

AWS Project

Automated EC2 Lifecycle Management with CloudFormation and Lambda

By: Subhradeep Manna | July 15, 2025

Project Description

This project demonstrates how to automate the lifecycle of AWS EC2 instances using a combination of AWS CloudFormation and AWS Lambda. The goal is to provision infrastructure as code and manage it efficiently by automatically cleaning up unused resources.

Project Overview

This project showcases how to automate the lifecycle of EC2 instances using AWS services:

1. **Create** resources (EC2 + Security Group) using CloudFormation.
 2. **Manually stop** the EC2 instances to simulate idle use.
 3. **Use a Lambda function** (Python-based) to check which EC2 instances have been stopped for over **5 minutes**, and **automatically delete** them.
 4. **Run Lambda manually** (no CloudWatch trigger).
-

Step-by-Step Process

Step 1: Create EC2 Instances Using CloudFormation

1. Open AWS Console → Go to CloudFormation
 - Click "Create Stack" → With new resources (standard)
2. Upload or paste this template:

```

AWSTemplateFormatVersion: "2010-09-09"
Description: Launch 3 EC2 instances in an existing VPC and subnet
Parameters:
  KeyName:
    Type: AWS::EC2::KeyPair::KeyName
    Description: Name of an existing EC2 key pair for SSH access
Resources:
  Security group that allows inbound SSH from anywhere
  InstanceSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupDescription: Enable SSH access
      VpcId: vpc-0c32d77a24c43392b
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0

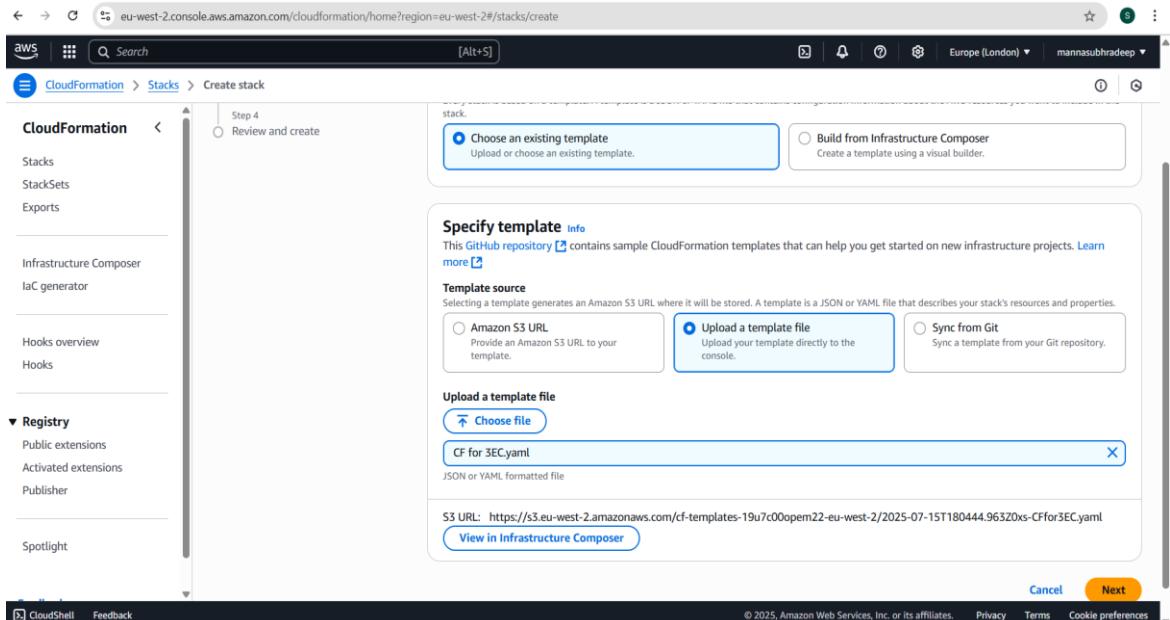
      Make sure the subnet is explicitly associated with the given route table
      SubnetRouteAssociation:
        Type: AWS::EC2::SubnetRouteTableAssociation
        Properties:
          SubnetId: subnet-0cc35aea439788393
          RouteTableId: rtb-05e1708fdcbcc5a88
--- EC2 instances
  EC2Instance1:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-0841b1152f02fa85e
      InstanceType: t3.micro
      KeyName: !Ref KeyName
      SubnetId: subnet-0cc35aea439788393
      SecurityGroupIds:
        - !Ref InstanceSecurityGroup
      Tags:
        - Key: Name
          Value: EC2-Instance-1
  EC2Instance2:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-0841b1152f02fa85e
      InstanceType: t3.micro
      KeyName: !Ref KeyName
      SubnetId: subnet-0cc35aea439788393
      SecurityGroupIds:
        - !Ref InstanceSecurityGroup
      Tags:
        - Key: Name
          Value: EC2-Instance-2
  EC2Instance3:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-0841b1152f02fa85e
      InstanceType: t3.micro
      KeyName: !Ref KeyName
      SubnetId: subnet-0cc35aea439788393
      SecurityGroupIds:
        - !Ref InstanceSecurityGroup
      Tags:
        - Key: Name
          Value: EC2-Instance-3
Outputs:
  InstanceIDs:
    Description: IDs of all EC2 instances
    Value: !Join [ ",", [ !Ref EC2Instance1, !Ref EC2Instance2, !Ref EC2Instance3 ] ]

