ACTIVITY :7

**Task 1: Connect to the CLI Host EC2 Instance Using SSH (macOS/Linux)**

**Step 1: Download the SSH Key**

* Open the **Details** dropdown above your lab instructions.
* Click **Show** to reveal credentials.
* Click **Download PEM** to save the file labsuser.pem.
* Save it to a known location (e.g., ~/Downloads).

**Step 2: Set Permissions on the PEM File**

* Open Terminal on your Mac.
* Navigate to the directory where you saved the PEM file:

cd ~/Downloads

* Change the permissions of the key file to be read-only:

chmod 400 labsuser.pem

**Step 3: Get the Public IP of the CLI Host**

* In the **Details** dropdown > **Show**, locate the CliHostIP value.
* Copy this IP address.

**Step 4: Connect to the CLI Host via SSH**

* Run the following command (replace <CliHostIP> with the actual IP address):

ssh -i labsuser.pem ec2-user@<CliHostIP>

* When prompted with:

Are you sure you want to continue connecting (yes/no)?

Type yes and press Enter.

* You will be connected to the CLI Host EC2 instance.

**Task 1.2: Configure the AWS CLI on the CLI Host EC2 Instance**

**Step 1: Discover the AWS Region**

* On the connected CLI Host terminal, run:

curl http://169.254.169.254/latest/dynamic/instance-identity/document | grep region

* Note the region value (e.g., us-east-1).

**Step 2: Configure AWS CLI Credentials**

* Run the AWS CLI configuration command:

aws configure

* Enter the following when prompted:
  + **AWS Access Key ID:** Copy from the **Details** dropdown > **Show** (look for AccessKey).
  + **AWS Secret Access Key:** Copy from the same **Credentials** section (SecretKey).
  + **Default region name:** Use the region you discovered above (e.g., us-east-1).
  + **Default output format:** Enter json.

**Task 2: Enable VPC Flow Logs**

**Step 1: Create an S3 Bucket for VPC Flow Logs**

1. **Open your terminal on the CLI host** (or wherever AWS CLI is configured).
2. **Run this command to create an S3 bucket:**

aws s3api create-bucket --bucket flowlog#### --region <region> --create-bucket-configuration LocationConstraint=<region>

* Replace #### with **four random numbers** (e.g., flowlog1234).
* Replace <region> with the AWS region where your EC2 instances are running (e.g., us-east-2, eu-west-2, etc.).

**Important:**

* If your region is **us-east-1**, **omit** the --create-bucket-configuration LocationConstraint=<region> part, like this:

aws s3api create-bucket --bucket flowlog1234 --region us-east-1

**Step 2: Get the VPC ID of VPC1**

Run this command to find the VPC ID for the VPC named **VPC1**:

aws ec2 describe-vpcs --query 'Vpcs[\*].[VpcId,Tags[?Key==`Name`].Value,CidrBlock]' --filters "Name=tag:Name,Values='VPC1'"

* Note down the **VpcId** from the output. It will look like vpc-0abcd1234efgh5678.

**Step 3: Enable VPC Flow Logs on VPC1**

Run this command to create the flow logs:

aws ec2 create-flow-logs --resource-type VPC --resource-ids <vpc-id> --traffic-type ALL --log-destination-type s3 --log-destination arn:aws:s3:::flowlog####

* Replace <vpc-id> with the VPC ID you got in Step 2.
* Replace flowlog#### with your S3 bucket name from Step 1.

**Step 4: Confirm the Flow Log Creation**

Run this command:

aws ec2 describe-flow-logs

* Check the output for:
  + **FlowLogStatus** = ACTIVE
  + The **log destination** ARN pointing to your S3 bucket.

# Task 3: Analyze and troubleshoot access to resources on the Web Server EC2 instance

## Initial context:

* Web Server instance is in **public subnet** of **VPC1**.
* The **webpage does not load (ERR\_CONNECTION\_TIMED\_OUT)** when browsing the public IP.
* SSH connection to the Web Server also times out.
* You have confirmed the instance is **running** and you have the correct **key pair**.

# Challenge #1: Website is unreachable and SSH fails

### Step 1: Check instance status and basic info

Run this command to get key info:

aws ec2 describe-instances --filter "Name=ip-address,Values='<WebServerIP>'" --query 'Reservations[\*].Instances[\*].[State.Name,PrivateIpAddress,InstanceId,SecurityGroups[\*].GroupId,SubnetId,KeyName]'

* Confirm **State.Name** is "running".
* Get **SecurityGroupIds**, **SubnetId**, **KeyName** (should match your key).

### Step 2: Check if any ports are open on the Web Server

On CLI Host:

sudo yum install -y nmap

nmap <WebServerIP>

* If no ports are open (especially port 80 for HTTP and 22 for SSH), then something blocks incoming connections.
* If ports are open, web and SSH should work, so problem lies elsewhere.

### Step 3: Check Security Group attached to the Web Server instance

1. Get security group IDs from step 1.
2. Describe the security groups:

aws ec2 describe-security-groups --group-ids <WebServerSecurityGroupId>

1. Analyze inbound rules:

* Must have rules that **allow inbound HTTP (port 80)** and **SSH (port 22)** from your IP or 0.0.0.0/0.

If inbound rules are missing or incorrect, add them with:

aws ec2 authorize-security-group-ingress --group-id <GroupID> --protocol tcp --port 80 --cidr 0.0.0.0/0

aws ec2 authorize-security-group-ingress --group-id <GroupID> --protocol tcp --port 22 --cidr <YourPublicIP>/32

### Step 4: Check Route Table associated with the subnet of the Web Server instance

Get subnet ID from step 1.

aws ec2 describe-route-tables --filter "Name=association.subnet-id,Values='<VPC1PublicSubnetID>'"

* Confirm there is a route that sends 0.0.0.0/0 traffic **to the internet gateway (IGW)**.
* If missing, add route:

aws ec2 create-route --route-table-id <RouteTableId> --destination-cidr-block 0.0.0.0/0 --gateway-id <InternetGatewayId>

### Step 5: After confirming the above, refresh the browser page at http://<WebServerIP>.

* You should now see "Hello from your web server!".
* Visiting http://<WebServerIP>/mompopcafe/ should show the cafe website.

# Challenge #2: SSH still fails despite above fixes

### Step 1: Verify the Network ACLs attached to the public subnet

aws ec2 describe-network-acls --filter "Name=association.subnet-id,Values='<VPC1PublicSubnetID>'" --query 'NetworkAcls[\*].[NetworkAclId,Entries]'

* Look for any **DENY rules** that block inbound or outbound traffic for port 22 or ephemeral ports.
* Network ACL rules are stateless: if inbound is allowed but outbound is denied, connection fails.

### Step 2: Analyze NACL entries

* Each entry has **RuleNumber**, **Protocol**, **Port Range**, **Egress** (true=outbound, false=inbound), **RuleAction** (allow or deny), and **CIDR block**.
* Ensure:
  + Inbound: Allow port 22 (TCP) from your IP or 0.0.0.0/0
  + Outbound: Allow ephemeral ports (1024-65535) or all outbound (0.0.0.0/0).

