



Intelligent Weather Alert System for Airlines Using AWS IAM, EC2 & SNS



Project Overview

Imagine an airline receiving real-time weather alerts before extreme conditions strike!
💡 What if you could automate this using AWS?

This project automates weather alerts using a Bash script on an AWS EC2 instance. The script fetches weather data, checks conditions, and triggers SNS alerts for freezing temperatures, ensuring timely notifications for necessary actions, helping airlines make informed decisions to enhance passenger safety.



Step-by-Step Guide

1 Create an IAM Role for EC2

- Navigate to IAM Console and create a new role.

The screenshot shows the AWS IAM Roles page. On the left, there's a navigation sidebar with 'Identity and Access Management (IAM)' selected. Under 'Access management', 'Roles' is also selected. The main area displays a table titled 'Roles (5)'. The table has columns for 'Role name', 'Trusted entities', and 'Last activity'. The roles listed are: 'AWSServiceRoleForAutoScaling', 'AWSServiceRoleForElasticLoadBalancing', 'AWSServiceRoleForSupport', 'AWSServiceRoleForTrustedAdvisor', and 'AWSServiceRoleForVPCTransitGateway'. A red box highlights the 'Create role' button at the top right of the table area.

- Assign EC2 and SNS , Cloudwatch fullaccess permissions to the role.

The screenshot shows the 'Add permissions' step of the IAM Create Role wizard. On the left, a vertical navigation bar lists steps: 'Step 1 Select trusted entity', 'Step 2 Add permissions' (which is currently selected), and 'Step 3 Name, review, and create'. The main area is titled 'Add permissions' and shows a table of 'Permissions policies (2/1039)'. One policy, 'AmazonEC2FullAccess', is selected and highlighted with a red box. The table includes columns for 'Policy name', 'Type', and 'Description'. A 'Filter by Type' dropdown is visible above the table. At the bottom, there's a note '▶ Set permissions boundary - optional' and a 'Next' button.

- Give name to role and complete the role creation.

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.
WeatherAlertEC2Role

Description
Add a short explanation for this role.
Allows EC2 instances to call AWS services on your behalf.

Step 1: Select trusted entities

Step 2: Add permissions

Permissions policy summary

Policy name	Type	Attached as
AmazonEC2FullAccess	AWS managed	Permissions policy
AmazonSNSFullAccess	AWS managed	Permissions policy

Step 3: Add tags

Add tags - optional
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.
No tags associated with the resource.

Create role

2 Launch an EC2 Instance & Attach IAM Role

- Open AWS Console and navigate to EC2.
- Click on Launch Instance > Give name to Instance>Select OS

Name and tags

Name
weather-data

Application and OS Images (Amazon Machine Image)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below.

Quick Start

Amazon Linux	macOS	Ubuntu	Windows	Red Hat	SUSE Linux	Debian
aws	Mac	ubuntu®	Microsoft	Red Hat	SUSE	debian

Summary

Number of instances
1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.6.2...[read more](#)

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Launch instance

- Choose **Amazon Linux**, configure the instance type, and proceed.

Instance type [Info](#) | [Get advice](#)

Instance type

t2.micro	Free tier eligible
Family: t2 1 vCPU 1 GiB Memory Current generation: true	
On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour	
On-Demand Linux base pricing: 0.0116 USD per Hour	
On-Demand SUSE base pricing: 0.0116 USD per Hour	
On-Demand Windows base pricing: 0.0162 USD per Hour	
On-Demand RHEL base pricing: 0.026 USD per Hour	

Additional costs apply for AMIs with pre-installed software

All generations

[Compare instance types](#)

Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

[Create new key pair](#)

- Ensure security group settings allow **SSH and HTTP** access.

