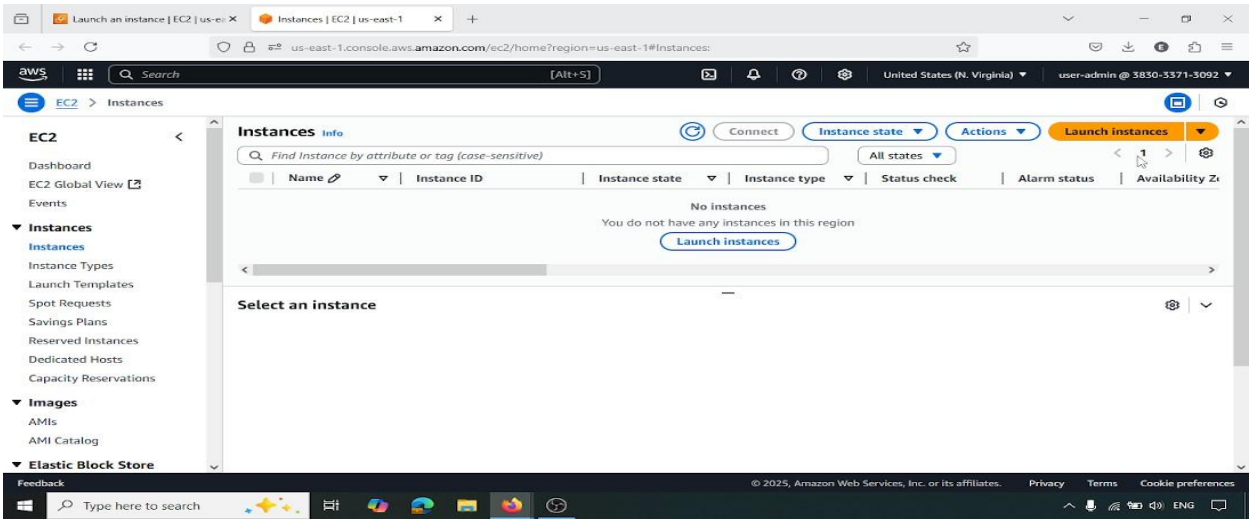
Overview of the Task:

The overall task involves the following steps:

1. Launching an EC2 Instance: This includes selecting an instance type, creating a key pair, configuring network settings, and adding a user data script to install and run NGINX.
2. Associating an Elastic IP: Allocating and associating a static public IP address with the EC2 instance to ensure consistent accessibility.
3. Configuring SNS Topics: Creating two Simple Notification Service (SNS) topics for alarms related to EC2 instance recovery and CPU performance.
4. Setting up CloudWatch Alarms: Creating CloudWatch alarms to monitor the EC2 instance. One alarm will trigger an instance recovery action upon a system check failure, and another will send a notification if CPU utilization exceeds a specified threshold.
5. Testing Alarms: Demonstrating the functionality of the configured alarms by stressing the CPU and observing the CloudWatch alarm state and email notifications.

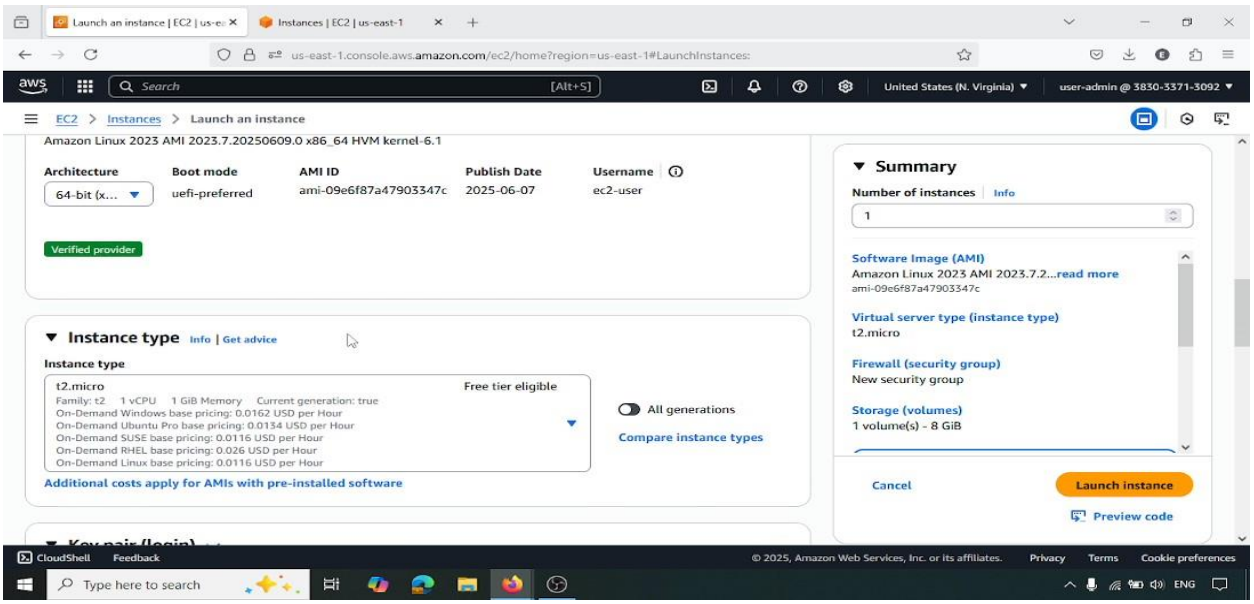
1. Launching an EC2 Instance

1:Launch Instance This screenshot shows the initial step within the AWS EC2 dashboard, where the "Launch instances" button is prominently displayed. Clicking this button initiates the guided process for creating and configuring a new virtual server in the cloud, setting the foundation for the NGINX deployment.



2: Choose Instance Type t2.micro

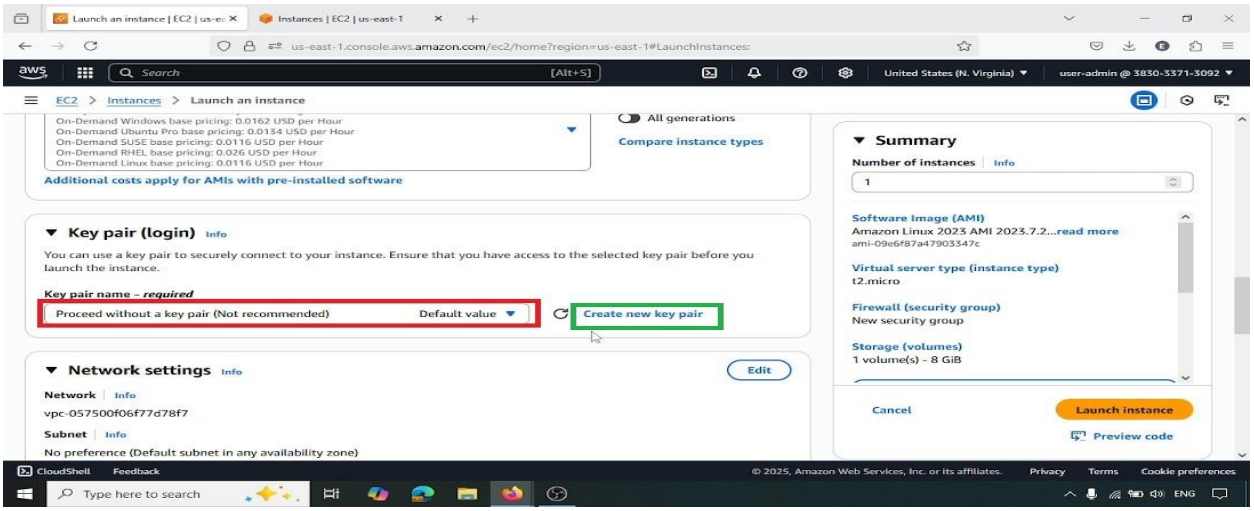
Here, the t2.micro instance type is selected. This choice is often preferred for light workloads and is frequently eligible under the AWS Free Tier, making it a cost-effective