### Step 3: Delete any blocking NACL entries (that deny port 22 or related traffic)

Example:

aws ec2 delete-network-acl-entry --network-acl-id <NetworkAclId> --rule-number <RuleNumber> --egress false

* Repeat for any other problematic entries.

### Step 4: Test SSH connection again:

ssh -i labsuser.pem ec2-user@<WebServerIP> -o ConnectTimeout=10

* You should now connect successfully.

### Step 5: Verify hostname on web server after SSH:

hostname

* Should return: web-server

**Task 4.1: Download and Extract the Flow Logs**

1. **Reconnect to your CLI Host EC2 instance:**

ssh -i your-key.pem ec2-user@<CLI-Host-IP>

1. **Create a local directory for flow logs:**

mkdir flowlogs

cd flowlogs

1. **List your S3 buckets to find your flow logs bucket name:**

aws s3 ls

Look for a bucket name that looks like flowlog#### or something similar.

1. **Download the flow logs from the bucket:**

Replace <flowlog####> with your actual bucket name from the previous step.

aws s3 cp s3://<flowlog####>/ . --recursive

You should see many files downloaded.

1. **Navigate inside the downloaded folders to find the log files:**

cd AWSLogs/<account-number>/vpcflowlogs/<region>/yyyy/mm/dd

ls

You will see multiple files with the .log.gz extension — these are compressed flow log files.

1. **Extract the logs:**

gunzip \*.gz

Now all .gz files will become plain .log files.

1. **Verify extraction:**

ls

You should see the .log files ready for analysis.

**Task 4.2: Analyze the Logs**

**Step 1: View the structure of a log file**

Pick a log file name from your directory, for example:

head <log-file-name>.log

You will see the header and log entries with fields like:

* version
* account-id
* interface-id
* source-address
* destination-address
* source-port
* destination-port
* protocol
* packets
* bytes
* start-time
* end-time
* action (ACCEPT or REJECT)
* log-status

**Step 2: Search for rejected traffic**

Find all lines with REJECT in all logs:

grep -rn REJECT .

Count how many rejected entries:

grep -rn REJECT . | wc -l

**Step 3: Narrow search to SSH port 22 rejected traffic**

grep -rn ' 22 ' . | grep REJECT

You should get fewer entries, specifically for port 22.

**Step 4: Get your public IP address as seen from AWS**

* In AWS Console → EC2 → Security Groups → WebSecurityGroup → Inbound rules → Edit
* Add a new rule, Source = **My IP**
* Copy the IP address (without the /32)

Cancel without saving any changes.

**Step 5: Refine your grep to your IP address (replace <ip-address>)**

bash

CopyEdit

grep -rn ' 22 ' . | grep REJECT | grep <ip-address>

This shows your failed SSH connection attempts recorded in the logs.

**Step 6: Verify the network interface ID matches your Web Server’s interface**

Get the public IP address of your Web Server instance (replace <WebServerIP>):

bash

CopyEdit

aws ec2 describe-network-interfaces --filters "Name=association.public-ip,Values='<WebServerIP>'" --query 'NetworkInterfaces[\*].[NetworkInterfaceId,Association.PublicIp]'

Compare the returned Network Interface ID to the one in your flow logs (third column typically).

**Step 7: Convert Unix timestamps to human-readable dates**

From one of the filtered logs, take a start-time or end-time timestamp, e.g., 1554496931, and convert it:

bash

CopyEdit

date -d @1554496931

This will display the human-readable date/time when the connection attempt was logged

ACTIVITY :8

### ****Task 1: Connect to the AWS CLI Host via SSH****

#### Task 1.1: SSH for Windows Users

* Use **PuTTY** with .ppk file.
* Enable keep-alives: Connection → Seconds between keepalives → 30.
* Auth path: Connection > SSH > Auth → Browse labsuser.ppk.
* Host IP: Use public IPv4 of **CLI Host EC2 instance**.
* Username: ec2-user.

#### Task 1.2: SSH for macOS/Linux Users

* Use terminal with .pem key.
* Navigate to PEM location:  
  cd ~/Downloads
* Change permission:  
  chmod 400 labsuser.pem
* Get public IPv4 address from EC2 instance.
* Connect via SSH:  
  ssh -i labsuser.pem ec2-user@<public-ip>

### ****Task 1.3: Configure AWS CLI****

Run in terminal:

bash

CopyEdit

curl http://169.254.169.254/latest/dynamic/instance-identity/document | grep region

Run CLI configuration:

bash

CopyEdit

aws configure

Provide:

* **AWS Access Key ID**
* **AWS Secret Access Key**
* **Region** (e.g., us-east-1)
* **Output format**: json

**Task 2: Create and initialize the Amazon S3 share bucket** using the AWS CLI (s3 subcommand):

**✅ Step 1: Create the S3 bucket**

Run the following command **in your SSH terminal**. Replace:

* xxxnnn → with your initials and a random 3-digit number (e.g., rb123)
* <region> → with your AWS region (e.g., us-east-1, us-west-2, etc.)

bash

CopyEdit

aws s3 mb s3://mompopcafe-rb123 --region us-west-2

✔️ If successful, you'll see:

makefile

CopyEdit

make\_bucket: mompopcafe-rb123

**✅ Step 2: Sync the initial images to the S3 bucket**

Upload the files from the ~/initial-images/ directory to the bucket under the /images prefix:

bash

CopyEdit

aws s3 sync ~/initial-images/ s3://mompopcafe-rb123/images

✔️ You should see output similar to:

bash

CopyEdit

upload: ~/initial-images/img1.jpg to s3://mompopcafe-rb123/images/img1.jpg

upload: ~/initial-images/img2.jpg to s3://mompopcafe-rb123/images/img2.jpg

...

**✅ Step 3: Verify the uploaded images**

List the contents of the /images prefix in your bucket, with a human-readable summary:

bash

CopyEdit

aws s3 ls s3://mompopcafe-rb123/images/ --human-readable --summarize

✔️ The output should look something like:

yaml

CopyEdit

2025-05-20 14:32:02 500.0 KiB img1.jpg

2025-05-20 14:32:02 200.0 KiB img2.jpg

Total Objects: 2

Total Size: 700.0 KiB

**🟢 Done!**

You’ve successfully:

* Created an S3 bucket
* Uploaded sample images
* Verified the upload with a detailed listing

### ✅ ****Task 3: Review the media company user and permissions****

#### 📘 ****Task 3.1: Review the**** mompopuser ****IAM User****

* Go to **IAM > Users > mompopuser**.
* Under **Permissions**, expand the AmazonS3ReadOnlyAccess policy.
  + ✅ **S3 Actions Allowed**:
    - Full List permissions
    - Limited Read permissions
  + ✅ **S3 Resources Affected**:
    - All objects in S3 (as defined in the policy JSON)

#### 📘 ****Task 3.2: Review the**** mediaco ****IAM Group****

* Go to **IAM > User groups > mediaco**.
* Under **Permissions**, expand the policies:

✅ **IAMUserChangePassword**

* + Allows group members to **change their own password**.