Network settings [Info](#)

Network [Info](#)

vpc-06f7a1d6b97526366

Subnet [Info](#)

No preference (Default subnet in any availability zone)

Auto-assign public IP [Info](#)

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group Select existing security group

We'll create a new security group called '**launch-wizard-1**' with the following rules:

<input checked="" type="checkbox"/> Allow SSH traffic from Helps you connect to your instance	Anywhere 0.0.0.0/0
<input type="checkbox"/> Allow HTTPS traffic from the internet To set up an endpoint, for example when creating a web server	
<input checked="" type="checkbox"/> Allow HTTP traffic from the internet To set up an endpoint, for example when creating a web server	

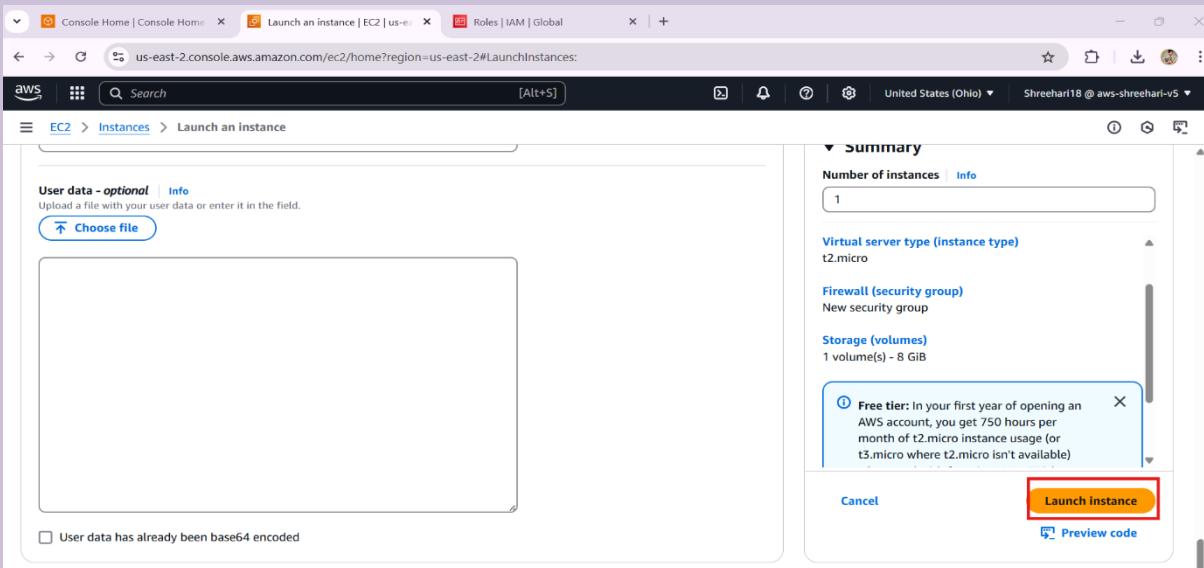
- In **Advanced Settings**, attach the **IAM Role** created earlier.

The screenshot shows the 'Configure storage' section of the AWS Lambda console. It displays a root volume configuration with 8 GiB of gp3 storage. A note indicates that free-tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. An 'Add new volume' button is present. Below this, a note says to click refresh to view backup information, mentioning Data Lifecycle Manager policies. A section for 'File systems' is shown with an 'Edit' button. At the bottom, a red box highlights the 'Advanced details' section, which includes fields for 'Domain join directory' (with a dropdown menu 'Select' and a 'Create new directory' button), 'IAM instance profile' (with a dropdown menu containing 'WeatherAlertEC2Role' and a link to 'Create new IAM profile'), 'Hostname type' (set to 'IP name'), and 'DNS Hostname' options for IPv4, IPv6, and AAAA records. The 'Instance auto-recovery' section is also visible at the bottom.

