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

🌟 Project Overview

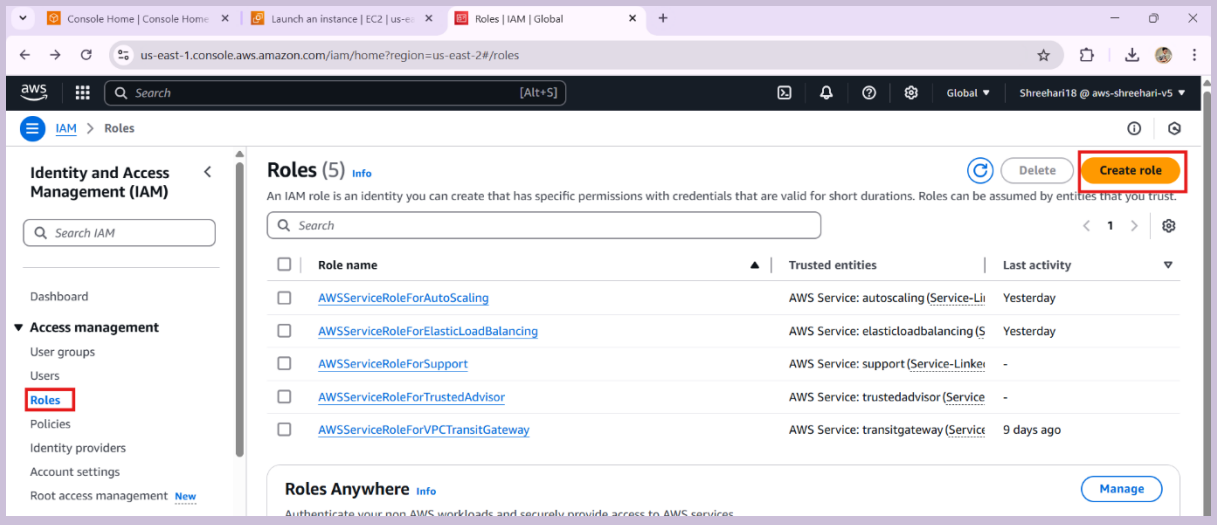
Imagine an airline receiving real-time weather alerts before extreme conditions strike!
✈️⚡ What if you could automate this using AWS?

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.