✅ **mediaCoPolicy**

* + **Statement 1 (AllowGroupToSeeBucketListInTheConsole)**
    - Lets users **view the list of S3 buckets** in the account (Console only).
  + **Statement 2 (AllowRootLevelListingOfTheBucket)**
    - Lets users **view top-level contents** in the mompopcafe bucket.
  + **Statement 3 (AllowUserSpecificActionsOnlyInTheSpecificPrefix)**
    - Grants **read, write, delete access** in mompopcafe-\*/images folder:
      * GetObject
      * PutObject
      * DeleteObject
      * Version-related permissions included

#### 📘 ****Task 3.3: Review the**** mediacouser ****IAM User****

* Go to **IAM > Users > mediacouser**.
* Under **Permissions**, verify:
  + IAMUserChangePassword ✅
  + mediaCoPolicy ✅
* Go to **Groups tab**, confirm membership in mediaco group.
* Go to **Security credentials tab > Access keys > Create access key**
  + 🔐 Download the .csv file with **Access Key & Secret Key**
* Copy your **12-digit AWS Account ID** (without dashes).
  + Found in the top-right dropdown (voclabs/user...)

#### 🧪 ****Task 3.4: Test**** mediacouser ****Permissions****

🧑‍💻 Open **Incognito** or different browser to login as mediacouser.

**Steps**:

1. Go to: <https://aws.amazon.com/console/>
2. Paste **Account ID** (no dashes), choose **IAM user login**.
3. Login:
   * **Username**: mediacouser
   * **Password**: Training1!

✅ **S3 Tests (Bucket: mompopcafe-xxxnnn)**:

1. 📂 Navigate to images folder.
2. 🖼️ **View**: Open Donuts.jpg – should load in new tab.
3. ⬆️ **Upload**: Upload a .png or .txt file – should succeed.
4. ❌ **Delete**: Select Cup-of-Hot-Chocolate.jpg, delete it – should succeed.
5. 🔒 **Unauthorized Test**:
   * Try accessing **Permissions tab** at bucket level – ❌ Access denied.
   * Try uploading file to **root** of bucket – ❌ Operation denied.

## ✅ Task 4.1: Create and Configure the SNS Topic

### 1. ****Create the SNS Topic****

* Go to **Simple Notification Service** (SNS) from the AWS Console.
* Select **Topics** → **Create topic**.
* Choose **Standard**.
* Enter **Name**: s3NotificationTopic.
* Click **Create topic**.

### 2. ****Save the Topic ARN****

* Copy the **Topic ARN** (e.g., arn:aws:sns:us-east-1:123456789012:s3NotificationTopic) for use in upcoming steps.

### 3. ****Set the Access Policy****

* Go to your newly created topic → click **Edit**.
* Expand **Access policy - optional**.
* Replace the JSON policy with the following **(customize ARN and bucket name)**:

json

CopyEdit

{

"Version": "2008-10-17",

"Id": "S3PublishPolicy",

"Statement": [

{

"Sid": "AllowPublishFromS3",

"Effect": "Allow",

"Principal": {

"Service": "s3.amazonaws.com"

},

"Action": "SNS:Publish",

"Resource": "arn:aws:sns:us-east-1:123456789012:s3NotificationTopic",

"Condition": {

"ArnLike": {

"aws:SourceArn": "arn:aws:s3:::mompopcafe-xxxnnn"

}

}

}

]

}

Replace:

* "Resource" with your **SNS Topic ARN**
* "aws:SourceArn" with your **S3 bucket ARN** (e.g., arn:aws:s3:::mompopcafe-xxxnnn)
* Click **Save changes**.

### 4. ****Subscribe**** mompopuser ****(You) to the Topic****

* Still in SNS, choose **Create subscription**.
* **Protocol**: Email
* **Endpoint**: <your email address> (you must access this inbox)
* Click **Create subscription**
* Go to your email inbox → confirm subscription via the **Confirm subscription** link.

## ✅ Task 4.2: Add Event Notification to the S3 Bucket

### 1. ****Create the Notification Configuration File****

* Open terminal/SSH into your **CLI Host instance**

bash

CopyEdit

vi s3EventNotification.json

* Press i to enter insert mode, then paste the following JSON (customized):

json

CopyEdit

{

"TopicConfigurations": [

{

"TopicArn": "arn:aws:sns:us-east-1:123456789012:s3NotificationTopic",

"Events": ["s3:ObjectCreated:\*", "s3:ObjectRemoved:\*"],

"Filter": {

"Key": {

"FilterRules": [

{

"Name": "prefix",

"Value": "images/"

}

]

}

}

}

]

}

Replace the "TopicArn" with your actual SNS topic ARN.

* Press ESC, then type :wq to save and exit.

### 2. ****Apply the Event Notification to the S3 Bucket****

Run this command in your SSH terminal:

bash

CopyEdit

aws s3api put-bucket-notification-configuration --bucket mompopcafe-xxxnnn --notification-configuration file://s3EventNotification.json

Replace mompopcafe-xxxnnn with your actual S3 bucket name.

### 3. ****Verify the Setup****

* Wait a moment and check your email inbox.
* You should receive an email with **subject: Amazon S3 Notification**.
* The body should look like:

json

CopyEdit

{

"Service": "Amazon S3",

"Event": "s3:TestEvent",

"Time": "2025-05-20T06:04:27.405Z",

"Bucket": "mompopcafe-xxxnnn",

"RequestId": "...",

"HostId": "..."

}

**Task 5: Test the Amazon S3 Share Bucket Event Notifications**! Here's a quick recap and checklist of what you did in this task to verify your S3 bucket configuration:

### ✅ Step-by-step Summary & Validation Checklist

#### 1. ****Configured AWS CLI with mediacouser credentials****

* Ran aws configure and entered:
  + **Access Key ID** & **Secret Access Key** from accessKeys.csv
  + Default region (kept unchanged)
  + Output format: json

#### 2. ****Tested**** put-object ****event notification****

* Command:

bash

CopyEdit

aws s3api put-object --bucket <mompopcafe-xxxnnn> --key images/Caramel-Delight.jpg --body ~/new-images/Caramel-Delight.jpg

* ✅ Email received with:
  + "eventName": "ObjectCreated:Put"
  + "key": "images/Caramel-Delight.jpg"

#### 3. ****Tested**** get-object ****operation (no notification expected)****

* Command:

bash

CopyEdit

aws s3api get-object --bucket <mompopcafe-xxxnnn> --key images/Donuts.jpg Donuts.jpg

* ✅ File downloaded
* ✅ **No email** received (as expected — get is not a tracked event)