- Select the **IAM policy**

The screenshot shows the 'Launch an instance' wizard in the AWS EC2 console. It is on the 'Advanced details' step. The 'IAM instance profile' dropdown is highlighted with a red box and contains the value 'WeatherAlertEC2Role'. Other options in the dropdown include 'Create new IAM profile'. Below this, there are sections for 'Hostname type' (set to 'IP name'), 'DNS Hostname' (with checkboxes for IPv4, IPv6, and AAAA records), and 'Instance auto-recovery'. On the right side of the screen, a sidebar displays navigation links for 'Compute', 'Amazon VPC', 'Amazon S3', 'Amazon Lambda', 'Amazon RDS', 'Amazon Kinesis', 'Amazon CloudWatch Metrics', and 'Amazon CloudWatch Logs'.

- Click on Launch Instance



3 Install Required Dependencies on EC2

- Connect to the EC2 instance via SSH.
- Update packages and install essential tools.

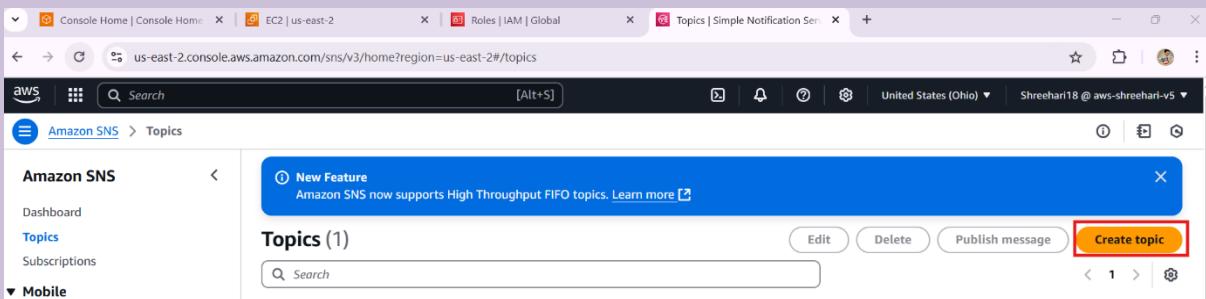
```
ec2-user@ip-172-31-19-158: ~
>Last login: Sun Mar 23 04:24:19 2025 from 117.99.246.26
[ec2-user@ip-172-31-19-158 ~]$ sudo yum update -y
sudo yum install -y jq curl
Last metadata expiration check: 0:36:52 ago on Sun Mar 23 04:04:11 2025.
Dependencies resolved.
Nothing to do.
Complete!
```

- Install pip and boto3 for AWS integration.

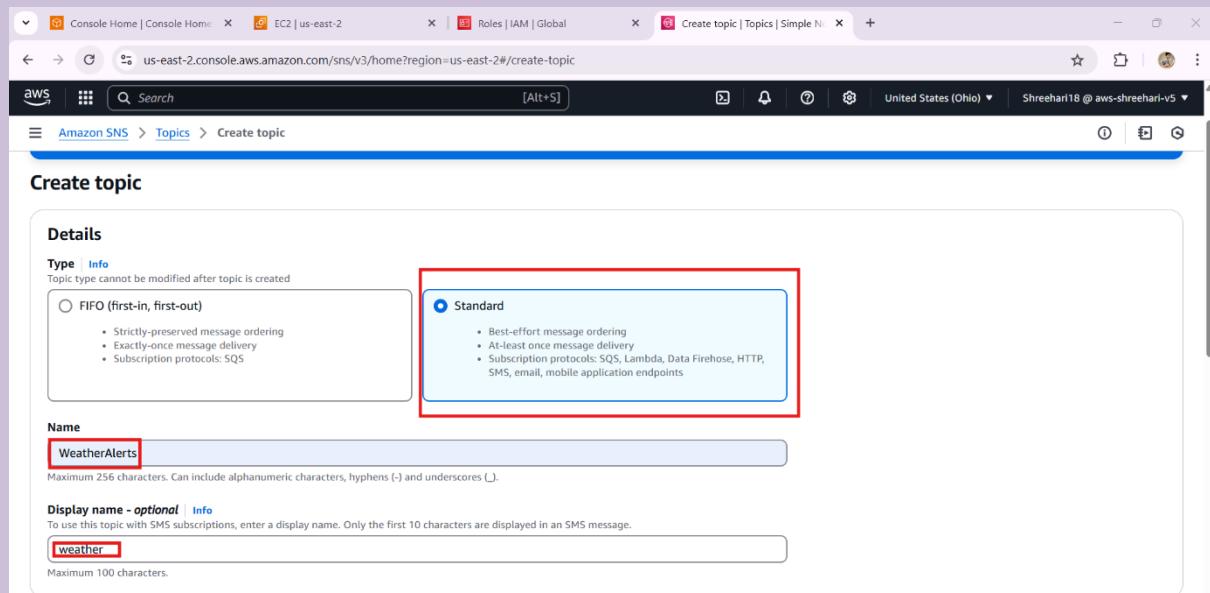
```
ec2-user@ip-172-31-19-158: ~
# Amazon Linux 2023
Amazon Linux 2023 Kernel Livepatch repository      137 kB/s |  14 kB     00:00
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-19-158 ~]$ sudo yum update -y
Amazon Linux 2023 Kernel Livepatch repository      137 kB/s |  14 kB     00:00
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-19-158 ~]$ sudo yum install -y python3 pip
Last metadata expiration check: 0:01:01 ago on Sun Mar 23 04:04:11 2025.
Package python3-3.9.20-1.amzn2023.0.3.x86_64 is already installed.
Dependencies resolved.
=====
Package          Architecture      Version       Repository
=====
Installing:
python3-pip      noarch        21.3.1-2.amzn2023.0.10   amazonl
Installing weak dependencies:
libxcrypt-compat x86_64        4.4.33-7.amzn2023   amazonl
Transaction Summary
```