BY : EbRam

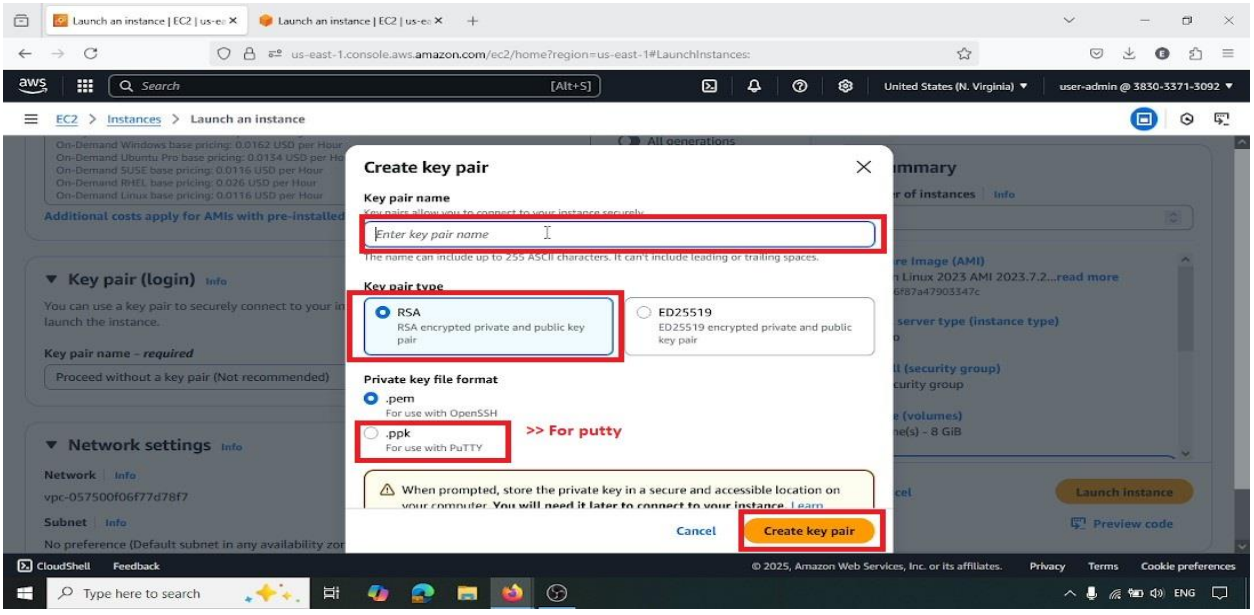
3: Choose Key Pair

highlights the crucial step of associating a key pair with the EC2 instance. A key pair, consisting of a public and private key, is essential for securely connecting to the instance via SSH after it has been launched, ensuring only authorized access.



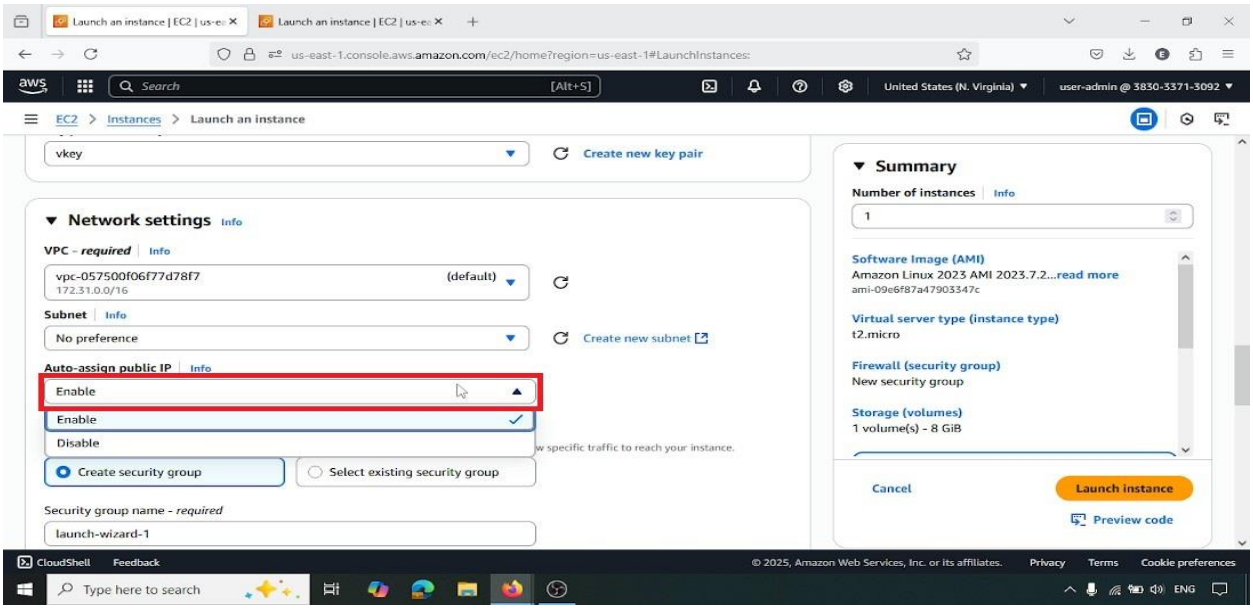
4: Create Key Pair

This screenshot illustrates the creation of a new key pair. Users are prompted to provide a name for their key pair, select the encryption type (e.g., RSA), and choose the private key file format (e.g., .pem for OpenSSH or .ppk for PuTTY), which is then downloaded for secure future use.

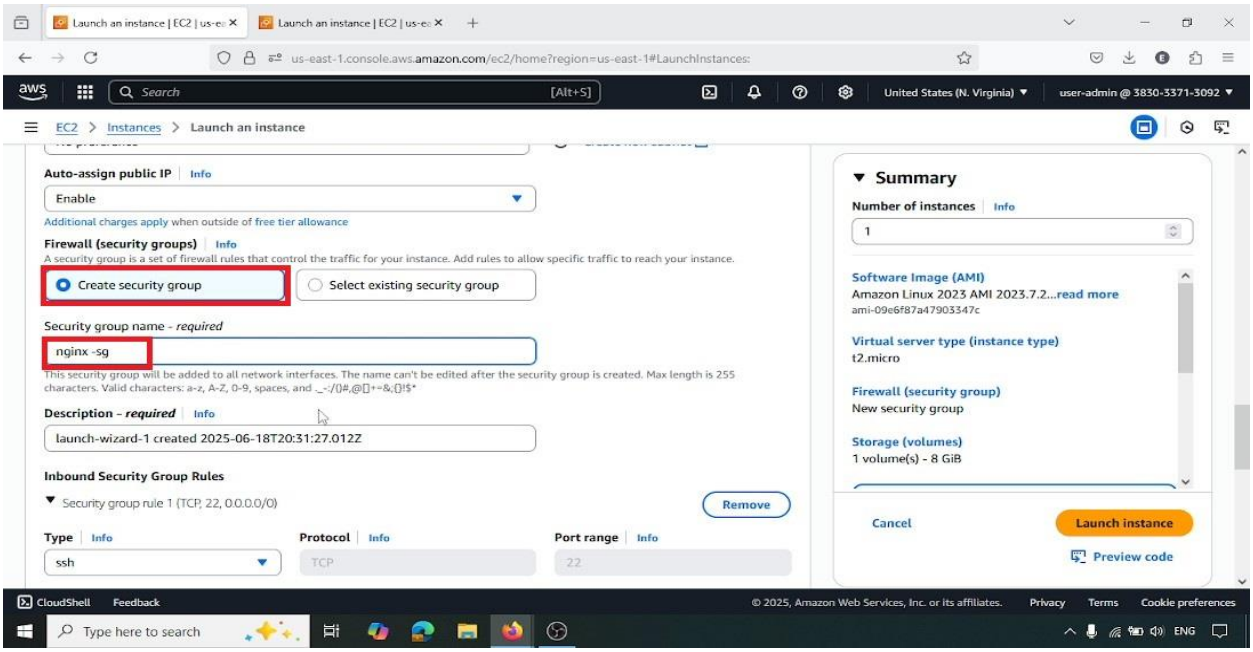


BY : EbRam

5: Enable Auto-assign IP Within the network settings, the "Auto-assign public IP" option is shown as enabled. This setting ensures that the EC2 instance automatically receives a public IPv4 address upon launch, allowing it to be accessible from the internet without manual IP configuration.