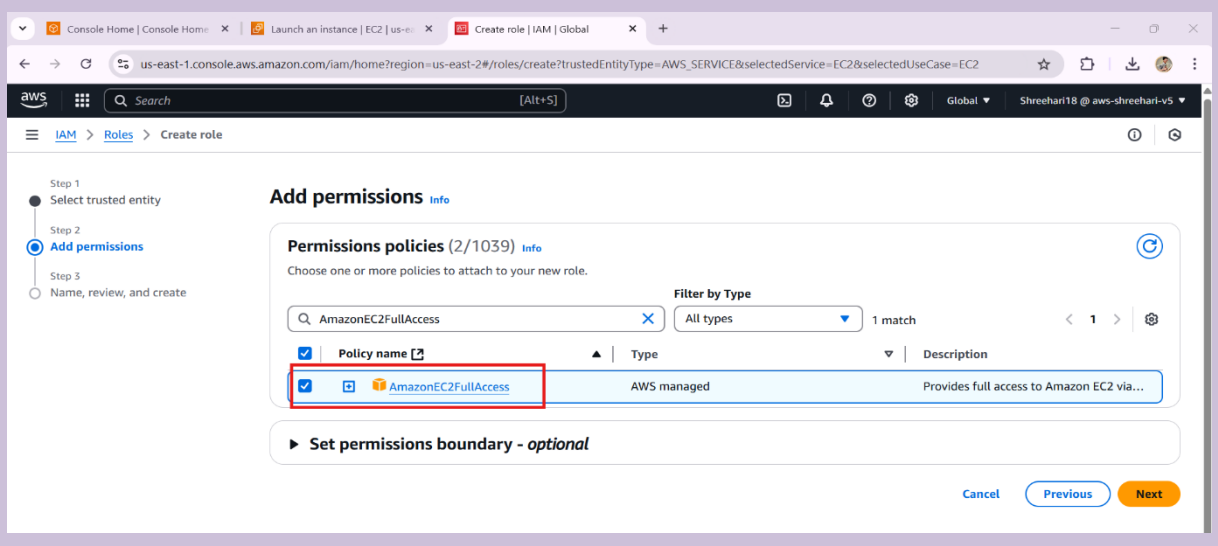
📌 Step-by-Step Guide

1️⃣ Create an IAM Role for EC2

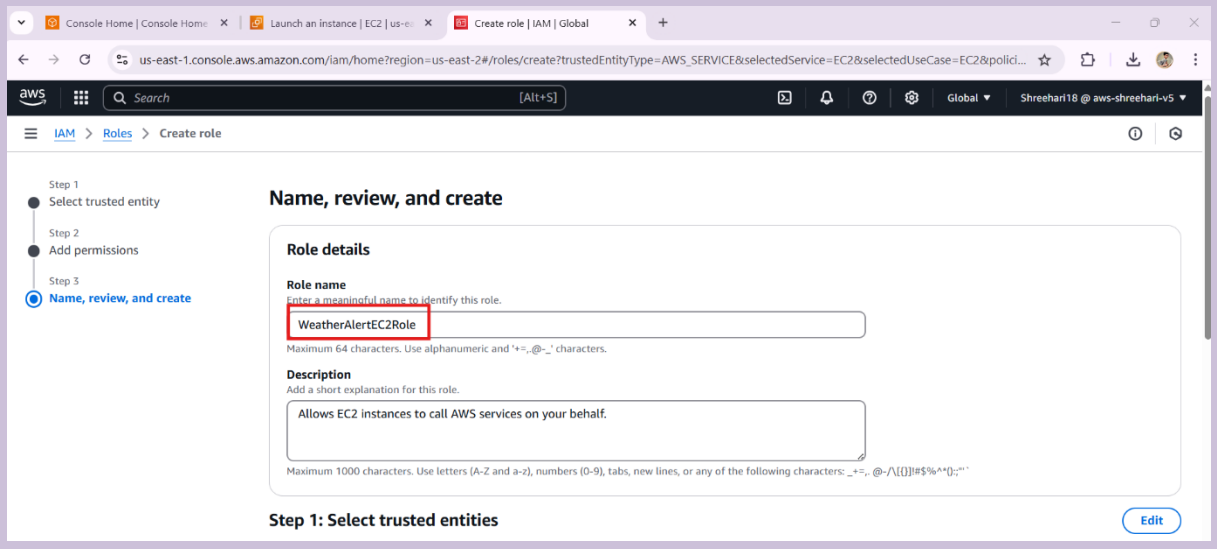
- Navigate to **IAM Console** and create a **new role**.

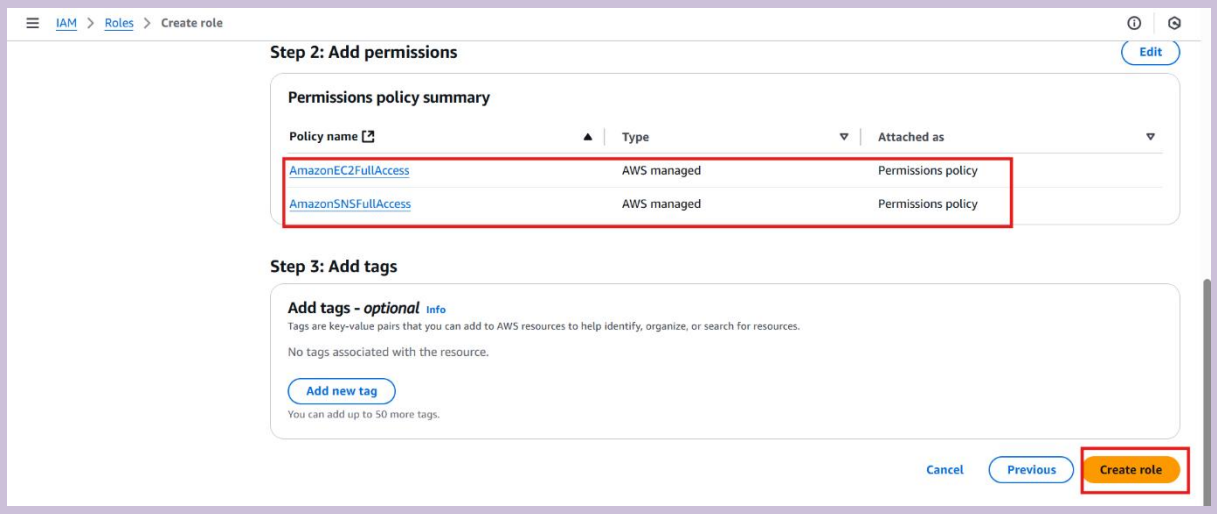


- Assign **EC2 and SNS , Cloudwatch fullaccess permissions** to the role.



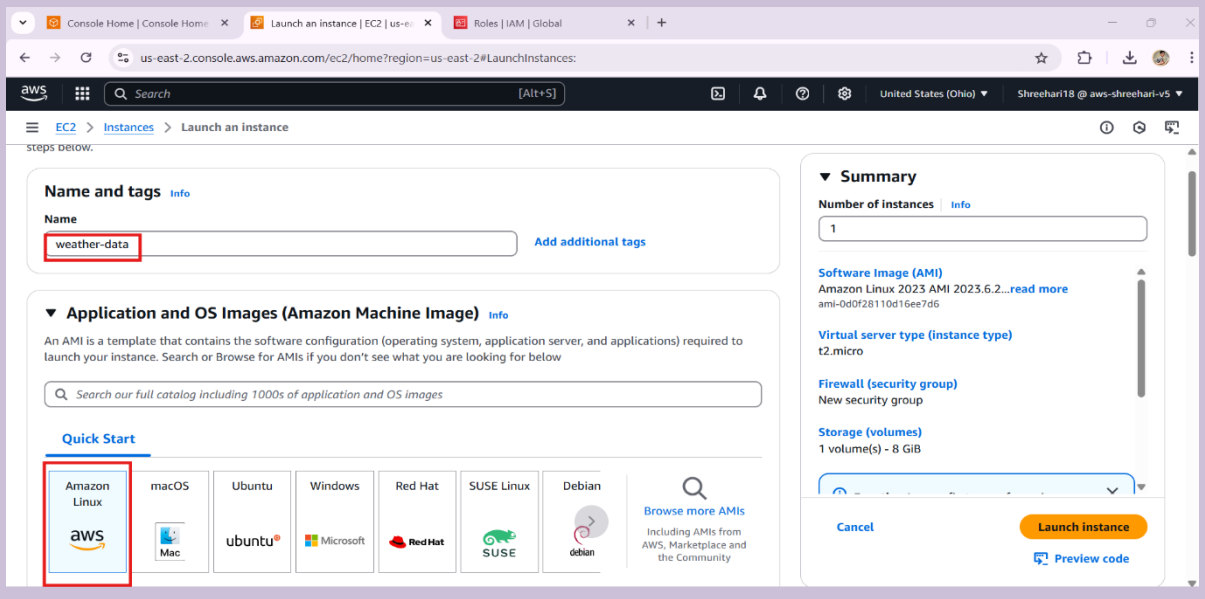
- Give name to role and complete the role creation.