```

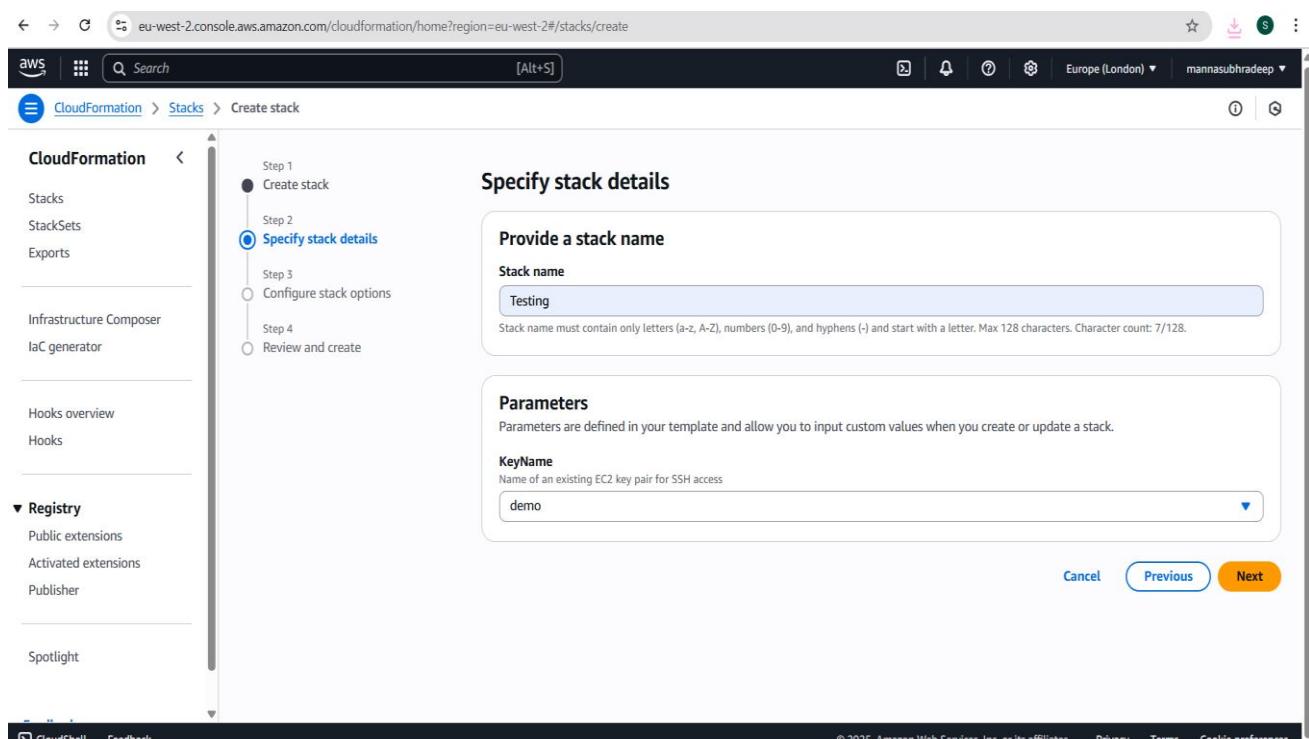
Replace subnet ID, VPC ID, Gateway ID with a valid ones in your VPC.