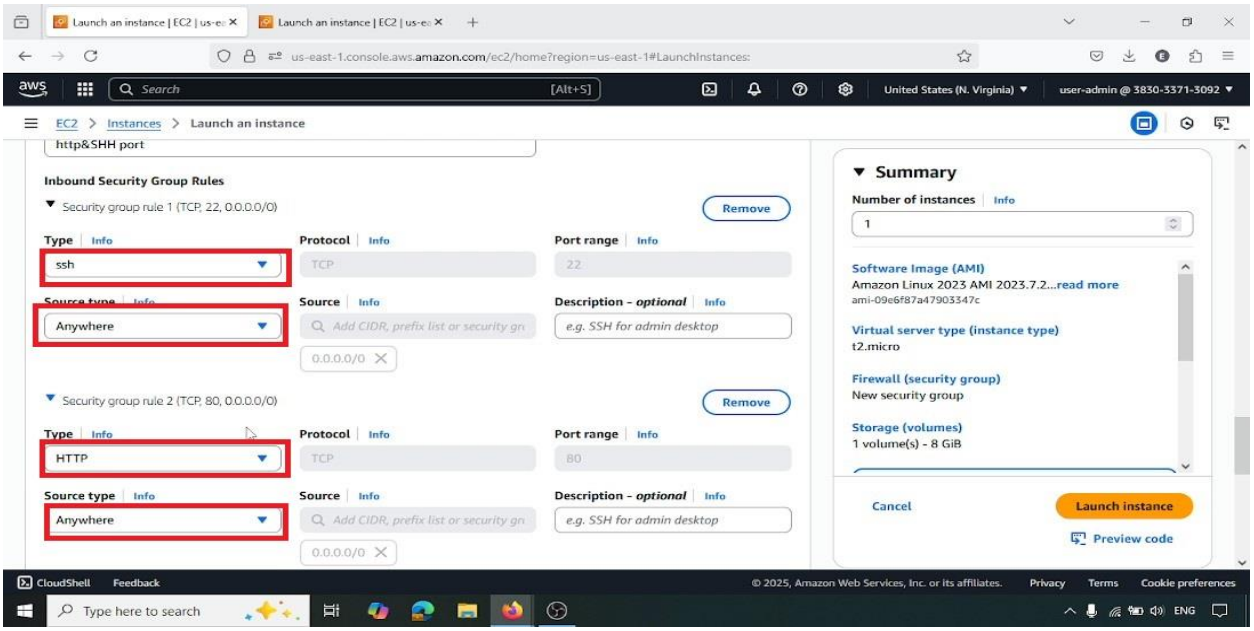


6: Create Security Group This image depicts the process of creating a new security group, named nginx-sg, which acts as a virtual firewall for the EC2 instance. Security groups control inbound and outbound traffic, defining which types of network communication are permitted to and from the instance.

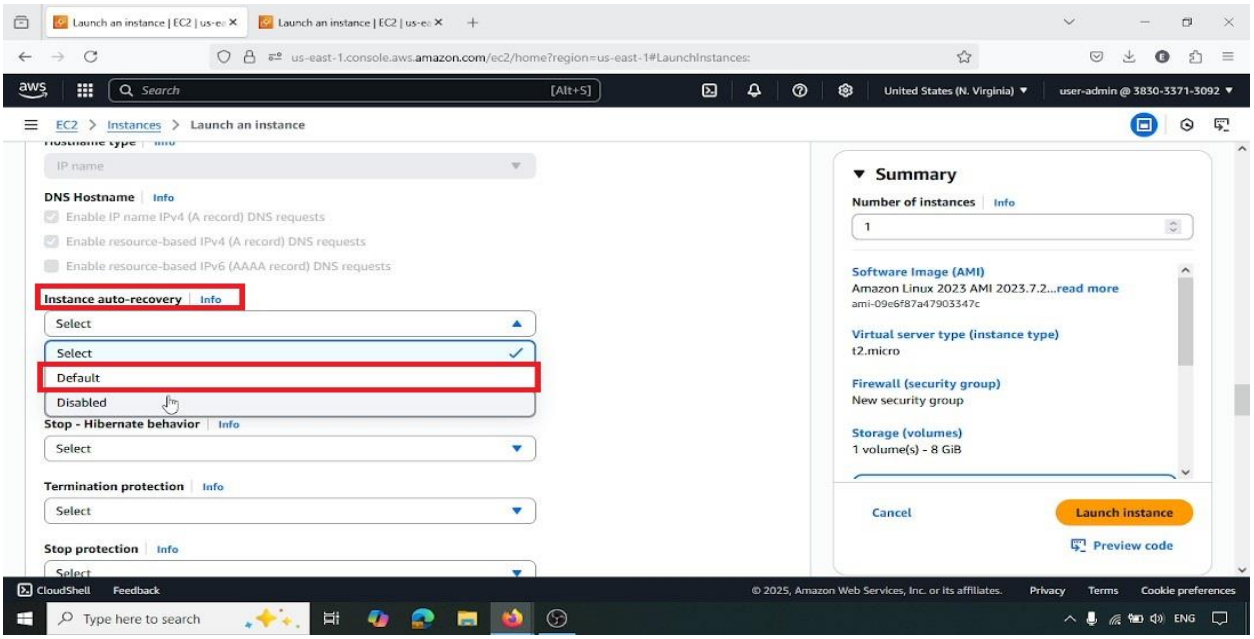


BY : EbRam

7: Enable Protocol SSH & HTTP Here, the inbound rules for the nginx-sg security group are configured. Specifically, rules are added to allow incoming traffic on port 22 (SSH) for secure remote access and port 80 (HTTP) for web traffic, enabling users to access the NGINX web server. Both are set to allow traffic from anywhere (0.0.0.0/0).

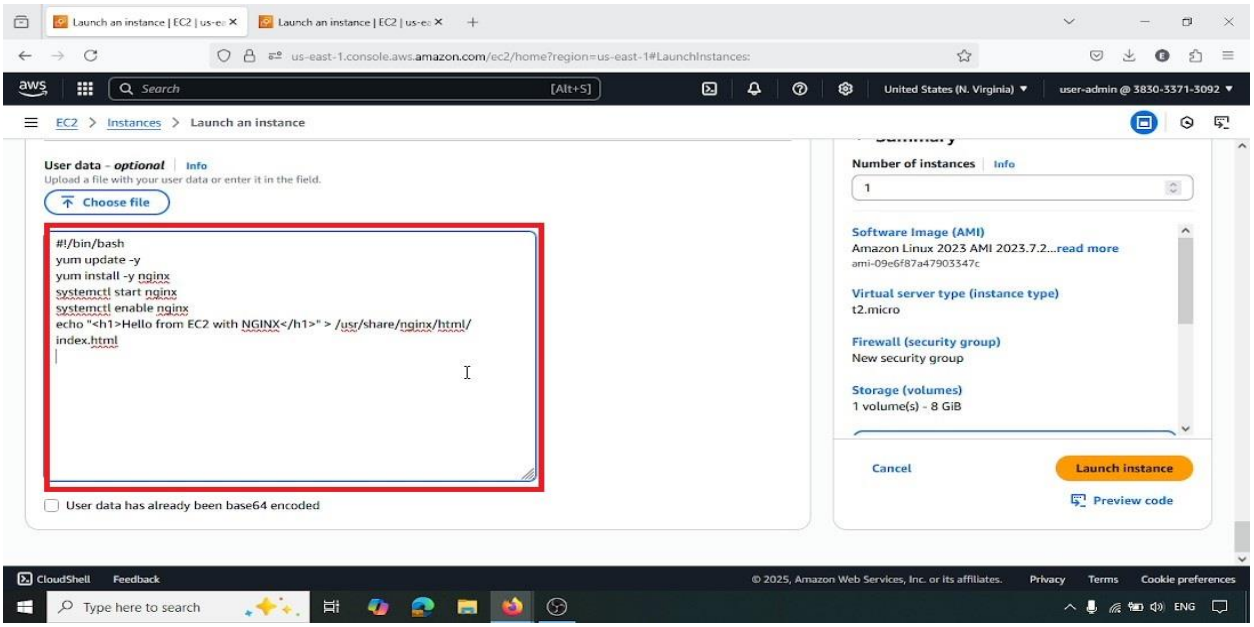


8: Instance Auto Recovery Make It Default This screenshot confirms that the instance auto-recovery setting is left at its default configuration. This feature allows AWS to automatically restart or recover the EC2 instance on different underlying hardware if it detects an unrecoverable system impairment, thereby enhancing availability.