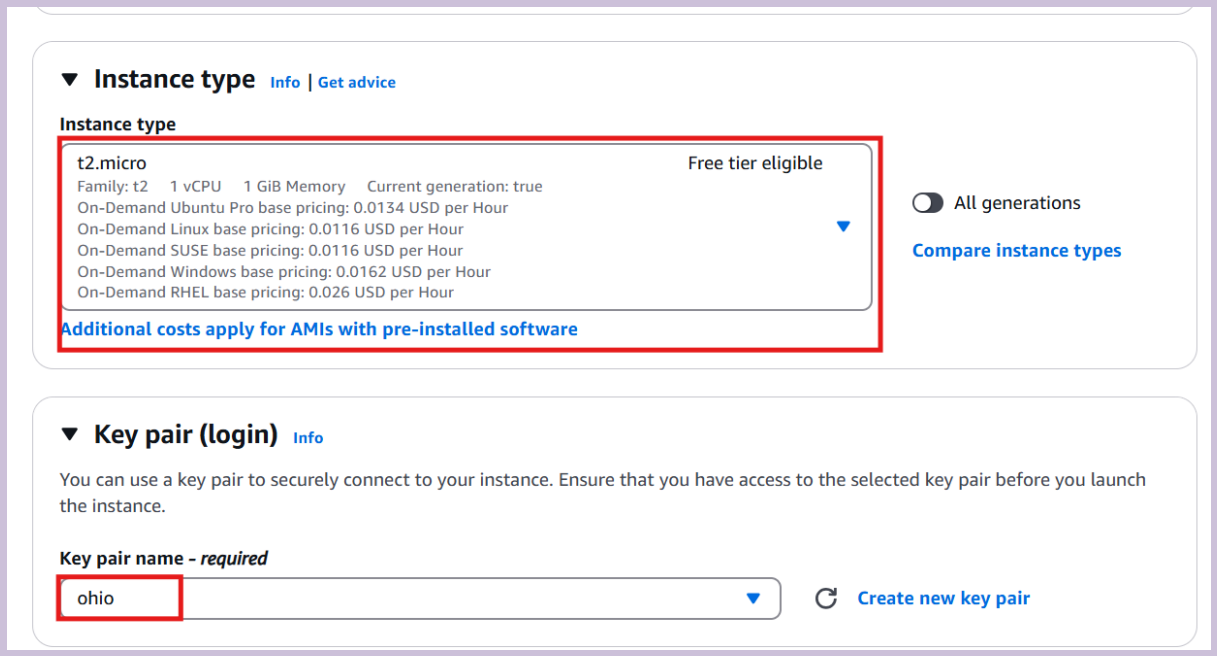


2 Launch an EC2 Instance & Attach IAM Role

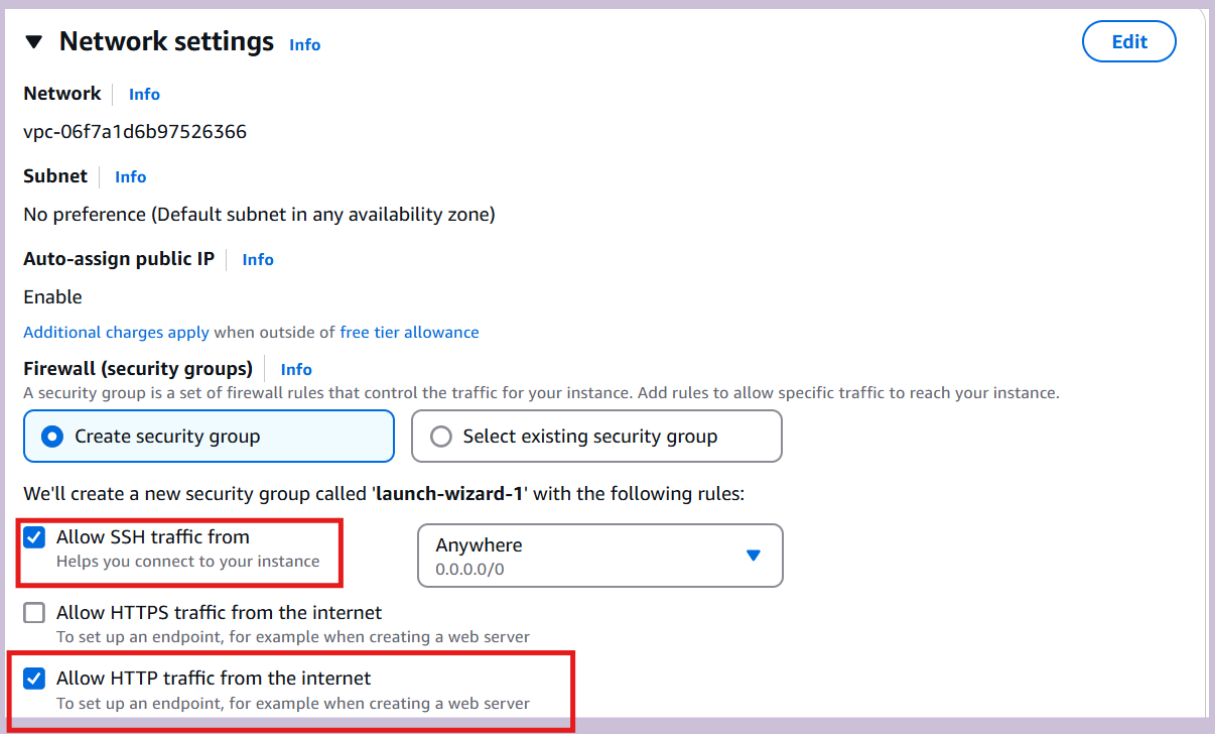
- Open **AWS Console** and navigate to **EC2**.
- Click on Launch Instance > Give name to Instance>Select OS