3. Create the stack

- Give a name like EC2AutoDeleteStack
- Select your **Key Pair**
- Click **Next** → Leave options default → **Create Stack**



The screenshot shows the 'CloudFormation' service in the AWS Management Console. On the left, the navigation pane includes 'Stacks', 'StackSets', 'Exports', 'Infrastructure Composer', 'IaC generator', 'Hooks overview', 'Hooks', and a 'Registry' section with 'Public extensions', 'Activated extensions', and 'Publisher'. A 'Spotlight' section is also present. The main area is titled 'Step 4 Review and create'. It contains two tabs: 'Choose an existing template' (selected) and 'Build from Infrastructure Composer'. Below this is a 'Specify template' section with a link to a GitHub repository. It includes fields for 'Template source': 'Amazon S3 URL' (radio button), 'Upload a template file' (radio button selected, showing 'CF for 3ECyaml'), and 'Sync from Git'. An 'Upload a template file' button is shown with 'CF for 3ECyaml' selected. A 'S3 URL' field contains a link to an S3 bucket. A 'View in Infrastructure Composer' button is also present. At the bottom right are 'Cancel', 'Previous', and 'Next' buttons.



The screenshot shows the 'CloudFormation' service in the AWS Management Console. The left navigation pane is identical to the previous screenshot. The main area is titled 'Step 1 Create stack'. It shows a progress bar with 'Step 1 Create stack' (filled), 'Step 2 Specify stack details' (selected), 'Step 3 Configure stack options', and 'Step 4 Review and create'. The 'Specify stack details' section is expanded, showing a 'Provide a stack name' field with 'Testing' entered. A note states: 'Stack name must contain only letters (a-z, A-Z), numbers (0-9), and hyphens (-) and start with a letter. Max 128 characters. Character count: 7/128.' Below this is a 'Parameters' section with a note: 'Parameters are defined in your template and allow you to input custom values when you create or update a stack.' A 'KeyName' field is shown with 'demo' entered. At the bottom right are 'Cancel', 'Previous', and 'Next' buttons.

The screenshot shows the AWS CloudFormation console with the 'Testing' stack selected. The 'Events' tab is active, displaying 17 events. All events are of type 'CREATE_COMPLETE' and occurred on 2025-07-15 at 22:04:09 UTC+0530. The events are listed as follows:

Timestamp	Logical ID	Status	Detailed status
2025-07-15 22:04:10 UTC+0530	Testing	CREATE_COMPLETE	-
2025-07-15 22:04:09 UTC+0530	EC2Instance2	CREATE_COMPLETE	-
2025-07-15 22:04:09 UTC+0530	EC2Instance1	CREATE_COMPLETE	-
2025-07-15 22:04:09 UTC+0530	EC2Instance3	CREATE_COMPLETE	-
2025-07-15 22:04:09 UTC+0530		CREATE_IN_PROGRESS	-

Step 2: Stop the EC2 Instances

After the stack is created:

1. Go to **EC2 Console**.
2. Select all 3 instances → Click "Instance State" → **Stop instance**.
3. Wait for them to fully enter the "**stopped**" state.

The screenshot shows the AWS EC2 Instances page. Three instances are listed: EC2-Instance-2, EC2-Instance-3, and EC2-Instance-1. All three instances are in a 'Stopping' state. A green success message at the top left states: "Successfully initiated stopping of i-0404624cfa2abe97f, i-070071b2831979e7e, i-0cb9d7ab0a97c". Below the instances, a message says "3 instances selected". The monitoring dashboard shows CPU utilization, Network in, Network out, and Network packets in for the selected instances.

Step 3: Create the Lambda Function

1. Go to AWS Console → Search Lambda → Click “Create Function”

- **Name:** DeleteStoppedEC2
- **Runtime:** Python 3.9
- **Permissions:** Choose **Create a new role with basic Lambda permission**

The screenshot shows the 'Create function' wizard in the AWS Lambda console. The top navigation bar shows 'Lambda > Functions > Create function'. The main section is titled 'Create function' with a 'Info' link. It says 'Choose one of the following options to create your function.' Three options are available: 'Author from scratch' (selected), 'Use a blueprint', and 'Container image'. The 'Basic information' section includes fields for 'Function name' (set to 'DeleteEC2'), 'Runtime' (set to 'Python 3.9'), and 'Architecture' (set to 'x86_64'). A note at the bottom states: 'By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.'

Lambda > Functions > Create function

Create function Info

Choose one of the following options to create your function.