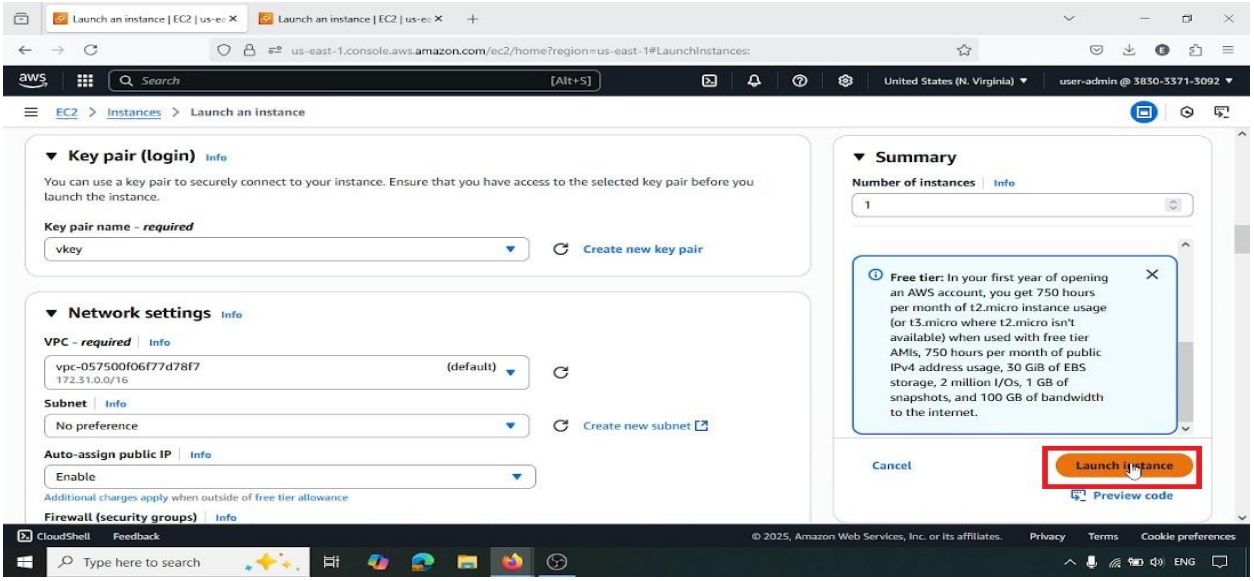


BY : EbRam

9: Paste the Script to Userdata for Launch NGINX When Started the Setup the EC2 This image shows the "User data" field where a shell script is input. This script is automatically executed when the EC2 instance first launches, performing a series of commands such as updating yum, installing NGINX, starting the NGINX service, enabling it to start on boot, and creating a simple index.html file to display "Hello from EC2 with NGINX."

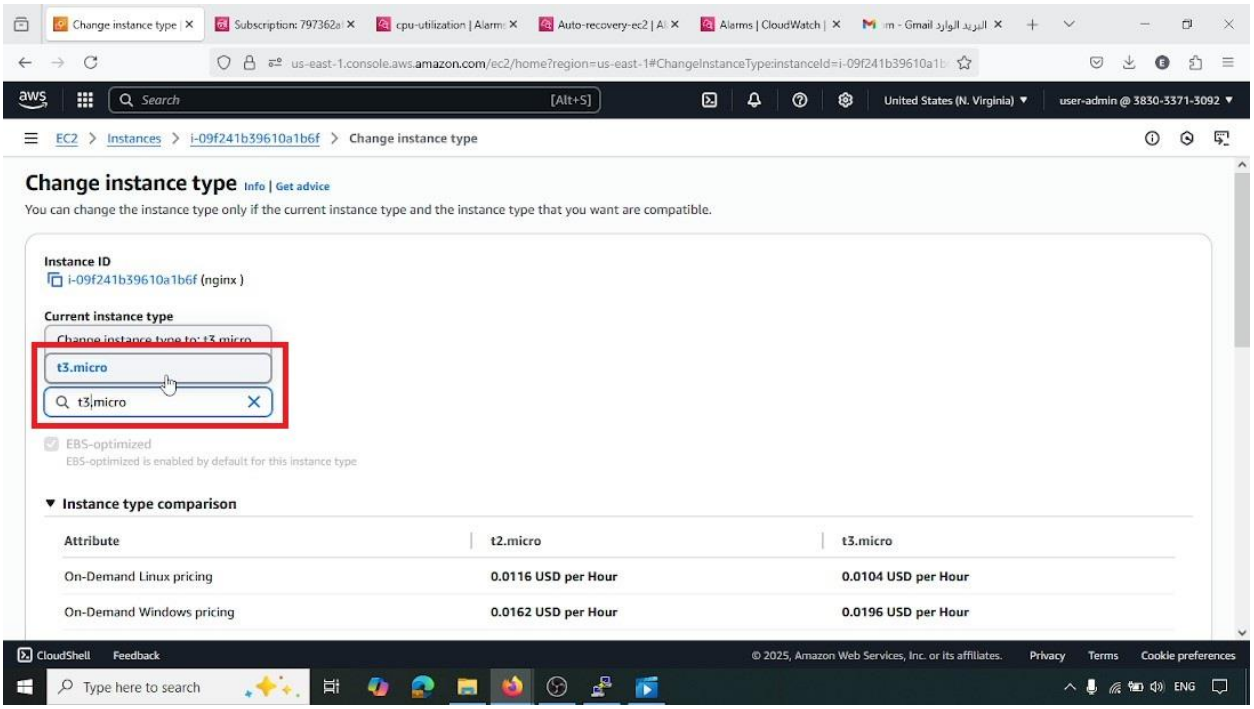


10: Then Launch Instance This final step in the instance launch wizard is depicted, showing the "Launch instance" button being clicked. After all configurations have been reviewed and confirmed, this action initiates the provisioning of the EC2 instance with all the specified settings.



BY : EbRam

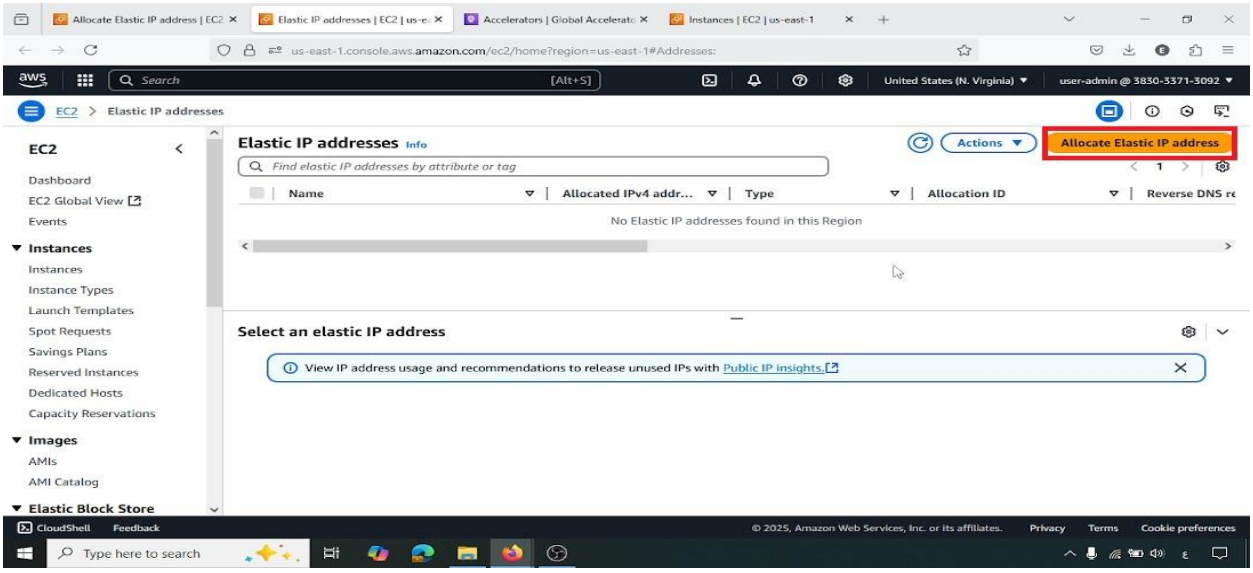
11: Change the Instance Type for Testing the CPU Utilization Usage This screenshot shows a user navigating to change the instance type, specifically to t3.micro. While not directly part of the initial NGINX setup, this action indicates preparation for testing scenarios, possibly to observe how different instance types affect CPU utilization and alarm behavior.