#### 4. ****Tested**** delete-object ****event notification****

* Command:

bash

CopyEdit

aws s3api delete-object --bucket <mompopcafe-xxxnnn> --key images/Strawberry-Tarts.jpg

* ✅ Email received with:
  + "eventName": "ObjectRemoved:Delete"
  + "key": "images/Strawberry-Tarts.jpg"

#### 5. ****Tested unauthorized action (****put-object-acl****)****

* Command:

bash

CopyEdit

aws s3api put-object-acl --bucket <mompopcafe-xxxnnn> --key images/Donuts.jpg --acl public-read

* ❌ Operation failed with:

plaintext

CopyEdit

An error occurred (AccessDenied) when calling the PutObjectAcl operation: Access Denied

* ✅ Expected — confirms permissions are correctly restricted

ACTIVITY :9

**Task 1: Modify Security Group and Check Website**

1. Open **EC2** service from AWS Services menu.
2. Go to **Instances**, select **Cafe Web Server** instance.
3. Click **Security** tab → click **WebSecurityGroup**.
4. Under **Inbound rules**, notice only one rule: HTTP (TCP port 80).
5. Click **Edit inbound rules** → **Add Rule**:
   * **Port Range:** 22
   * **Source:** My IP (make sure it shows your exact IP with /32, NOT 0.0.0.0/0)
6. Click **Save rules**.
7. To check website:
   * Copy **Public IPv4 address** from Cafe Web Server instance.
   * Open browser and go to: http://<WebServerIP>/mompopcafe/ (replace <WebServerIP> with actual IP).
8. Website loads normally (e.g., bakery photos display correctly).

### Task 2: Create AWS CloudTrail Log & Observe the Hacked Website

#### Task 2.1: Create AWS CloudTrail Trail

1. In AWS Console, search and open **CloudTrail** service.
2. Go to **Trails** → **Create trail**.
3. Configure trail:
   * **Trail name:** Monitor (must be exactly this name)
   * **Storage location:** Create a new S3 bucket
   * **Trail log bucket/folder:** monitoring#### (#### = 4 random digits)
   * **Log file SSE-KMS encryption:** Uncheck (disable)
   * Keep other defaults, click **Next**.
4. Log events settings:
   * **Event type:** Keep Management events checked
   * **API activity:** Keep both Read and Write selected
   * Click **Next** → then **Create trail**.

#### Task 2.2: Observe the Hacked Website

1. Go back to your browser tab with the Café website, **refresh** page (hold **SHIFT** + refresh to avoid cache).
2. Wait about 1 minute for the hack to appear.
3. Notice the website is hacked — wrong image shown.
4. Investigate:
   * Go to **EC2** service → select Cafe Web Server instance → **Security** tab → **WebSecurityGroup** → **Inbound rules**.
   * You’ll see:
     + Your previous rule: port 22 from your IP only
     + New suspicious rule: port 22 open from anywhere (0.0.0.0/0)
5. Question: Who added this security hole?
   * Use **CloudTrail logs** to find who made this change. CloudTrail tracks user actions in your AWS account.

# Task 3.1 & 3.2: SSH to the EC2 Instance

### For macOS/Linux:

bash

CopyEdit

cd ~/Downloads

chmod 400 labsuser.pem

ssh -i labsuser.pem ec2-user@<Public-IPv4-of-Cafe-Web-Server-EC2>

Type yes if prompted to trust the host.

# Task 3.3: Download and extract the CloudTrail logs

bash

CopyEdit

mkdir ctraillogs

cd ctraillogs

aws s3 ls

Look for the bucket starting with monitoring in the output.

Then run (replace <monitoring####> with your bucket name):

bash

CopyEdit

aws s3 cp s3://<monitoring####>/ . --recursive

Once logs are downloaded, navigate to the logs directory (something like AWSLogs/<account-number>/CloudTrail/<region>/<year>/<month>/<day>):

bash

CopyEdit

cd AWSLogs/<account-num>/CloudTrail/<region>/<yyyy>/<mm>/<dd>/

ls

gunzip \*.gz

ls

# Task 3.4: Analyze logs using grep

First, pick a file name from the extracted logs, e.g., 1234567890\_CloudTrail\_us-east-1\_20230501.json

To pretty-print and see structure:

bash

CopyEdit

cat 1234567890\_CloudTrail\_us-east-1\_20230501.json | python3 -m json.tool

### Filter logs by IP address (your web server IP)

Replace <WebServerIP> with the public IP of your Cafe Web Server EC2:

bash

CopyEdit

ip=<WebServerIP>

for i in $(ls); do

echo "File: $i"

cat $i | python3 -m json.tool | grep sourceIPAddress

done

### Filter logs for event names:

bash

CopyEdit

for i in $(ls); do

echo "File: $i"

cat $i | python3 -m json.tool | grep eventName

done

# Task 3.5: Use AWS CLI CloudTrail commands to analyze events

### Lookup console login events:

bash

CopyEdit

aws cloudtrail lookup-events --lookup-attributes AttributeKey=EventName,AttributeValue=ConsoleLogin

### Lookup all actions on Security Groups:

bash

CopyEdit

aws cloudtrail lookup-events --lookup-attributes AttributeKey=ResourceType,AttributeValue=AWS::EC2::SecurityGroup --output text

### Find the security group ID attached to the Cafe Web Server:

bash

CopyEdit

region=$(curl -s http://169.254.169.254/latest/dynamic/instance-identity/document | grep region | cut -d '"' -f4)

sgId=$(aws ec2 describe-instances --filters "Name=tag:Name,Values='Cafe Web Server'" --query 'Reservations[\*].Instances[\*].SecurityGroups[\*].GroupId' --region $region --output text)

echo $sgId

### Filter CloudTrail events by the Security Group ID:

bash

CopyEdit

aws cloudtrail lookup-events --lookup-attributes AttributeKey=ResourceType,AttributeValue=AWS::EC2::SecurityGroup --region $region --output text | grep $sgId

# Task 4: Analyze CloudTrail Logs Using Amazon Athena

### Overview

CloudTrail logs contain detailed, verbose event data for every action in your AWS account. Searching these large JSON logs manually or via command-line filtering is tedious.  
**Amazon Athena** allows you to run SQL queries on logs stored in S3, making log analysis efficient and flexible.

## 4.1 Creating the Athena Table for CloudTrail Logs

* Navigate to **CloudTrail > Event history** in the AWS Console.
* Choose **Create Athena table** from Event history.
* Select the S3 bucket (e.g., monitoring####) where CloudTrail stores logs.
* Analyze the generated CREATE EXTERNAL TABLE SQL:
  + Maps CloudTrail JSON fields to table columns.
  + Nested JSON fields (like userIdentity) become STRUCT or ARRAY types.
  + Location points to S3 bucket holding the log files.
* Create the table. It appears in Athena with a default name incorporating the bucket.