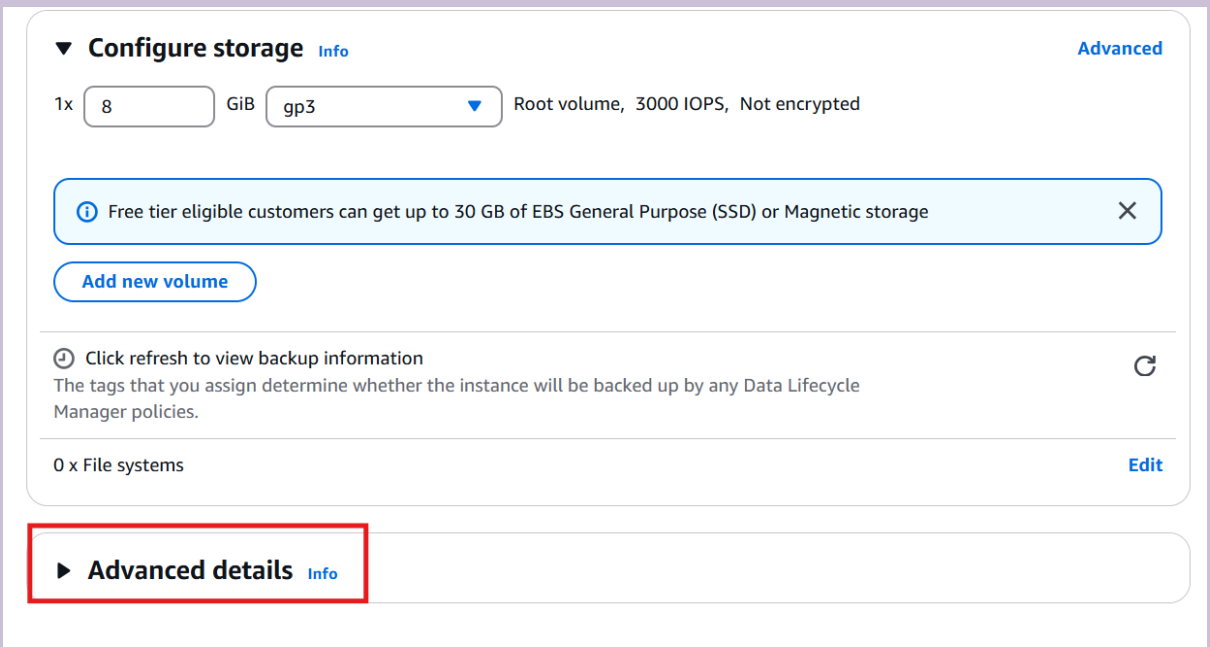
- Choose **Amazon Linux**, configure the instance type, and proceed.



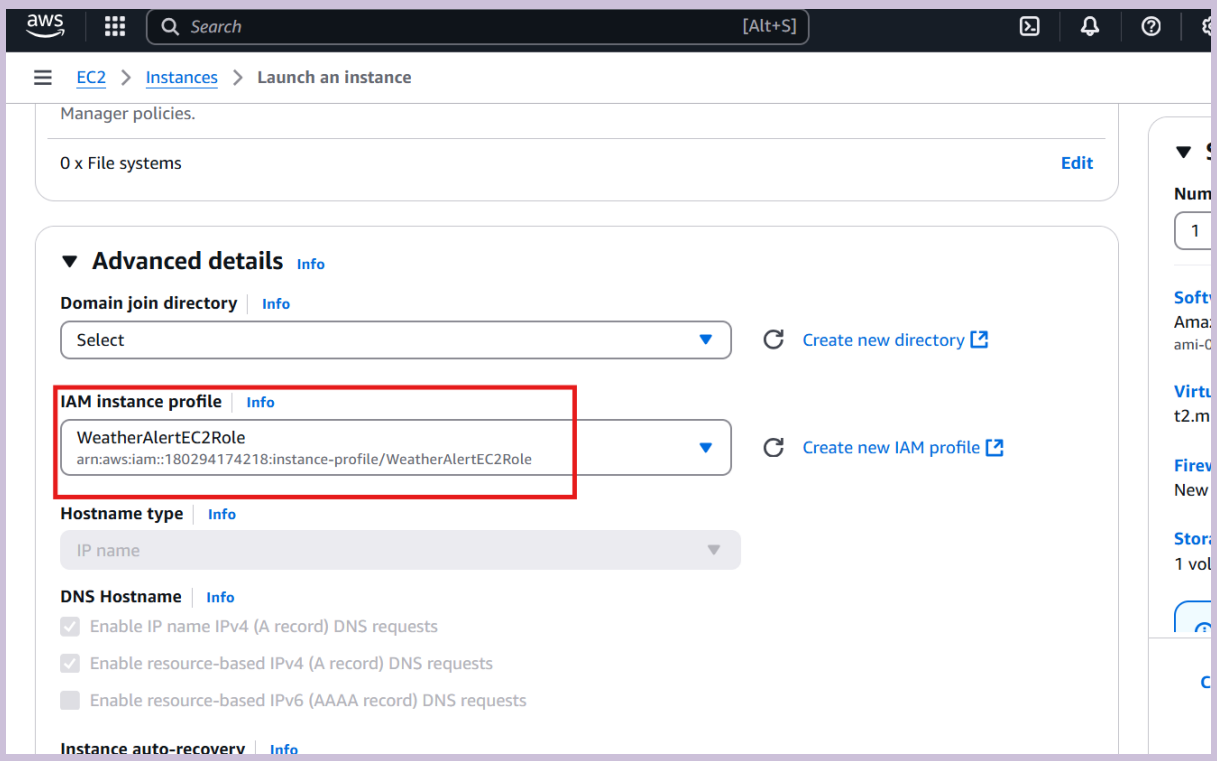
- Ensure security group settings allow **SSH and HTTP access**.