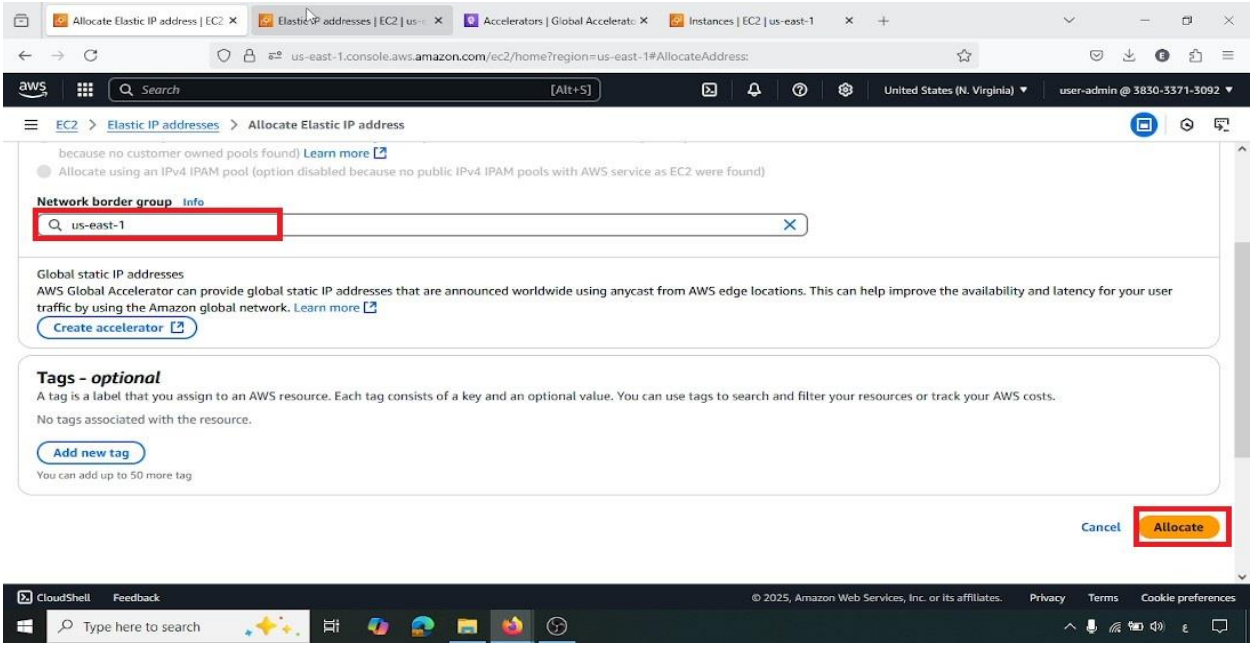
BY : EbRam

2. Associating an Elastic IP

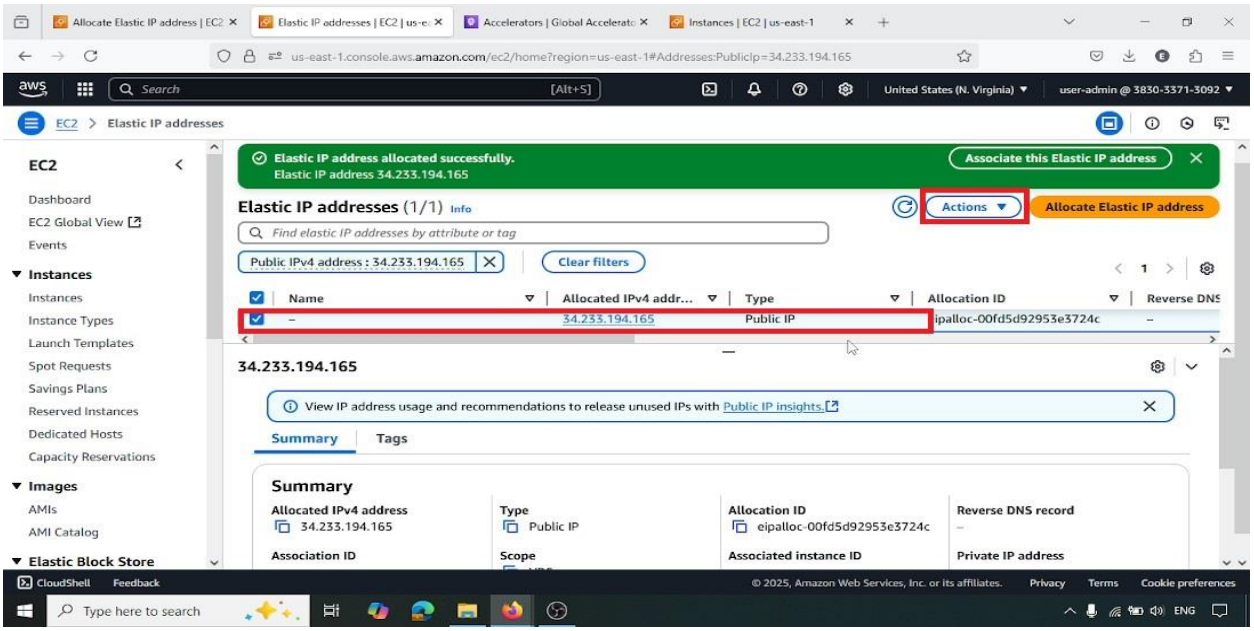
12: Allocate Elastic IP This image captures the initiation of the Elastic IP allocation process from the EC2 dashboard. An Elastic IP is a static, public IPv4 address that you can associate with your EC2 instance, providing a permanent endpoint for your application regardless of instance stop/start cycles.



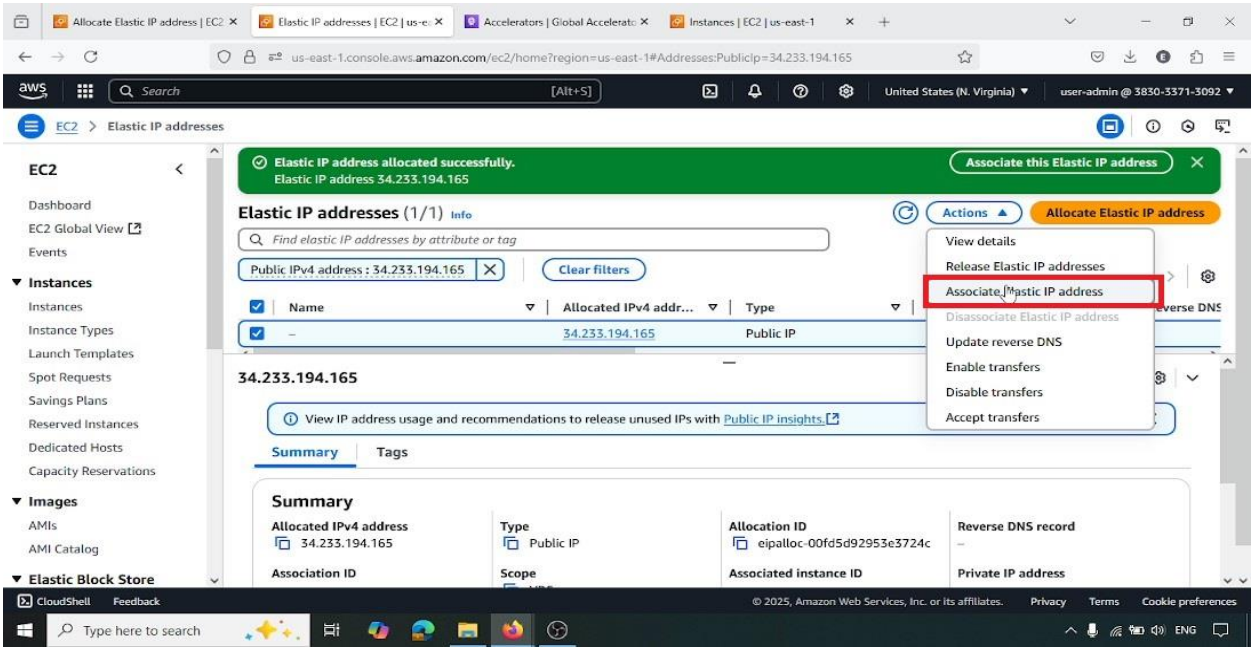
13: Choose the Region Then Next Here, the AWS region for the Elastic IP allocation is selected, in this case, us-east-1. Choosing the correct region is important as Elastic IPs are region-specific and can only be associated with instances within the same region.



14: Select the IP We Created Then Click Action After an Elastic IP has been allocated, this screenshot shows it being selected from the list. The "Actions" dropdown menu is then clicked to reveal options for managing the selected Elastic IP, such as associating it with an instance.