## 4.2 Analyzing Logs in Athena

* Open **Athena console > Query Editor**.
* Locate the created CloudTrail table (e.g., cloudtrail\_logs\_monitoring####).
* Set query results location:
  + Browse to your monitoring S3 bucket.
  + Append /results/ to the path (e.g., s3://monitoring####/results/).
* Run simple exploratory queries:

sql

CopyEdit

SELECT \* FROM cloudtrail\_logs\_monitoring#### LIMIT 5;

* Inspect key columns:  
  useridentity, eventtime, eventsource, eventname, requestparameters.
* Query specific columns for clarity:

sql

CopyEdit

SELECT useridentity.userName, eventtime, eventsource, eventname, requestparameters

FROM cloudtrail\_logs\_monitoring####

LIMIT 30;

## Challenge: Identify the Hacker

### Goal

Find who modified the **security group** associated with the Cafe Web Server, when, from where, and how.

### Strategy & Useful Tips

* Use WHERE clauses to filter relevant events:
  + Focus on eventsource = 'ec2.amazonaws.com' (EC2-related events).
  + Look for eventname containing 'Security' or 'AuthorizeSecurityGroupIngress' (commonly used for modifying security groups).
  + Remove LIMIT to query the entire dataset.
  + Use LIKE with wildcards % to search for keywords inside event names or parameters.
* Sample query to list recent users and event types:

sql

CopyEdit

SELECT DISTINCT useridentity.userName, eventname, eventsource

FROM cloudtrail\_logs\_monitoring####

WHERE from\_iso8601\_timestamp(eventtime) > date\_add('day', -1, now())

ORDER BY eventsource;

* Check details of suspicious events to find:
  + User name who made the change.
  + Exact event time.
  + Source IP address (sourceIPAddress column).
  + Whether the hack was performed via Console or API (userAgent or additionalEventData).

### Sample Query to Identify Security Group Changes:

sql

CopyEdit

SELECT

useridentity.userName,

eventtime,

sourceIPAddress,

userAgent,

eventname,

requestparameters

FROM cloudtrail\_logs\_monitoring####

WHERE eventsource = 'ec2.amazonaws.com'

AND eventname LIKE '%SecurityGroup%'

ORDER BY eventtime DESC

LIMIT 20;

### Final Findings (Example)

* **Hacker User:** badactor123
* **Event Time:** 2025-05-18T14:23:45Z
* **Source IP:** 203.0.113.55
* **Method:** Programmatic access via AWS CLI (userAgent contains aws-cli/2.x)
* **Action:** AuthorizeSecurityGroupIngress — opened port 22 (SSH) to 0.0.0.0/0, creating a security hole.

# Task 5: Further Analysis of the Hack and Improving Security

## 5.1 Check the OS Users on the Web Server

* **Goal:** Identify unauthorized OS users who logged into the EC2 instance.
* Run to see recent authentication events:

bash

CopyEdit

sudo aureport --auth

* **Finding:** A suspicious user chaos-user logged in besides ec2-user.
* Check currently logged-in users:

bash

CopyEdit

who

* The chaos-user is still actively logged in.
* Attempt to delete the user:

bash

CopyEdit

sudo userdel -r chaos-user

* + This fails because the user session is active.
* Kill the active process associated with chaos-user (replace ProcNum with actual PID):

bash

CopyEdit

sudo kill -9 ProcNum

* Verify chaos-user is disconnected:

bash

CopyEdit

who

* Now delete chaos-user successfully:

bash

CopyEdit

sudo userdel -r chaos-user

* Check other users who can log in:

bash

CopyEdit

sudo cat /etc/passwd | grep -v nologin

* Confirm no other suspicious users except standard system accounts (root, sync, shutdown, halt).

## 5.2 Update SSH Security Settings

* Check SSH daemon config file and note recent modification:

bash

CopyEdit

sudo ls -l /etc/ssh/sshd\_config

* Edit the SSH config:

bash

CopyEdit

sudo vi /etc/ssh/sshd\_config

* Use :set number in vi to see line numbers.
* At line 61: PasswordAuthentication yes → **comment out** by adding # at the start.
* At line 63: #PasswordAuthentication no → **uncomment** by removing the #.
* Save and exit (:wq).
* Restart SSH daemon to apply changes:

bash

CopyEdit

sudo service sshd restart

* Note: Restarting SSH may disconnect current session; reconnect if needed.
* In the **EC2 Console**, go to the **Security Group** for the web server:
  + Edit inbound rules.
  + **Delete** the rule allowing port 22 access from 0.0.0.0/0 (open to the world).
  + Save the rules.

## 5.3 Fix the Website Content

* Navigate to website images directory:

bash

CopyEdit

cd /var/www/html/mompopcafe/images/

ls -l

* Hacker left a backup of original image.
* Restore the original image by renaming the backup:

bash

CopyEdit

sudo mv Coffee-and-Pastries.backup Coffee-and-Pastries.png

* Refresh the website in the browser:
  + Use Shift + refresh to avoid cached image.
* Verify the website looks correct again.

## 5.4 Delete the AWS Hacker IAM User

* Hacker also accessed AWS CLI and changed security group rules.
* In AWS Console, open **IAM** service.
* Go to **Users**, select the user chaos.
* Attempt to delete user — error appears because login profile exists.
* Delete the login profile via CLI:

bash

CopyEdit

aws iam delete-login-profile --user-name chaos

* Return to IAM Console and delete the chaos user successfully.

ACTIVITY :10

## Task 1.1.2: Connect to Mom & Pop Café instance (Ubuntu/Linux)

1. **Download the PEM key file** (labsuser.pem) from the AWS Console Credentials panel.
2. Open your terminal, navigate to the folder where labsuser.pem is saved. For example:

bash

CopyEdit

cd ~/Downloads

1. Set proper permissions on the PEM file:

bash

CopyEdit

chmod 400 labsuser.pem

1. Go to AWS Console → EC2 → Instances, select **MomPopCafeInstance**, and copy its **IPv4 Public IP**.
2. Connect via SSH:

bash

CopyEdit

ssh -i labsuser.pem ec2-user@<public-ip>

Replace <public-ip> with the actual IP you copied.

1. Type yes if prompted about the authenticity of the host.

You should now be connected to the **MomPopCafeInstance**.

## Task 1.1.3: Configure AWS CLI on the instance

While still connected to the MomPopCafeInstance:

1. Find the region the instance is running in:

bash

CopyEdit

curl http://169.254.169.254/latest/dynamic/instance-identity/document | grep region

Note the region value, e.g., us-east-1.

1. Run the AWS CLI configuration:

bash

CopyEdit

aws configure

When prompted, enter:

* + **AWS Access Key ID** (from the Credentials window)
  + **AWS Secret Access Key**
  + **Default region name** (the region you just found)
  + **Default output format**: json

Leave this SSH session open.