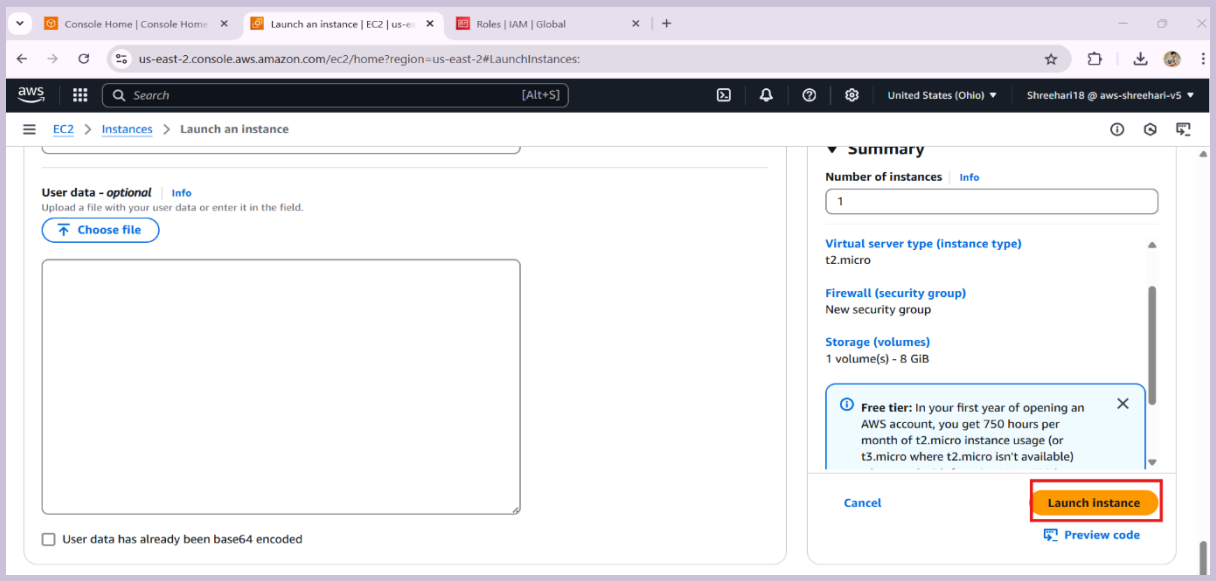
- In **Advanced Settings**, attach the **IAM Role** created earlier.



