where to get Access Keys:

1. Go to AWS Console
2. Click on your name (top right)

3. Click "Security credentials"
4. Scroll to "Access keys"
5. Click "Create access key"
6. Save the Access Key ID and Secret Access Key

STEP 2: CREATE THE SHELL SCRIPT

Create the script file:

Command: vi aws_resource_tracker.sh

Write the script:

#!/bin/bash ################# #Author: Sushmita Hubli #Date: 6th November 2024

#Version: v1 #This script will report the AWS resource usage ################# #AWS S3 #AWS EC2 #AWS Lambda #AWS IAM Users

set -x

List S3 buckets

echo "Print list of S3 buckets" aws s3 ls

List EC2 instances

echo "Print list of EC2 instances" aws ec2 describe-instances

List AWS Lambda functions

echo "Print list of Lambda functions" aws lambda list-functions

List IAM Users

echo "Print list of IAM users" aws iam list-users

Save the script: Esc + :wq!

Grant permissions:

Command: chmod 777 aws_resource_tracker.sh

Execute the script:

Command: sh aws_resource_tracker.sh

Result: You will get all the info, but this info is too much.

STEP 3: FILTERING SPECIFIC INFORMATION

Problem: What if I just want to print the instance ID and not all the info related to that instance?

Solution: The instance ID is present inside: Reservations → Instances → InstanceId

Command to get only Instance IDs:

```
aws ec2 describe-instances | jq '.Reservations[].Instances[].InstanceId'
```

Output: "i-04bdb20699c4e6783"

Command Explanation:

aws ec2 describe-instances:

- AWS CLI command to get information about EC2 instances
- Returns JSON output with all instance details

| (pipe):

- Sends output of first command to second command

jq:

- Command-line JSON processor
- Used to parse and filter JSON data
- Must be installed: sudo apt install jq

'.Reservations[].Instances[].InstanceId':

- JSON path to extract specific data
- . = Root of JSON
- Reservations[] = Array of reservations (iterate through all)
- Instances[] = Array of instances inside each reservation
- InstanceId = The specific field we want

Breakdown:

JSON Structure:

```
{
 "Reservations": [
  {
      "Instances": [
       {
         "InstanceId": "i-04bdb20699c4e6783",
         "InstanceType": "t2.micro",
         ...
       }
     ]
   }
 ]
}
```

The jq command navigates this structure and extracts only InstanceId.

Install jq if not available:

Command: sudo apt install jq

STEP 4: IMPROVED SCRIPT WITH FILTERED OUTPUT

Updated script:

#!/bin/bash ################# #Author: Sushmita Hubli #Date: 6th November 2024

#Version: v1 #This script will report the AWS resource usage #################

set -x

List S3 buckets

echo "List of S3 buckets:" aws s3 ls

List EC2 instance IDs

echo "List of EC2 Instance IDs:" aws ec2 describe-instances | jq '.Reservations[].Instances[].InstanceId'

List Lambda function names

echo "List of Lambda functions:" aws lambda list-functions | jq '.Functions[].FunctionName'

List IAM usernames

echo "List of IAM users:" aws iam list-users | jq '.Users[].UserName'

Additional useful filters:

Get Instance ID and Instance Type: aws ec2 describe-instances | jq '.Reservations[].Instances[] | {InstanceId, InstanceType}'

Get only running instances: aws ec2 describe-instances --filters "Name=instance-state-name,Values=running" | jq '.Reservations[].Instances[].InstanceId'

Get S3 bucket names only: aws s3 ls | awk '{print $3}'

STEP 5: REDIRECT OUTPUT TO A FILE

How to save output to a file:

Method 1: Using > (Overwrite)

Command: sh aws_resource_tracker.sh > resource_tracker.txt

This will:

- Execute the script
- Save ALL output to resource_tracker.txt
- Overwrite file if it already exists

Method 2: Using >> (Append)

Command: sh aws_resource_tracker.sh >> resource_tracker.txt

This will:

- Execute the script
- Append output to resource_tracker.txt
- Keep existing content and add new content