## Task 1.2: Connect to the CLI Host instance (new terminal window)

1. Open a new terminal window.
2. Repeat the **Task 1.1.2 steps** for the CLI Host instance:
   * Download PEM (if different, otherwise use the same PEM)
   * Set permissions on PEM
   * Find CLI Host public IP from EC2 Console
   * SSH into CLI Host:

bash

CopyEdit

ssh -i labsuser.pem ec2-user@<cli-host-public-ip>

1. Configure AWS CLI on the CLI Host just like you did on MomPopCafeInstance:

bash

CopyEdit

aws configure

Use the same credentials and region.

## Task 1.3: Remove MariaDB and resize the instance (on MomPopCafeInstance & CLI Host)

1. **Back on the MomPopCafeInstance SSH window:**

Stop MariaDB and uninstall it:

bash

CopyEdit

sudo systemctl stop mariadb

sudo yum -y remove mariadb-server

You should see Complete! after uninstall.

1. Close this SSH session to MomPopCafeInstance.
2. **Switch to CLI Host SSH window** (you opened in Task 1.2).
3. Find the Instance ID of MomPopCafeInstance:

bash

CopyEdit

aws ec2 describe-instances \

--filters "Name=tag:Name,Values=MomPopCafeInstance" \

--query "Reservations[\*].Instances[\*].InstanceId" \

--output text

Note the instance ID (e.g., i-0123456789abcdef0).

1. Stop the instance:

bash

CopyEdit

aws ec2 stop-instances --instance-ids <Instance-ID>

Replace <Instance-ID> with the real ID.

1. Change instance type to t2.micro:

bash

CopyEdit

aws ec2 modify-instance-attribute \

--instance-id <Instance-ID> \

--instance-type "{\"Value\": \"t2.micro\"}"

1. Start the instance:

bash

CopyEdit

aws ec2 start-instances --instance-ids <Instance-ID>

1. Wait for the instance to be running:

bash

CopyEdit

aws ec2 describe-instances \

--instance-ids <Instance-ID> \

--query "Reservations[\*].Instances[\*].[InstanceType,PublicDnsName,PublicIpAddress,State.Name]" \

--output table

Repeat until State.Name is running.

1. Note the new **Public DNS Name** and **Public IP Address** from the output.

## Task 1.3 final step: Test the website

Open your browser and visit:

pgsql

CopyEdit

http://<Public-DNS-Name>/mompopcafe

Replace <Public-DNS-Name> with the DNS you just recorded.

Check all functionality to confirm the website works with the downsized instance and remote RDS.

## 🧾 Task 2: Use the AWS Pricing Calculator to Estimate AWS Service Costs

### 🎯 ****Goal****

To estimate the monthly AWS cost of running the Mom & Pop Café website **before and after EC2 optimization** using the AWS Pricing Calculator, and then determine the **projected cost savings**.

## 🛠️ Task 2.1: ****Calculate the Costs Before Optimization****

### 🔧 Configuration (Before Optimization Topology):

| **Component** | **Configuration Detail** |
| --- | --- |
| **Region** | US East (N. Virginia) (example used) |
| **EC2 Instance** | - Instance type: t2.small  - Pricing: On-Demand  - OS: Linux  - Usage: 100% |
| **EBS Volume** | - Type: General Purpose SSD (gp2)  - Size: **40 GB** (includes 20 GB from old DB) |
| **RDS Instance** | - Type: db.t3.micro  - Engine: MariaDB  - Deployment: Single-AZ  - Storage: 20 GB gp2 |

### 📊 Steps to Estimate Cost (Before Optimization):

1. **Open AWS Pricing Calculator**  
   URL: <https://calculator.aws>  
   → Click **Create estimate**
2. **Add EC2 Instance:**
   * Click **Amazon EC2 > Configure**
   * Select Region: e.g., US East (N. Virginia)
   * Click **Advanced estimate**
   * Set:
     + OS: Linux
     + Usage: Constant
     + Instances: 1
     + Instance type: t2.small
     + Pricing model: On-Demand
     + Storage: General Purpose SSD (gp2), Size: 40 GB
   * Click **Save and add service**
3. **Add RDS Instance:**
   * Click **Amazon RDS for MariaDB > Configure**
   * Region: Same as EC2
   * Quantity: 1
   * Instance type: db.t3.micro
   * Deployment: Single-AZ
   * Pricing model: On-Demand
   * Storage: 20 GB, gp2
   * Click **Save and view summary**
4. **Export Estimate:**
   * Click **Share > Copy public link**
   * Click **Export > CSV** and save locally

### 💵 Estimated Monthly Costs (Before Optimization):

| **Service** | **Monthly Cost** |
| --- | --- |
| EC2 Instance | $20.89 |
| RDS Instance | $14.71 |
| **Total** | **$35.60** |

## 🛠️ Task 2.2: ****Calculate the Costs After Optimization****

### 🔧 Configuration Changes (After Optimization):

| **Component** | **Optimization Change** |
| --- | --- |
| **EC2 Instance** | - Changed instance type from t2.small → t2.micro |
| **EBS Volume** | - Reduced size from 40 GB → **20 GB** (removed old DB data) |
| **RDS Instance** | - Unchanged |

### 📊 Steps to Modify Estimate (After Optimization):

1. **Edit EC2 Entry:**
   * Click **Edit** next to EC2
   * Change Instance type to t2.micro
   * Change EBS storage to 20 GB
   * Click **Update**
2. **Export Estimate:**
   * Click **Export > CSV**
   * Save file to your system

### 💵 Estimated Monthly Costs (After Optimization):

| **Service** | **Monthly Cost** |
| --- | --- |
| EC2 Instance | $10.47 |
| RDS Instance | $14.71 |
| **Total** | **$25.18** |

## 🧮 Task 2.3: ****Calculate Cost Savings****

| **Scenario** | **EC2 Cost** | **RDS Cost** | **Total Monthly Cost** |
| --- | --- | --- | --- |
| **Before Optimization** | $20.89 | $14.71 | $35.60 |
| **After Optimization** | $10.47 | $14.71 | $25.18 |
| **💰 Savings** |  |  | **$10.42/month** |

ACTIVITY :11

## ✅ ****Task 1: Practice querying JSON-formatted data by using JMESPath****

### 🔹 Step-by-step Breakdown:

### ****1. Open the JMESPath Practice Tool****

Go to: <https://jmespath.org/>

This website allows you to **practice JSON queries interactively.**

### ****2. Replace the default JSON with:****

json

CopyEdit

{

"desserts": [

{

"name": "Chocolate cake",

"price": "20.00"

},

{

"name": "Ice cream",

"price": "15.00"

},

{

"name": "Carrot cake",

"price": "22.00"

}

]

}

### ****3. Try These Queries in the Expression Box****

| **Query Expression** | **Result** | **Explanation** |
| --- | --- | --- |
| desserts | Full array | Returns the entire desserts list |
| desserts[1] | Ice cream object | Returns second item (0-based index) |
| desserts[0].name | "Chocolate cake" | Returns only the name of first dessert |
| desserts[0].[name,price] | ["Chocolate cake", "20.00"] | Returns specific attributes |
| desserts[].name | ["Chocolate cake", "Ice cream", "Carrot cake"] | Returns names of all desserts |
| desserts[?name=='Carrot cake'] | [{ "name": "Carrot cake", "price": "22.00" }] | Uses filter to find dessert by name |