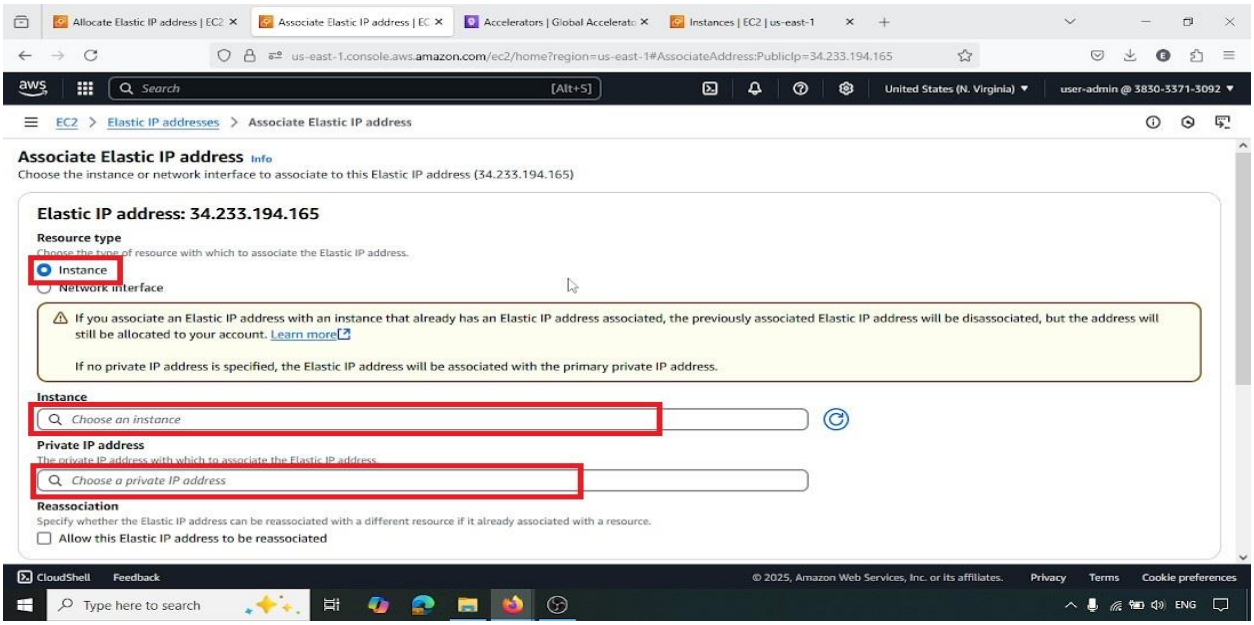


15: Click Associate Elastic IP From the "Actions" menu, the "Associate Elastic IP address" option is chosen. This action prepares the system to link the newly allocated static IP address to a running EC2 instance, ensuring a stable public IP.

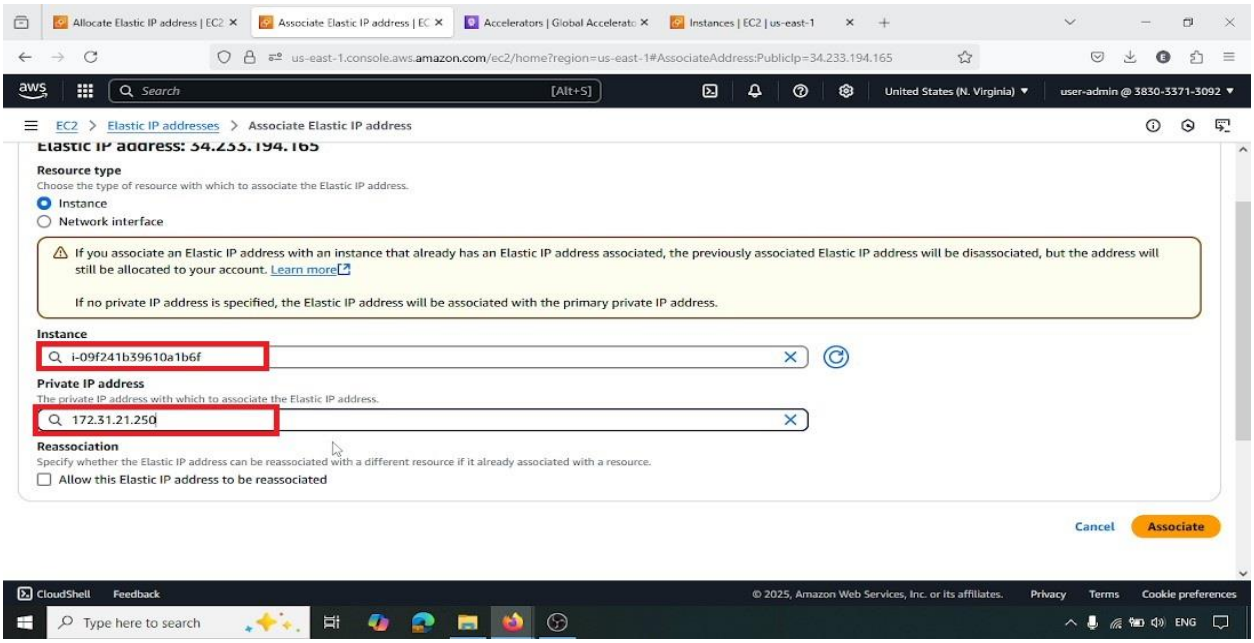


BY : EbRam

16: Choose Instance Then Type EC2 We Put the IP in It Then Write the Private IP of EC2 This image illustrates the crucial step of selecting the target EC2 instance and inputting its private IP address for the association. This ensures the Elastic IP is correctly mapped to the desired instance, enabling external access.

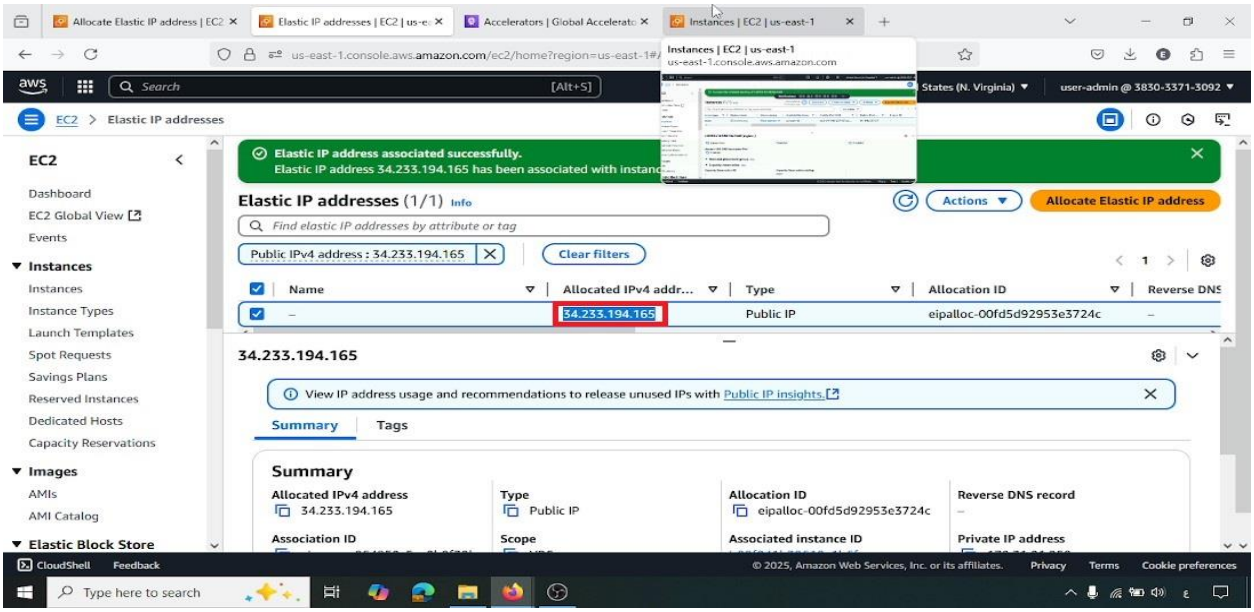


17: Like This This screenshot provides a visual confirmation of the previous step, showing how the selected instance and its private IP address are displayed within the Elastic IP association dialogue box. This serves as a quick visual check before finalizing the association.