- **Select the IAM policy**

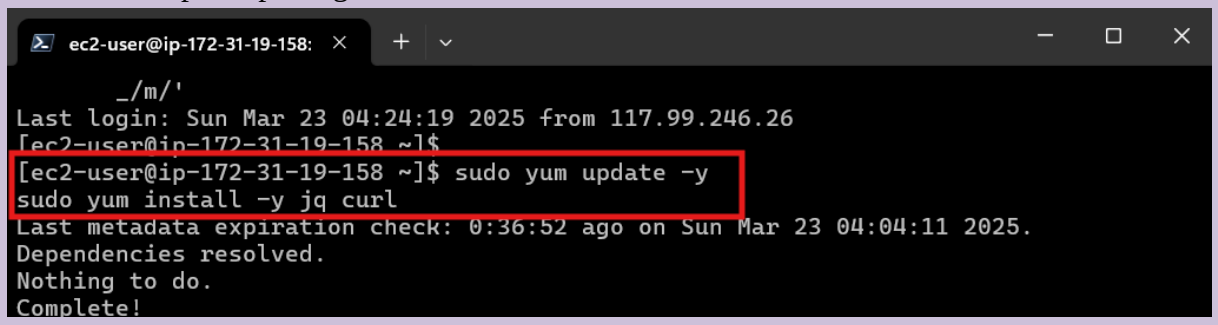


- **Click on Launch Instance**

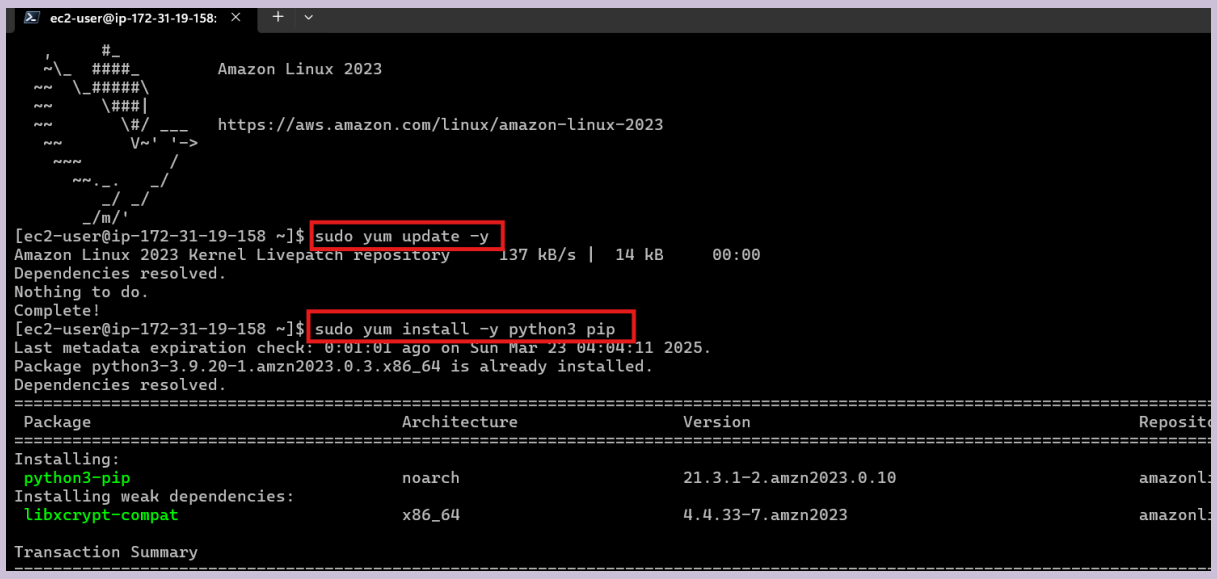


3 Install Required Dependencies on EC2

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

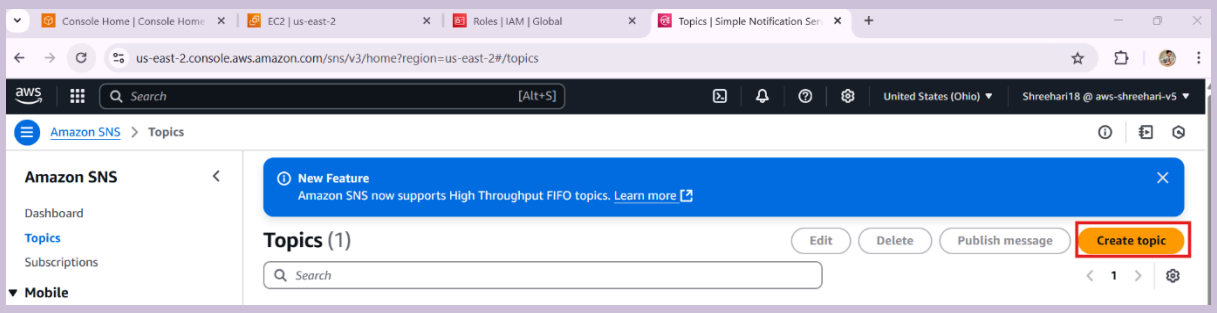


- Install **pip** and **boto3** for AWS integration.

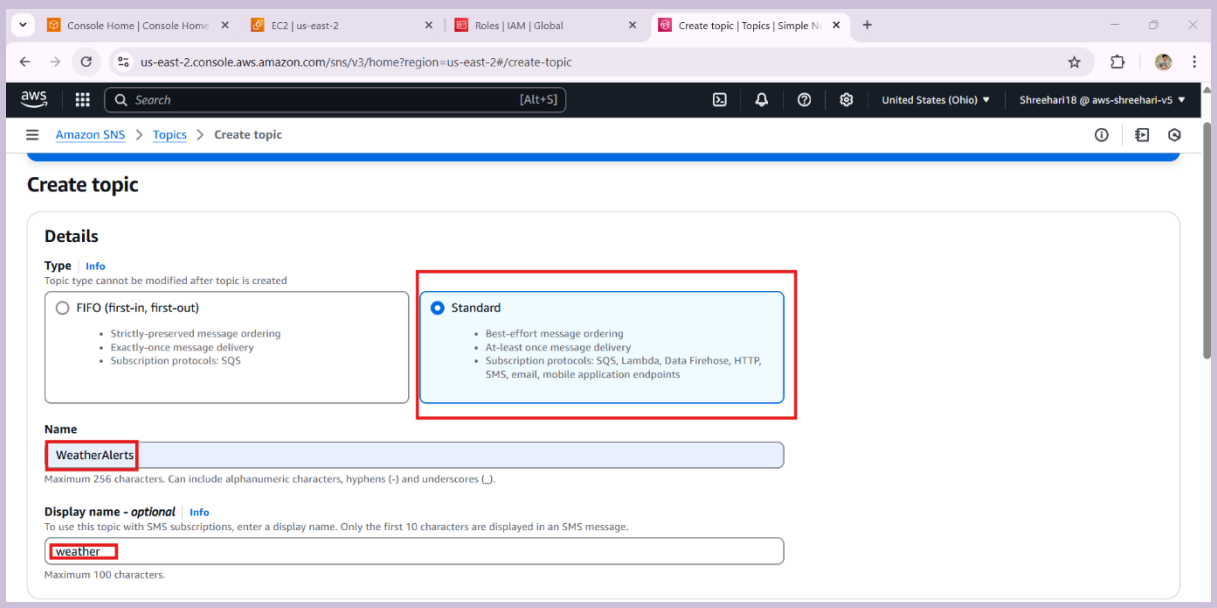


4 Create an SNS Topic

- Open **AWS SNS Console** and create a new topic.