4 Create an SNS Topic

- Open AWS SNS Console and create a new topic.

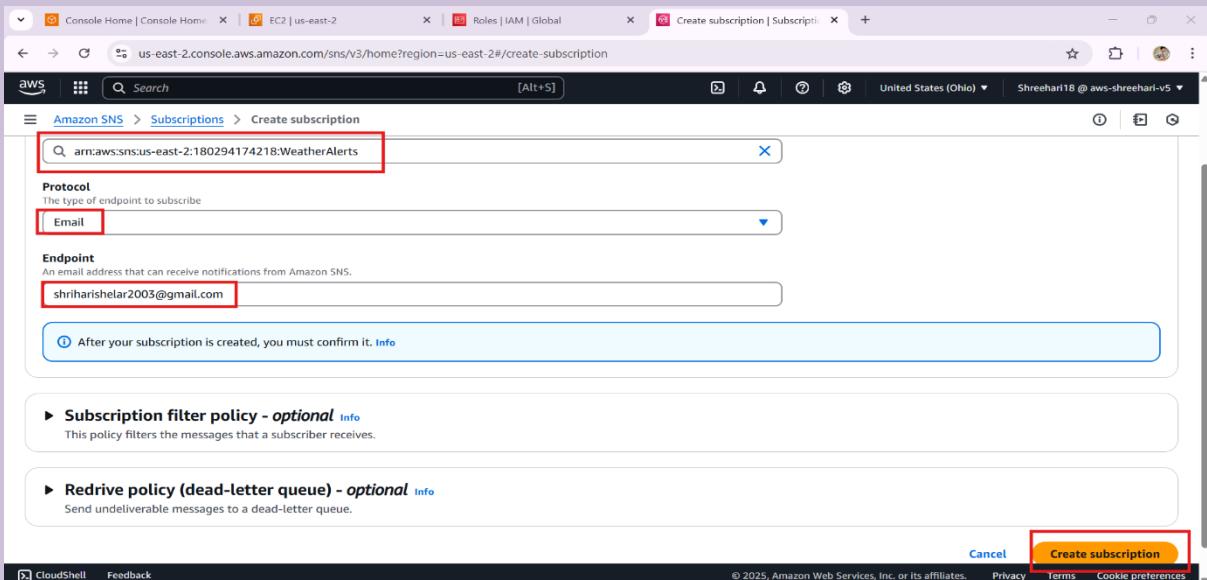


- Define a topic name and description > Click on create topic.