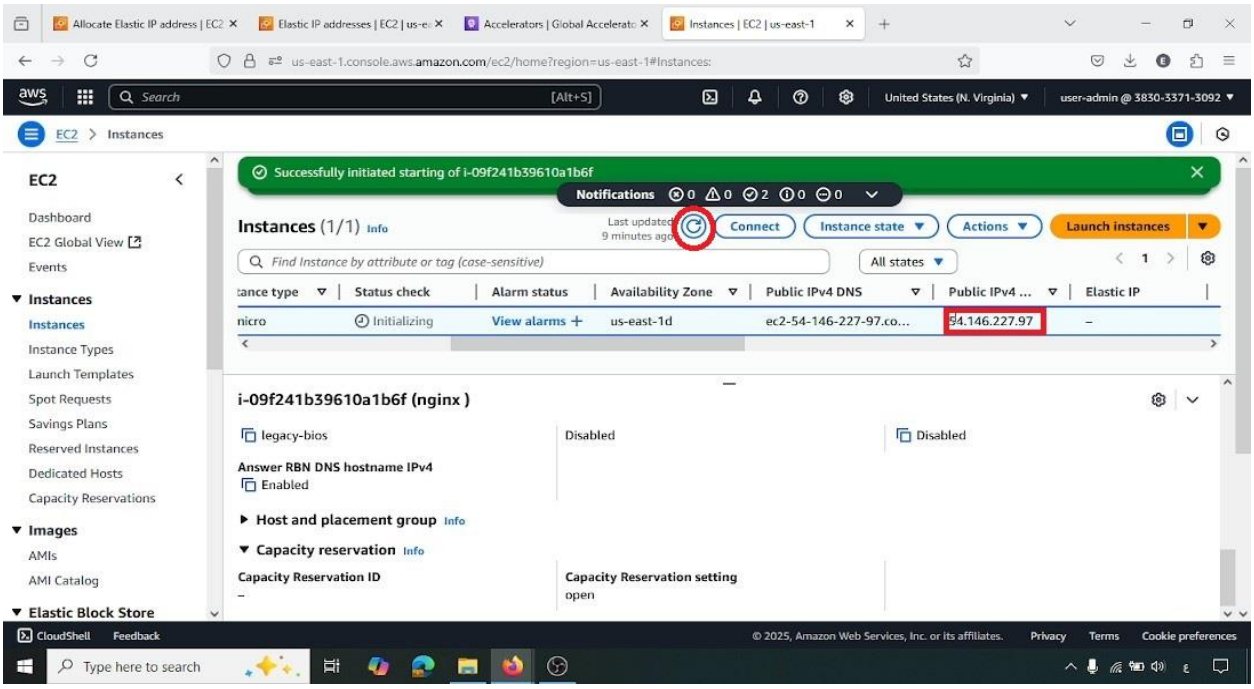


BY : EbRam

18: This the Elastic IP This image explicitly displays the Elastic IP address that has been successfully allocated. This static IP will now be used to consistently access the NGINX web server running on the EC2 instance, regardless of its underlying public IP changes.

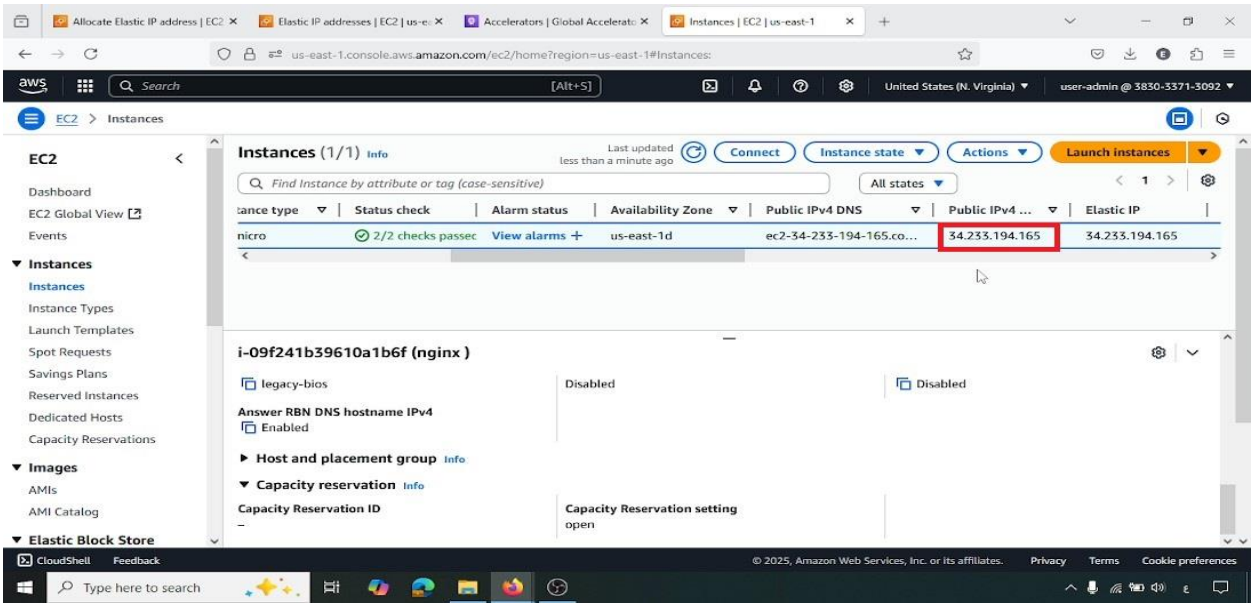


19: The Random IP of EC2 Prior to the Elastic IP association, this screenshot shows the EC2 instance with its initial, randomly assigned public IPv4 address. This dynamic IP would change upon instance stops and starts, highlighting the need for an Elastic IP for stable access.

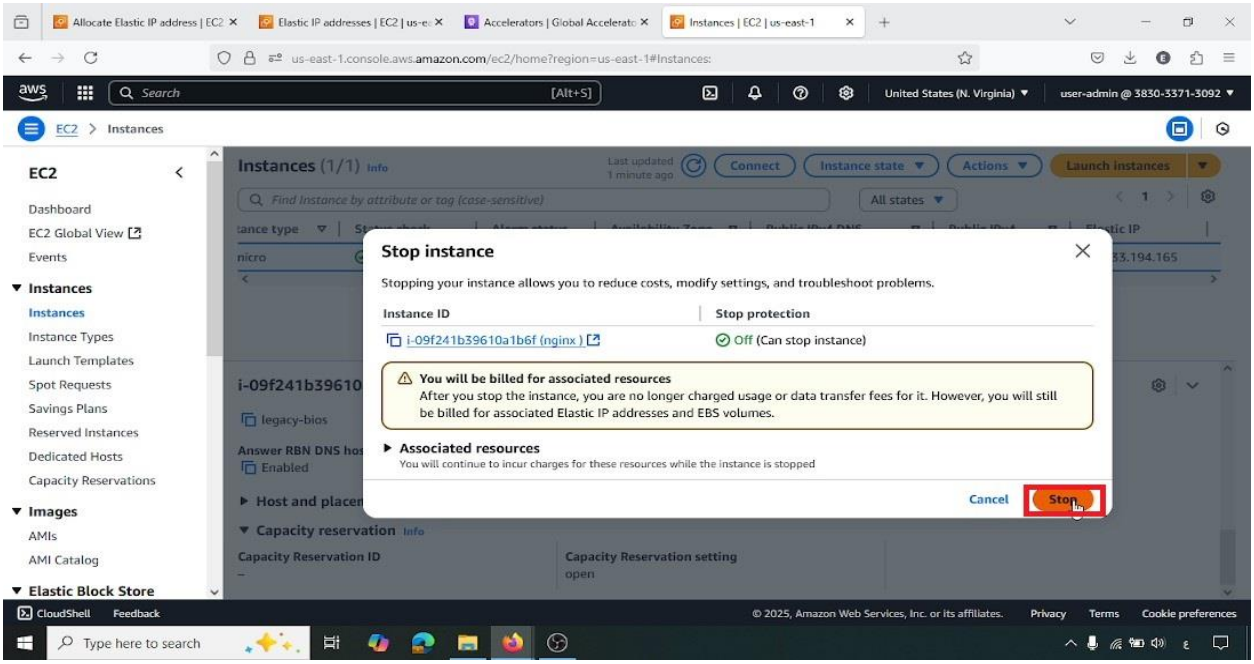


BY : EbRam

20: After Refresh the EC2 Get the Elastic IP This image, captured after refreshing the EC2 instances view, demonstrates that the EC2 instance now displays the newly associated Elastic IP address in its public IP field. This confirms that the static IP has been successfully linked and is active.