### ****4. Now Test with CloudFormation-style JSON:****

Replace the JSON with:

json

CopyEdit

{

"StackResources": [

{

"LogicalResourceId": "VPC",

"ResourceType": "AWS::EC2::VPC"

},

{

"LogicalResourceId": "PublicSubnet1",

"ResourceType": "AWS::EC2::Subnet"

},

{

"LogicalResourceId": "CliHostInstance",

"ResourceType": "AWS::EC2::Instance"

}

]

}

Try this query:

jmespath

CopyEdit

StackResources[?ResourceType == 'AWS::EC2::Instance'].LogicalResourceId

✅ **Result:**

json

CopyEdit

["CliHostInstance"]

## ✅ ****Task 2 Summary: CloudFormation Stack Troubleshooting****

### ### 🔐 ****2.1: SSH to CLI Host****

Depending on your OS:

#### ****Windows:****

1. **Download the labsuser.ppk** file.
2. Open **PuTTY**, go to:
   * Session: Enter the **Public IP** of CLI Host.
   * Connection: Set keepalives to 30.
   * Connection > SSH > Auth: Load the .ppk file.
3. Click **Open**, then **Accept**, and log in as:

bash

CopyEdit

ec2-user

#### ****macOS/Linux:****

1. **Download the labsuser.pem** file.
2. Set permissions:

bash

CopyEdit

chmod 400 labsuser.pem

1. SSH into the CLI host:

bash

CopyEdit

ssh -i labsuser.pem ec2-user@<public-ip>

### 🌐 ****2.2: Configure AWS CLI****

1. Discover region:

bash

CopyEdit

curl http://169.254.169.254/latest/dynamic/instance-identity/document | grep region

Example output: "region" : "us-east-1"

1. Configure AWS CLI:

bash

CopyEdit

aws configure

* + Access Key ID: from Credentials panel
  + Secret Access Key: from Credentials panel
  + Region: as above (e.g., us-east-1)
  + Output format: json

### 📦 ****2.3: Create CloudFormation Stack****

1. View template:

bash

CopyEdit

less template1.yaml

# Press q to exit

1. Create the stack:

bash

CopyEdit

aws cloudformation create-stack \

--stack-name myStack \

--template-body file://template1.yaml \

--capabilities CAPABILITY\_NAMED\_IAM \

--parameters ParameterKey=KeyName,ParameterValue=vockey

1. Watch stack resource creation:

bash

CopyEdit

watch -n 5 -d \

aws cloudformation describe-stack-resources \

--stack-name myStack \

--query 'StackResources[\*].[ResourceType,ResourceStatus]' \

--output table

1. Wait a few minutes. Notice resources **start to delete**. This means **stack creation failed**.
2. View stack status:

bash

CopyEdit

watch -n 5 -d \

aws cloudformation describe-stacks \

--stack-name myStack \

--output table

1. Get **failure reason**:

bash

CopyEdit

aws cloudformation describe-stack-events \

--stack-name myStack \

--query "StackEvents[?ResourceStatus == 'CREATE\_FAILED']"

* + You’ll likely see:

json

CopyEdit

"ResourceStatusReason": "WaitCondition timed out. The WaitCondition didn’t receive the required number of signals from an Amazon EC2 instance."

This means the **user-data script didn’t signal success** back, possibly due to an error in that script or execution delay.

1. Confirm final stack status:

bash

CopyEdit

aws cloudformation describe-stacks \

--stack-name myStack \

--output table

→ Status should be ROLLBACK\_COMPLETE

1. Delete the failed stack:

bash

CopyEdit

aws cloudformation delete-stack --stack-name myStack

## ****Task 2.4: Avoid Rollback on an AWS CloudFormation Stack****

Now, re-run the create-stack command with rollback disabled. This allows you to debug failures without losing resources.

bash

CopyEdit

aws cloudformation create-stack \

--stack-name myStack \

--template-body file://template1.yaml \

--capabilities CAPABILITY\_NAMED\_IAM \

--on-failure DO\_NOTHING \

--parameters ParameterKey=KeyName,ParameterValue=vockey

--on-failure DO\_NOTHING tells CloudFormation not to delete any resources if stack creation fails. This helps with investigation.

### ****Monitor Stack Resource Creation****

Run the following to check resource creation status:

bash

CopyEdit

watch -n 5 -d \

aws cloudformation describe-stack-resources \

--stack-name myStack \

--query 'StackResources[\*].[ResourceType,ResourceStatus]' \

--output table

Press **CTRL+C** to exit the watch utility once all resources are in a final state (CREATE\_COMPLETE or CREATE\_FAILED).

### ****Check Stack Status****

bash

CopyEdit

aws cloudformation describe-stacks \

--stack-name myStack \

--output table

You should see StackStatus as CREATE\_FAILED.

### ****View Failure Events****

bash

CopyEdit

aws cloudformation describe-stack-events \

--stack-name myStack \

--query "StackEvents[?ResourceStatus == 'CREATE\_FAILED']"

You’ll likely see that the **WaitCondition** failed due to timeout.

### ****Get EC2 Public IP****

bash

CopyEdit

aws ec2 describe-instances \

--filters "Name=tag:Name,Values='Web Server'" \

--query 'Reservations[].Instances[].[State.Name,PublicIpAddress]'

Copy the public IP address of the EC2 instance.

### ****Connect to EC2 Instance via SSH****

Open a **new terminal** and run:

bash

CopyEdit

ssh -i vockey.pem ec2-user@<Public-IP-Here>

Replace <Public-IP-Here> with the IP you copied. Use correct user and PEM file based on your OS and AMI.

### ****Inspect Log Files for Failure****

#### View cloud-init log:

bash

CopyEdit

sudo tail -50 /var/log/cloud-init-output.log

Look for:

* No package http available.
* util.py[WARNING]: Failed running /var/lib/cloud/instance/scripts/part-001

### ****View the Userdata Script****

bash

CopyEdit

sudo cat /var/lib/cloud/instance/scripts/part-001

You'll see a script that includes:

bash

CopyEdit

#!/bin/bash -e

...

yum install -y http

The problem is: there is **no package named http** — it should be httpd (Apache).

The -e in the shebang causes the script to **fail immediately** on any error.

Exit the instance:

bash

CopyEdit

exit

## ****Task 2.5: Fix the Issue and Successfully Create the Stack****

### ****Edit the CloudFormation Template****

Open the template:

bash

CopyEdit

vim template1.yaml

Use the arrow keys or type :128 to go to line 128.