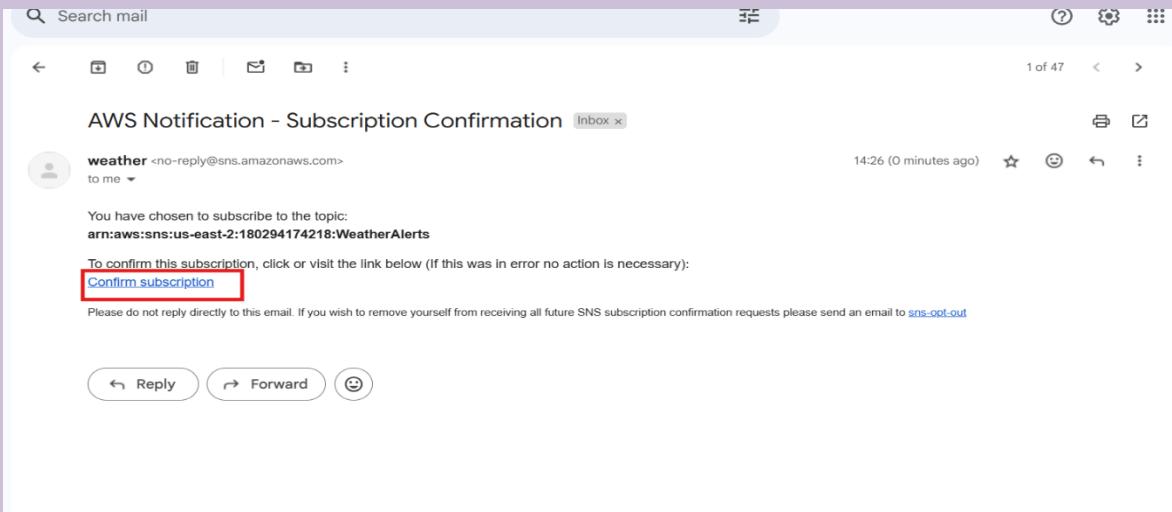


5 Subscribe to the SNS Topic

Create an email subscription to receive alert



- Confirm the subscription via email verification.



- Verify Subscriptions Status

```
[ec2-user@ip-172-31-19-158 ~]$ aws sns list-subscriptions-by-topic --topic-arn arn:aws:sns:us-east-2:180294174218:alters
{
  "Subscriptions": [
    {
      "SubscriptionArn": "arn:aws:sns:us-east-2:180294174218:alters:323d5393-cd46-4acd-acbf-01d80af27acc",
      "Owner": "180294174218",
      "Protocol": "email",
      "Endpoint": "xearic442@boyaga.com",
      "TopicArn": "arn:aws:sns:us-east-2:180294174218:alters"
    }
  ]
}
[ec2-user@ip-172-31-19-158 ~]$ aws sns publish --topic-arn arn:aws:sns:us-east-2:180294174218:alters --message "Test alert: SNS subscription check!"
{
  "MessageId": "51e0b090-a385-59ea-8429-d0c97e01c3ce"
}
[ec2-user@ip-172-31-19-158 ~]$ client_loop: send disconnect: Connection reset
PS C:\Users\Shrihari>
```

6 Deploy the Weather Alert Script

- Store the API key as an environment variable.

```
[ec2-user@ip-172-31-19-158 ~]$ echo 'export API_KEY="e0beb37cffb14eb0a0343806252303"' >> ~/.bashrc
[ec2-user@ip-172-31-19-158 ~]$ source ~/.bashrc
[ec2-user@ip-172-31-19-158 ~]$
```