Method 3: Redirect within script

Modified script with output redirection:

#!/bin/bash ################# #Author: Sushmita Hubli #Date: 6th November 2024

#Version: v1 #This script will report the AWS resource usage #################

Output file

OUTPUT_FILE="/home/ubuntu/resource_tracker_$(date +%Y%m%d_%H%M%S).txt"

Redirect all output to file

exec > $OUTPUT_FILE 2>&1

```
echo "AWS Resource Usage Report" echo "Generated on: $(date)" echo
"================================"
```

List S3 buckets

```
echo "" echo "List of S3 buckets:" aws s3 ls
```

List EC2 instance IDs

```
echo "" echo "List of EC2 Instance IDs:" aws ec2 describe-instances | jq
'.Reservations[].Instances[].InstanceId'
```

List Lambda function names

```
echo "" echo "List of Lambda functions:" aws lambda list-functions | jq
'.Functions[].FunctionName'
```

List IAM usernames

```
echo "" echo "List of IAM users:" aws iam list-users | jq '.Users[].UserName'
```

```
echo "" echo "================================" echo "Report generation
completed"
```

Explanation:

- OUTPUT_FILE="/path/to/file_$(date +%Y%m%d_%H%M%S).txt" - Creates filename with timestamp
- exec > $OUTPUT_FILE 2>&1 - Redirects all output (stdout and stderr) to file
- $(date +%Y%m%d_%H%M%S) - Adds timestamp to filename (e.g., 20241106_180000)

Result: Creates a file like: resource_tracker_20241106_180000.txt

---

STEP 6: INTEGRATE WITH CRONJOB

What we want: Execute the script everyday at 6 PM and save output to file.

Open crontab editor:

Command: crontab -e

If first time, it will ask which editor to use. Choose nano or vi.

Add the following line:

```
0 18 * * * /home/ubuntu/aws_resource_tracker.sh >
/home/ubuntu/resource_tracker_$(date +%Y%m%d).txt 2>&1
```

Explanation of cron syntax:

0 18 * * * │ │ │ │ │ │ │ │ │ └─ Day of week (0-7, Sunday=0 or 7) │ │ │ └─── Month (1-12) │ │ └───── Day of month (1-31) │ └─────── Hour (0-23) └───────── Minute (0-59)

Breakdown:

- 0 = Minute 0 (on the hour)
- 18 = 6 PM (18:00 in 24-hour format)
-
  - = Every day of month
-
  - = Every month
-
  - = Every day of week

Full command explanation:

- /home/ubuntu/aws_resource_tracker.sh = Full path to script
- = Redirect output
- /home/ubuntu/resource_tracker_$(date +%Y%m%d).txt = Output file with date
- 2>&1 = Redirect errors to same file

Important notes:

1. Use FULL paths in cron jobs (not relative paths)
2. Escape % in cron: %
3. Make sure script has execute permissions (chmod 777)

Alternative cron entry with better practices:

0 18 * * * /bin/bash /home/ubuntu/aws_resource_tracker.sh >> /home/ubuntu/resource_tracker.log 2>&1

This appends to the same log file instead of creating new files.

Save and exit:

- If using nano: Ctrl+X, then Y, then Enter
- If using vi: Esc, then :wq!

Verify cron job:

Command: crontab -l

Output should show your cron job.

Check cron logs:

Command: grep CRON /var/log/syslog

This shows when cron jobs run.

COMPLETE SOLUTION

Final Script: aws_resource_tracker.sh

```
#!/bin/bash ################# #Author: Sushmita Hubli #Date: 6th November 2024

#Version: v1 #This script will report the AWS resource usage #################
```

Define output file with timestamp

```
TIMESTAMP=$(date +%Y-%m-%d_%H-%M-%S)
OUTPUT_FILE="/home/ubuntu/logs/resource_tracker_$TIMESTAMP.txt"
```

Create logs directory if it doesn't exist

```
mkdir -p /home/ubuntu/logs
```

Redirect all output to file

```
exec > $OUTPUT_FILE 2>&1

echo "========================================" echo "AWS Resource Usage
Report" echo "Generated on: $(date)" echo
"========================================"
```