Author from scratch
Start with a simple Hello World example.

Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.

Container image
Select a container image to deploy for your function.

Basic information

Function name
Enter a name that describes the purpose of your function.
DeleteEC2

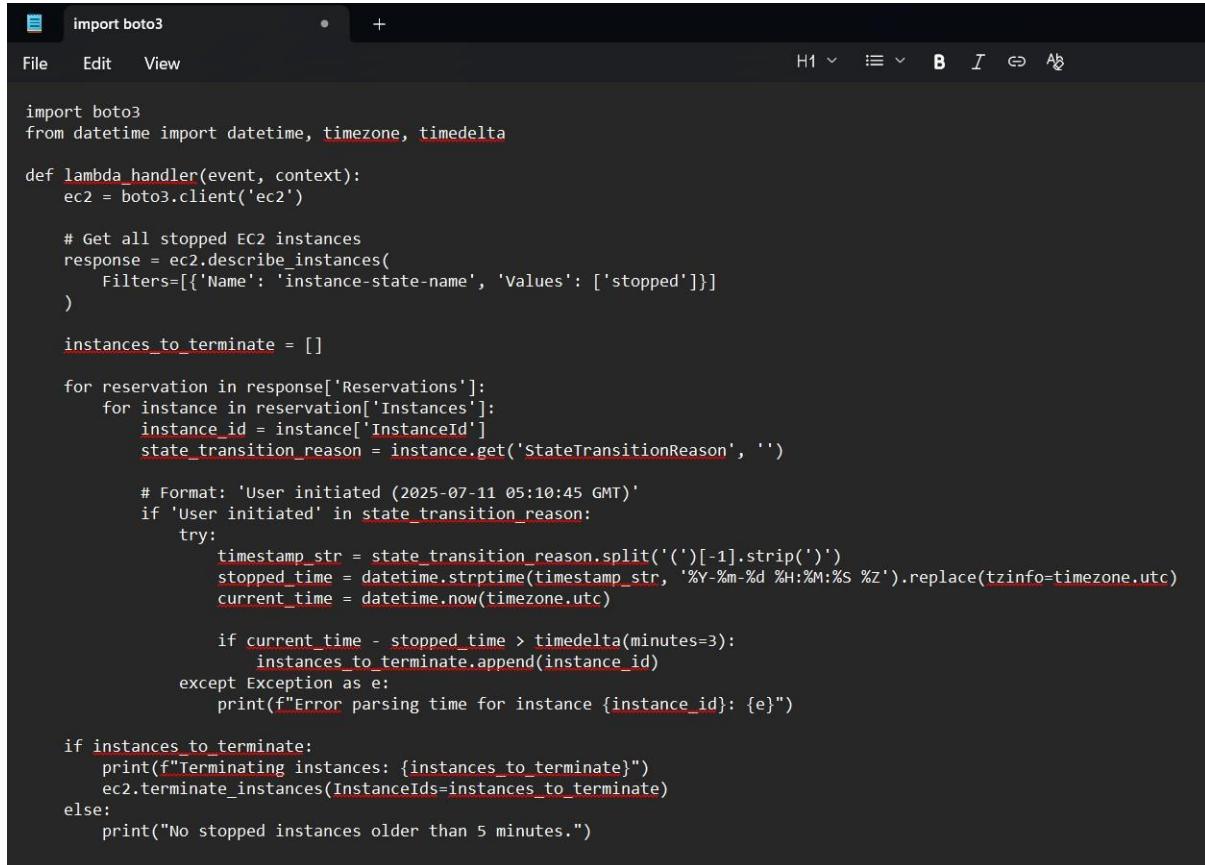
Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.9

Architecture Info
Choose the instruction set architecture you want for your function code.
 arm64
 x86_64