- Upload and modify the Bash script (written for automating weather alerts).

```
GNU nano 5.8                                     weather_alert.sh
#!/bin/bash

# Configuration
CITY="New York"
API_KEY="e0beb37cffb14eb0a0343806252303"
CITY_ENCODED=$(echo "$CITY" | sed 's/ /%20/g')
LOG_FILE="weather_log.txt"

# Fetch weather data
RESPONSE=$(curl -s "http://api.weatherapi.com/v1/current.json?key=$API_KEY&q=$CITY_ENCODED")

# Debug: Log full API response
echo "$(date) 🌡️ Debug API Response: $RESPONSE" >> "$LOG_FILE"

# Check if API response is valid
if [[ -z "$RESPONSE" || "$RESPONSE" == *"error"* ]]; then
  echo "$(date) ⚠️ API request failed. Check API key or city name." | tee -a "$LOG_FILE"
  exit 1
fi

# Extract data using jq
TEMP_C=$(echo "$RESPONSE" | jq -r '.current.temp_c // empty')
TEMP_F=$(echo "$RESPONSE" | jq -r '.current.temp_f // empty')
CONDITION=$(echo "$RESPONSE" | jq -r '.current.condition.text // empty')

# Validate extracted values
if [[ -z "$TEMP_C" || "$TEMP_C" == "null" ]]; then
  echo "$(date) ⚠️ Failed to extract weather data." | tee -a "$LOG_FILE"
  exit 1
fi

# Display weather
echo "$(date) 🌡️ Weather in $CITY: $TEMP_C°C ($TEMP_F°F), $CONDITION" | tee -a "$LOG_FILE"

# Custom alerts based on temperature
if (( $(echo "$TEMP_C < 5" | bc -l) )); then
  echo "$(date) ❄️ Alert: It's freezing in $CITY! Temperature: $TEMP_C°C" | tee -a "$LOG_FILE"
```

- Test the script and validate the output.

```
[ec2-user@ip-172-31-19-158 ~]$ nano weather_alert.sh
[ec2-user@ip-172-31-19-158 ~]$ ./weather_alert.sh
Enter city name (default: New York): usa
🌐 Weather in usa: 10.1°C, Clear
```

7 Automate the Script Using Cron Jobs

- Configure `crontab` to run the script at regular intervals.

```
[ec2-user@ip-172-31-19-158 ~]$ crontab -e
-bash: crontab: command not found
[ec2-user@ip-172-31-19-158 ~]$ sudo yum install cronie -y
Last metadata expiration check: 1:39:18 ago on Sun Mar 23 04:04:11 2025.
Dependencies resolved.
=====
 Package          Arch      Version       Repository      Size
=====
 Installing:
  cronie          x86_64    1.5.7-1.amzn2023.0.2   amazonlinux    115 k
Installing dependencies:
  cronie-anacron  x86_64    1.5.7-1.amzn2023.0.2   amazonlinux    32 k
Transaction Summary
=====
```

- Run and check Status of `crontab`

```
[ec2-user@ip-172-31-19-158 ~]$ sudo systemctl start crond
[ec2-user@ip-172-31-19-158 ~]$ sudo systemctl enable crond
[ec2-user@ip-172-31-19-158 ~]$ sudo systemctl enable crond
[ec2-user@ip-172-31-19-158 ~]$ systemctl status crond
● crond.service - Command Scheduler
   Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled; preset: enabled)
   Active: active (running) since Sun 2025-03-23 05:43:47 UTC; 27s ago
     Main PID: 29213 (crond)
        Tasks: 1 (limit: 1111)
       Memory: 980.0K
          CPU: 8ms
         CGroup: /system.slice/crond.service
                   └─29213 /usr/sbin/crond -n

Mar 23 05:43:47 ip-172-31-19-158.us-east-2.compute.internal crond[29213]: (CRON) >
Mar 23 05:43:47 ip-172-31-19-158.us-east-2.compute.internal systemd[1]: Started c>
Mar 23 05:43:47 ip-172-31-19-158.us-east-2.compute.internal crond[29213]: (CRON) >
lines 1-15/15 (END)...skipping...
● crond.service - Command Scheduler
   Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled; preset: enabled)
   Active: active (running) since Sun 2025-03-23 05:43:47 UTC; 27s ago
     Main PID: 29213 (crond)
        Tasks: 1 (limit: 1111)
       Memory: 980.0K
          CPU: 8ms
         CGroup: /system.slice/crond.service
                   └─29213 /usr/sbin/crond -n
```