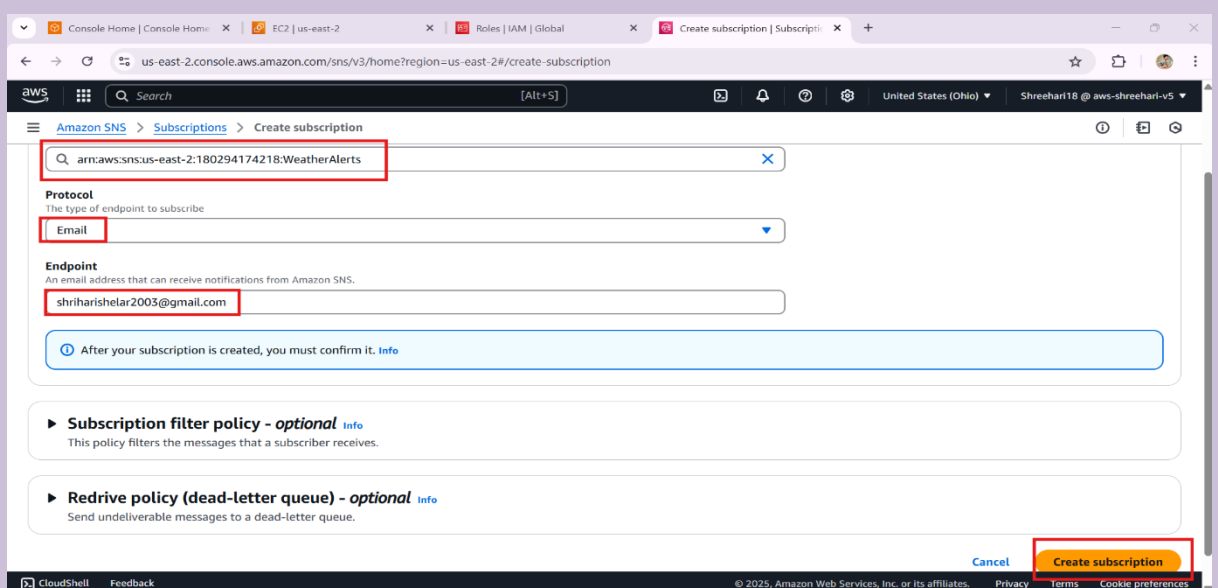


- Define a **topic name and description > Click on create topic**.

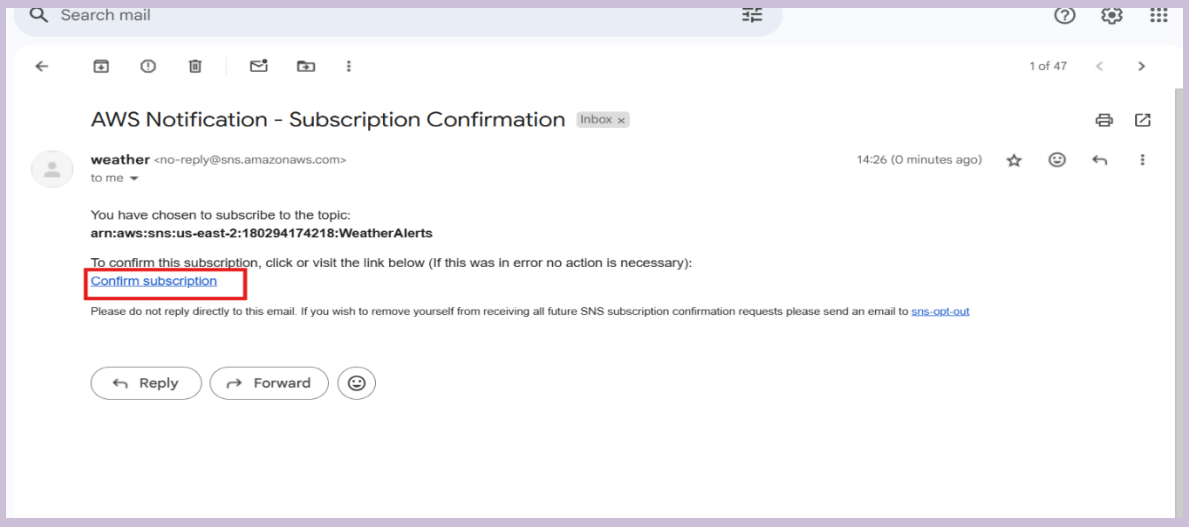


5 Subscribe to the SNS Topic

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": "xecaric442@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="e0beb37cffb14eb0a034380625230
3"' >> ~/.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)**.

```
ec2-user@ip-172-31-19-158:
  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 "$(date) ⚠️ API request failed. Check API key or city name." | tee -a "$OG_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
```