List S3 buckets

```
echo "" echo "========== S3 BUCKETS ==========" aws s3 ls
```

List EC2 instances

```
echo "" echo "========== EC2 INSTANCES ==========" echo "Instance IDs:" aws ec2
describe-instances | jq '.Reservations[].Instances[] | {InstanceId, InstanceType, State:
.State.Name}'
```

List Lambda functions

```
echo "" echo "========== LAMBDA FUNCTIONS ==========" aws lambda list-
functions | jq '.Functions[] | {FunctionName, Runtime, LastModified}'
```

List IAM users

echo "" echo "========== IAM USERS ==========" aws iam list-users | jq '.Users[] | {UserName, CreateDate}'

echo "" echo "==" echo "Report generation completed successfully" echo "=="

Crontab entry:

0 18 * * * /home/ubuntu/aws_resource_tracker.sh

Or with email notification:

0 18 * * * /home/ubuntu/aws_resource_tracker.sh && echo "AWS Resource Report Generated" | mail -s "AWS Report" manager@example.com

---

TESTING THE SETUP

Test 1: Run script manually

sh aws_resource_tracker.sh

Check if file is created and contains expected output.

Test 2: Test cron with different time

For testing, set cron to run in 2 minutes:

1. Check current time: date
2. Set cron for 2 minutes later
3. Example: If current time is 14:30, set cron for 14:32

32 14 * * * /home/ubuntu/aws_resource_tracker.sh

Test 3: Check cron execution

Wait for scheduled time, then check:

Command: ls -ltr /home/ubuntu/logs/

You should see the generated file.

Test 4: View the generated report

Command: cat /home/ubuntu/logs/resource_tracker_*.txt

ADDITIONAL ENHANCEMENTS

1. Send email with report:

Install mail utility: sudo apt install mailutils

Modify script to send email: mail -s "AWS Resource Report" manager@example.com < $OUTPUT_FILE

2. Upload report to S3:

Add at end of script: aws s3 cp $OUTPUT_FILE s3://your-bucket-name/reports/

3. Clean up old reports:

Add to script: find /home/ubuntu/logs/ -name "resource_tracker_*.txt" -mtime +7 -delete

This deletes reports older than 7 days.

4. Add error handling:

if [$? -eq 0]; then echo "Report generated successfully" else echo "Error generating report" | mail -s "AWS Report Error" admin@example.com fi

5. Add resource counts:

echo "Total S3 buckets: $(aws s3 ls | wc -l)" echo "Total EC2 instances: $(aws ec2 describe-instances | jq '.Reservations[].Instances[] | length')"

TROUBLESHOOTING

Issue 1: Cron job not running

Check cron service: sudo systemctl status cron

Start cron: sudo systemctl start cron

Issue 2: Permission denied

Fix permissions: chmod 777 aws_resource_tracker.sh

Issue 3: AWS credentials not found

Verify AWS configuration: aws configure list

Check credentials file: cat ~/.aws/credentials

Issue 4: jq command not found

Install jq: sudo apt install jq

Issue 5: Output file not created

Check directory permissions: ls -la /home/ubuntu/logs/

Create directory with proper permissions: mkdir -p /home/ubuntu/logs chmod 755 /home/ubuntu/logs

---

SUMMARY

What we accomplished:

1. Installed and configured AWS CLI
2. Created shell script to track AWS resources (EC2, S3, Lambda, IAM)
3. Used jq to filter and format JSON output
4. Redirected output to a file with timestamp
5. Integrated script with cron job to run daily at 6 PM
6. Automated resource usage reporting

Key commands learned:

- aws s3 ls
- aws ec2 describe-instances
- aws lambda list-functions
- aws iam list-users
- jq for JSON parsing
- crontab for scheduling
- Output redirection (>, >>)

DevOps benefit:

- Automated cost tracking
- Daily resource usage monitoring
- No manual intervention required
- Audit trail of resource usage
- Easy to identify unused resources