Change this line:

yaml

CopyEdit

yum install -y http

to:

yaml

CopyEdit

yum install -y httpd

Then save and exit:

vim

CopyEdit

Press `a` to edit, then ESC, then type :wq and hit ENTER

### ****Verify the Fix****

bash

CopyEdit

cat template1.yaml | grep httpd

You should see the corrected line(s).

### ****Delete the Failed Stack****

bash

CopyEdit

aws cloudformation delete-stack --stack-name myStack

Then monitor deletion:

bash

CopyEdit

watch -n 5 -d \

aws cloudformation describe-stacks \

--stack-name myStack \

--output table

When output disappears (stack no longer exists), press **CTRL+C**.

### ****Recreate the Stack****

bash

CopyEdit

aws cloudformation create-stack \

--stack-name myStack \

--template-body file://template1.yaml \

--capabilities CAPABILITY\_NAMED\_IAM \

--on-failure DO\_NOTHING \

--parameters ParameterKey=KeyName,ParameterValue=vockey

Monitor resource creation:

bash

CopyEdit

watch -n 5 -d \

aws cloudformation describe-stack-resources \

--stack-name myStack \

--query 'StackResources[\*].[ResourceType,ResourceStatus]' \

--output table

Wait until all resources have completed. Then press **CTRL+C**.

### ****Verify Final Stack Status****

bash

CopyEdit

aws cloudformation describe-stacks \

--stack-name myStack \

--output table

You should now see:

bash

CopyEdit

StackStatus: CREATE\_COMPLETE

Also, check the **Outputs** section for:

* **Public IP** of the EC2 instance (web server)
* **S3 Bucket name**

### ****Test the Web Server****

1. Copy the public IP from the outputs.
2. Open a browser tab.
3. Enter the public IP into the address bar.

✅ You should see:

pgsql

CopyEdit

Hello from your web server!

## Task 3 Summary: Detecting and Handling Drift in AWS CloudFormation

### ****Step 3.1: Manually Modify Security Group****

1. Go to **AWS Management Console → EC2 → Instances → Select "Web Server"**.
2. Click on the security group **WebServerSG** linked in the description.
3. Go to **Inbound rules → Edit**.
4. Change the **SSH (port 22) rule's Source** from 0.0.0.0/0 (any IP) to **My IP** (your current IP).
5. Save the changes.

### ****Step 3.2: Add an Object to the S3 Bucket****

In your **CLI Host terminal:**

bash

CopyEdit

# Get the bucket name from CloudFormation stack outputs and store in a variable

bucketName=$(aws cloudformation describe-stacks \

--stack-name myStack \

--query "Stacks[\*].Outputs[?OutputKey=='BucketName'].[OutputValue]" \

--output text)

echo "bucketName = $bucketName"

# Create an empty file

touch myfile

# Copy the file to the S3 bucket

aws s3 cp myfile s3://$bucketName/

# Verify the file exists in the bucket

aws s3 ls $bucketName/

### ****Step 3.3: Detect Drift****

1. Start drift detection:

bash

CopyEdit

driftId=$(aws cloudformation detect-stack-drift --stack-name myStack --query StackDriftDetectionId --output text)

echo "Drift detection started: $driftId"

1. Check drift detection status (repeat until status is not DETECTION\_IN\_PROGRESS):

bash

CopyEdit

aws cloudformation describe-stack-drift-detection-status --stack-drift-detection-id $driftId --query StackDriftStatus --output text

1. Confirm stack drift status:

bash

CopyEdit

# Should show DRIFTED if manual changes detected

1. List resource drifts with a readable query:

bash

CopyEdit

aws cloudformation describe-stack-resources \

--stack-name myStack \

--query 'StackResources[\*].[ResourceType,ResourceStatus,DriftInformation.StackResourceDriftStatus]' \

--output table

* Expect all resources **IN\_SYNC** except the manually modified **security group** (should be **MODIFIED**).
* The S3 bucket remains **IN\_SYNC** since file content changes do not register as drift.

1. See detailed drift differences for modified resources:

bash

CopyEdit

aws cloudformation describe-stack-resource-drifts \

--stack-name myStack \

--stack-resource-drift-status-filters MODIFIED

* Property differences will show that port 22 source IP changed from 0.0.0.0/0 to your specific IP.

### ****Attempt to Update Stack****

Try updating the stack normally:

bash

CopyEdit

aws cloudformation update-stack \

--stack-name myStack \

--template-body file://template1.yaml \

--parameters ParameterKey=KeyName,ParameterValue=vockey

* **Expected:** Update fails due to drift.
* Drift is **not automatically resolved** by updates.
* You must either update the CloudFormation template to match manual changes or manually revert changes in AWS Console.

**Task 4: Deleting the Stack When S3 Bucket Has Objects**

**Step 1: Try deleting the stack (expected to fail)**

bash

CopyEdit

aws cloudformation delete-stack --stack-name myStack

Then watch stack resources deletion status:

bash

CopyEdit

watch -n 5 -d aws cloudformation describe-stack-resources --stack-name myStack --query 'StackResources[\*].[ResourceType,ResourceStatus]' --output table

You will notice that all resources except the S3 bucket get deleted. The bucket fails because it has objects.

**Step 2: Check stack status (will show DELETE\_FAILED)**

bash

CopyEdit

aws cloudformation describe-stacks --stack-name myStack --output table

Notice "StackStatus": "DELETE\_FAILED" and "StackStatusReason": "The following resource(s) failed to delete: [MyBucket]."

**Challenge: Keep the bucket and file, but still delete the stack**

**Why?**

* You **cannot delete an S3 bucket if it has objects**.
* You want to keep the bucket and objects **intact**.
* You want the **stack deleted successfully**.

**Step 3: Get the Logical Resource ID for the S3 bucket**

Run this command to get the logical resource ID for the bucket resource in your stack:

bash

CopyEdit

aws cloudformation list-stack-resources --stack-name myStack --query "StackResourceSummaries[?ResourceType=='AWS::S3::Bucket'].LogicalResourceId" --output text

Suppose the output is:

nginx

CopyEdit

MyBucket

**Step 4: Delete the stack *retaining* the S3 bucket resource**

Use the --retain-resources option with the logical ID of the bucket:

bash

CopyEdit

aws cloudformation delete-stack --stack-name myStack --retain-resources MyBucket

This will:

* Delete all stack resources **except** the specified bucket.
* The bucket and its files remain untouched.
* The stack will delete successfully.

**Step 5: Verify the stack deletion**

Run:

bash

CopyEdit

aws cloudformation describe-stacks --stack-name myStack

* You will either get an error if the stack is deleted or see status DELETE\_COMPLETE.

**Optional Bonus: One-liner script**

Get the bucket logical ID and delete the stack retaining the bucket:

bash

CopyEdit

bucketLogicalId=$(aws cloudformation list-stack-resources --stack-name myStack --query "StackResourceSummaries[?ResourceType=='AWS::S3::Bucket'].LogicalResourceId | [0]" --output text)

aws cloudformation delete-stack --stack-name myStack --retain-resources $bucketLogicalId