- Monitor scheduled executions and output logs.

```
[ec2-user@ip-172-31-19-158 ~]$ nano weather_alert.sh
[ec2-user@ip-172-31-19-158 ~]$ ./weather_alert.sh
Sun Mar 23 08:33:47 UTC 2025 🌐 Weather in New York: 1.1°C (34.0°F), Clear
Sun Mar 23 08:33:47 UTC 2025 ⚠ Alert: It's freezing in New York! Temperature: 1.1°C
[ec2-user@ip-172-31-19-158 ~]$ cat /home/ec2-user/weather.log
Enter city name (default: New York): Enter city name (default: New York): Sun Mar 23 06:48:06 UTC 2025 🌐 Weather in india: 30.1°C, Mist
Sun Mar 23 06:50:37 UTC 2025 🌐 Weather in india: 30.1°C, Mist
Sun Mar 23 06:51:05 UTC 2025 🌐 Weather in india: 30.1°C, Mist
Sun Mar 23 06:53:02 UTC 2025 🌐 Weather in india: 30.1°C, Mist
Sun Mar 23 06:55:13 UTC 2025 🌐 Weather in india: 30.1°C, Mist
Sun Mar 23 07:46:51 UTC 2025 🌐 Weather in New York: 2.2°C, Clear
Sun Mar 23 07:46:51 UTC 2025 ⚠ Alert: It's too cold in New York!
Sun Mar 23 07:47:35 UTC 2025 🌐 Weather in New York: 2.2°C, Clear
Sun Mar 23 07:47:35 UTC 2025 ⚠ Alert: It's too cold in New York!
Sun Mar 23 08:00:01 UTC 2025 🌐 Weather in New York: 2.2°C, Clear
Sun Mar 23 08:00:01 UTC 2025 ⚠ Alert: It's too cold in New York!
Sun Mar 23 08:10:20 UTC 2025 ✗ Failed to fetch weather data.
Sun Mar 23 08:11:45 UTC 2025 ✗ Failed to fetch weather data.
Sun Mar 23 08:26:19 UTC 2025 🌐 Weather in New York: 1.1°C, Clear
Sun Mar 23 08:26:19 UTC 2025 🌐 Weather in New York: 1.1°C, Clear
[ec2-user@ip-172-31-19-158 ~]$ crontab -e
crontab: installing new crontab
```

8 Receive Weather Alerts via SNS

- Verify SNS updates and **subscription status**.

The screenshot shows an email from AWS Notifications (no-reply@sns.amazonaws.com) with a red border. The subject is "AWS Notification Message". The body contains a test alert message and unsubscribe instructions.

AWS Notifications
no-reply@sns.amazonaws.com

Date: 23-03-2025 15:28:29

Subject: AWS Notification Message

Test alert: SNS subscription check!

--

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
<https://sns.us-east-2.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-2:180294174218:alters:323d5393-cd46-4acd-acbf-01d80af27acc&Endpoint=xecaric442@boyaga.com>

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at <https://aws.amazon.com/support>

- Test **SNS notifications** for freezing temperatures.

SENDER	SUBJECT	VIEW
• AWS Notifications no-reply@sns.amazonaws.com	AWS Notification Message	>
• AWS Notifications no-reply@sns.amazonaws.com	AWS Notification Message	>
• AWS Notifications no-reply@sns.amazonaws.com	AWS Notification - Subscription Confirmation	>

✓ Conclusion

🚀 This project successfully automates weather alerts using AWS services! With real-time data fetching and AWS SNS notifications, this setup can be extended to include **multiple conditions, locations, or even mobile push notifications for airline operations!**



- Would you build something like this? Let me know your thoughts! 🤜