7️⃣ 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`**

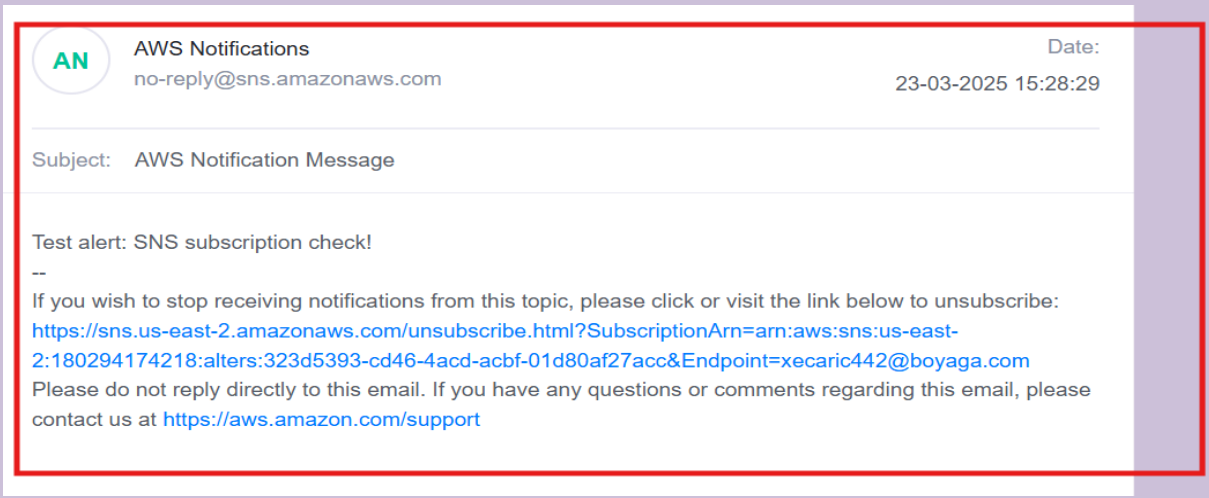
```
[ec2-user@ip-172-31-19-158 ~]$ sudo systemctl start crond
[ec2-user@ip-172-31-19-158 ~]$ sudo systemctl enable crond
[ec2-user@ip-172-31-19-158 ~]$ sudo systemctl enable crond
[ec2-user@ip-172-31-19-158 ~]$ systemctl status crond
● crond.service - Command Scheduler
     Loaded: loaded (/usr/lib/systemd/system/crond.service; enabled; preset: enab>
     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) >
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 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**.

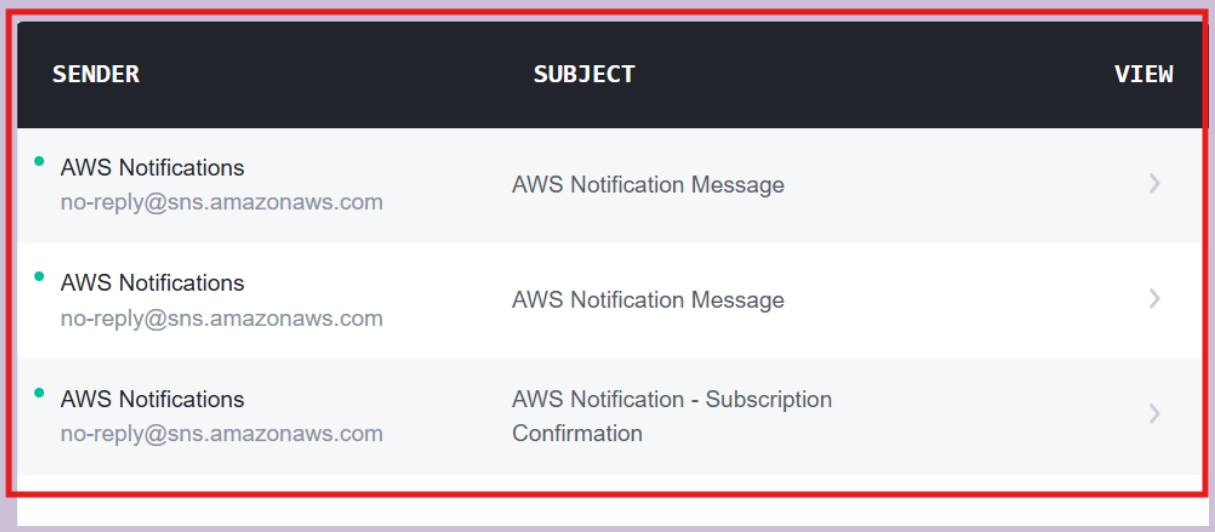
```
[ec2-user@ip-172-31-19-158 ~]$ ./weather_alert.sh
Sun Mar 23 08:33:47 UTC 2025 🌍 Weather in New York: 1.1°C (34.0°F), Clear
Sun Mar 23 08:33:47 UTC 2025 ❄️ Alert: It's freezing in New York! Temperature: 1.1°C
[ec2-user@ip-172-31-19-158 ~]$ cat /home/ec2-user/weather.log
Enter city name (default: 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
[ec2-user@ip-172-31-19-158 ~]$ crontab -e
crontab: installing new crontab
```

8️⃣ Receive Weather Alerts via SNS

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



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



☑️ 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!** 👇