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

2. After function is created, open the Code tab and paste this:



```
import boto3
from datetime import datetime, timezone, timedelta

def lambda_handler(event, context):
    ec2 = boto3.client('ec2')

    # Get all stopped EC2 instances
    response = ec2.describe_instances(
        Filters=[{'Name': 'instance-state-name', 'Values': ['stopped']}]
    )

    instances_to_terminate = []

    for reservation in response['Reservations']:
        for instance in reservation['Instances']:
            instance_id = instance['InstanceId']
            state_transition_reason = instance.get('StateTransitionReason', '')

            # Format: 'User initiated (2025-07-11 05:10:45 GMT)'
            if 'User initiated' in state_transition_reason:
                try:
                    timestamp_str = state_transition_reason.split('(')[-1].strip(')')
                    stopped_time = datetime.strptime(timestamp_str, '%Y-%m-%d %H:%M:%S %z').replace(tzinfo=timezone.utc)
                    current_time = datetime.now(timezone.utc)

                    if current_time - stopped_time > timedelta(minutes=3):
                        instances_to_terminate.append(instance_id)
                except Exception as e:
                    print(f"Error parsing time for instance {instance_id}: {e}")

    if instances_to_terminate:
        print(f"Terminating instances: {instances_to_terminate}")
        ec2.terminate_instances(InstanceIds=instances_to_terminate)
    else:
        print("No stopped instances older than 5 minutes.")
```

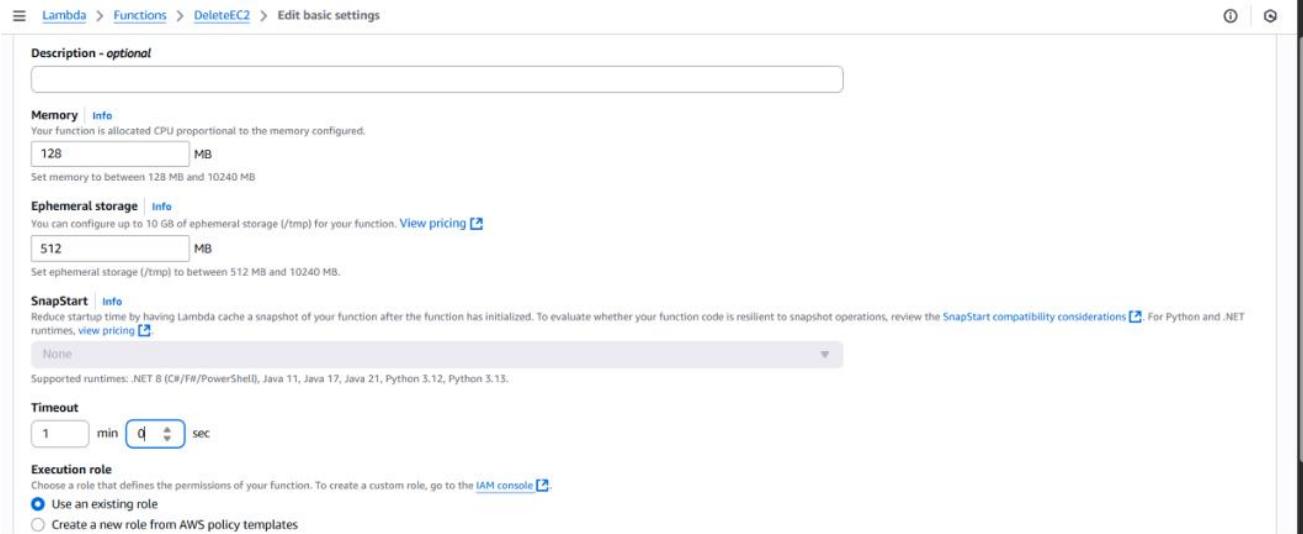
3. Click Deploy

Step 4: Increase Timeout

1. Go to Configuration → General configuration

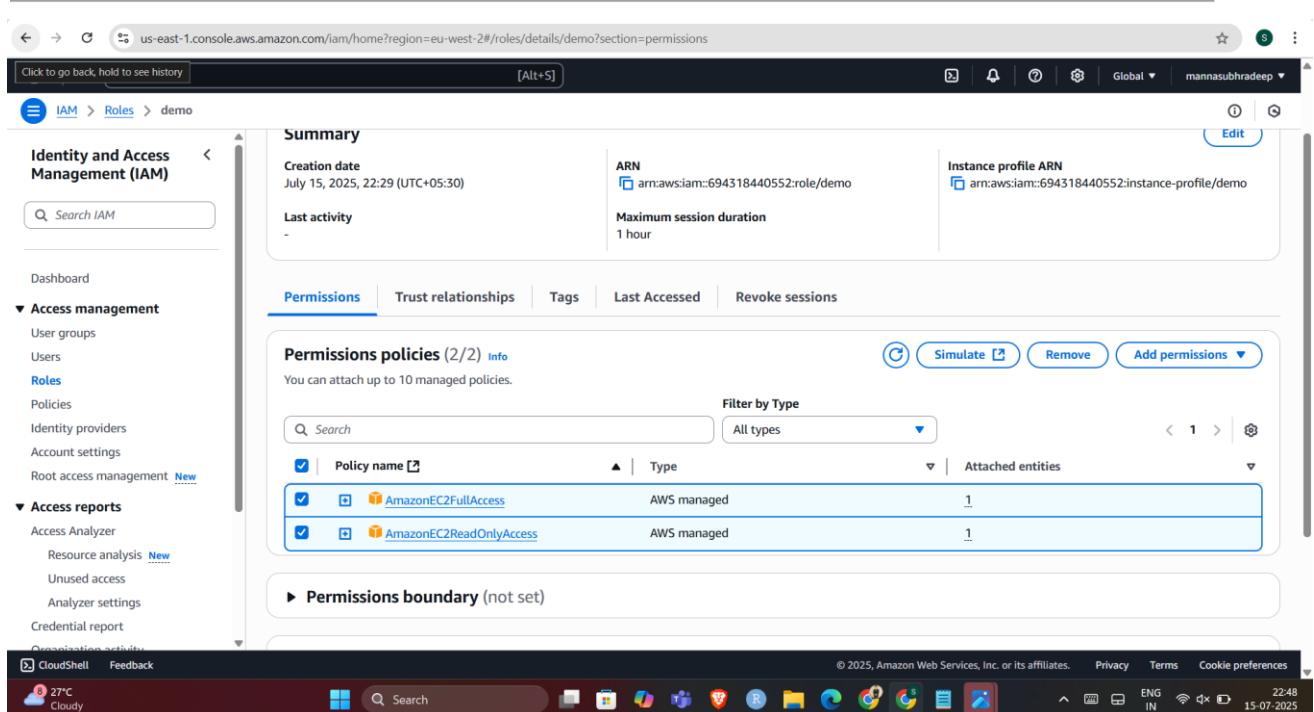
2. Click Edit → Set Timeout = 1 minute

3. Click Save



Step 5: Add EC2 Permissions to Lambda

1. Go to Permissions tab → Click IAM Role name
2. Click Attach policies
3. Search and attach:
 - **AmazonEC2ReadOnlyAccess**
 - **AmazonEC2FullAccess (or create custom policy)**



Step 6: Manually Test the Lambda Function

1. Go to Code tab → Click Test
2. Configure a new test event (leave JSON empty {})

3. Click Test

You will see:

- Terminated: ['i-xxxx'] if any instance is older than 5 mins
- Or "No instances found for termination."

The screenshot shows the AWS Lambda function editor for the 'DeleteStoppedEC2' function. The code in 'lambda_function.py' is as follows:

```
def lambda_handler(event, context):
    for reservation in response['Reservations']:
        if instances_to_terminate:
            print(f"Terminating instances: {instances_to_terminate}")
            ec2.terminate_instances(InstanceIds=instances_to_terminate)
        else:
            print("No stopped instances older than 5 minutes.")

Amazon Q Tip 1/3: Start typing to get suggestions ([ESC] to exit)
```

The sidebar shows deployment options like 'Deploy' and 'Test'. The 'TEST EVENTS' section is expanded, showing a saved event named 'subhra'. The 'ENVIRONMENT VARIABLES' section is also visible.

At the bottom, the 'Code properties' section shows details like package size, SHA256 hash, and last modified time. The browser status bar indicates it's 22:44 on 15-07-2025, with network and battery status.

💡 Output

- All EC2s that were stopped for more than 5 minutes will be automatically deleted.
- Logs show which were terminated.
- You save cost and keep your environment clean!

The screenshot shows the AWS EC2 Instances page. The left sidebar is collapsed, and the main content area displays a table of instances. The table has columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IP. Three instances are listed, all of which are terminated:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
EC2-Instance-2	i-0404624cfa2abe97f	Terminated	t3.micro	-	View alarms +	eu-west-2c	-
EC2-Instance-3	i-070071b2831979e7e	Terminated	t3.micro	-	View alarms +	eu-west-2c	-
EC2-Instance-1	i-0cb9d7a9d7ab0a97c	Terminated	t3.micro	-	View alarms +	eu-west-2c	-

Below the table, there is a section titled "Select an instance" with a dropdown menu. The bottom of the screen shows the Windows taskbar with various pinned icons and the date/time (15-07-2025).