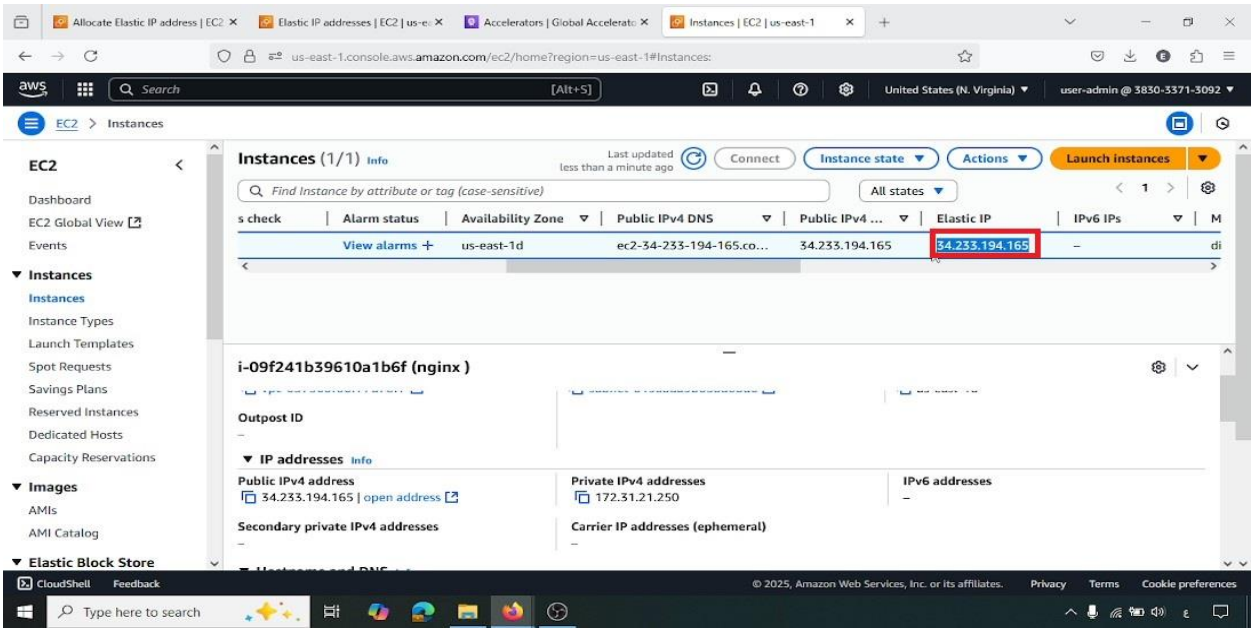


21: To Test That Is the Elastic IP We Stop the Instance To verify the persistent nature of the Elastic IP, this screenshot shows the user stopping the EC2 instance. Unlike dynamic public IPs, an Elastic IP is designed to remain associated with the instance even when it's stopped, ensuring continuous addressing.

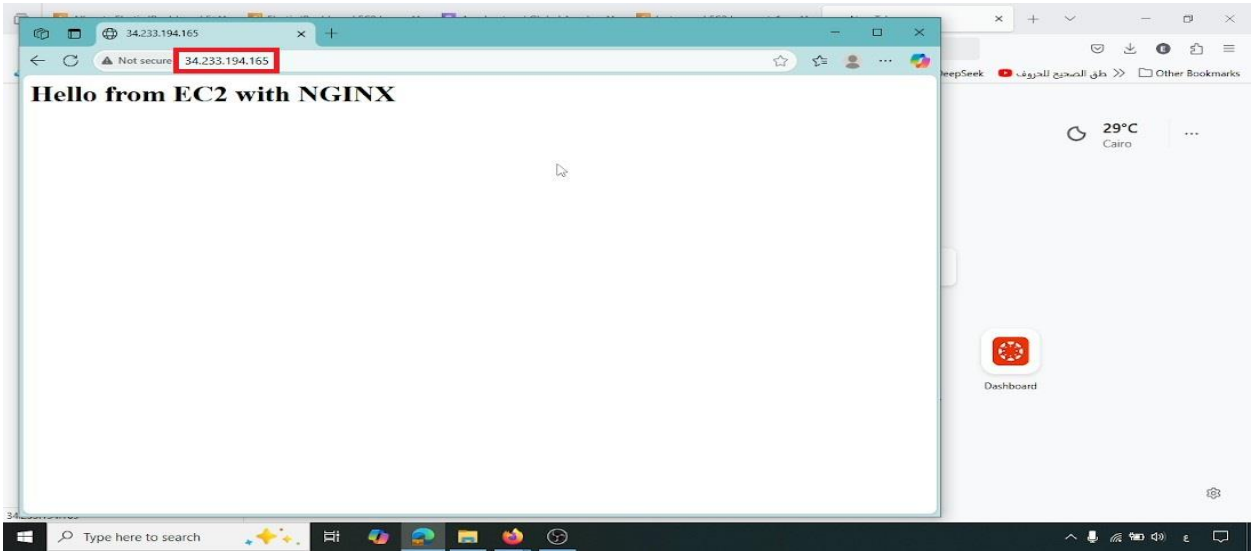


BY : EbRam

22: After EC2 Stopped the IP Still Assigned to EC2 This image confirms that even after the EC2 instance has been stopped, the Elastic IP address remains assigned to it. This crucial characteristic of Elastic IPs ensures that the public IP endpoint for the application remains constant, simplifying DNS management and external access.

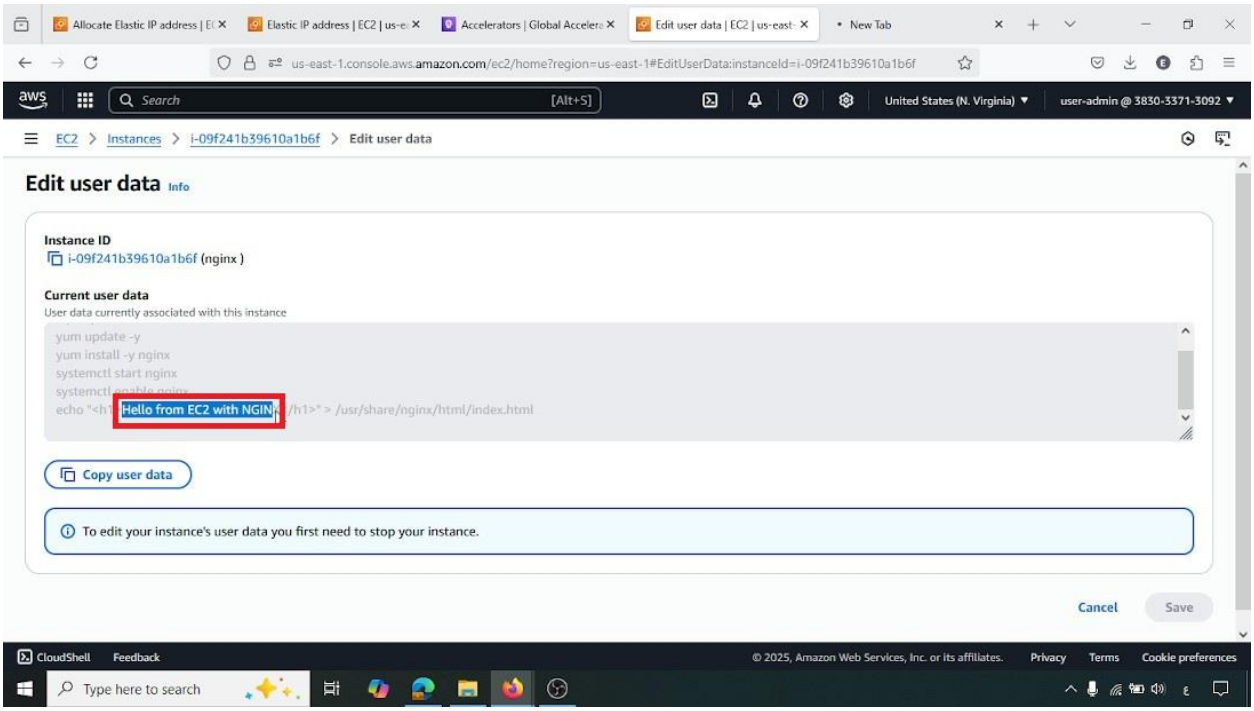


23: Now Paste the IP to Another Tab and As You Can See the Script That We Write in Userdata Is Worked and Display the Sentence This screenshot demonstrates a successful test of the NGINX deployment. The Elastic IP address is entered into a web browser, and the browser displays "Hello from EC2 with NGINX," confirming that the user data script correctly installed and configured NGINX and that it is accessible via the static IP.



BY : EbRam

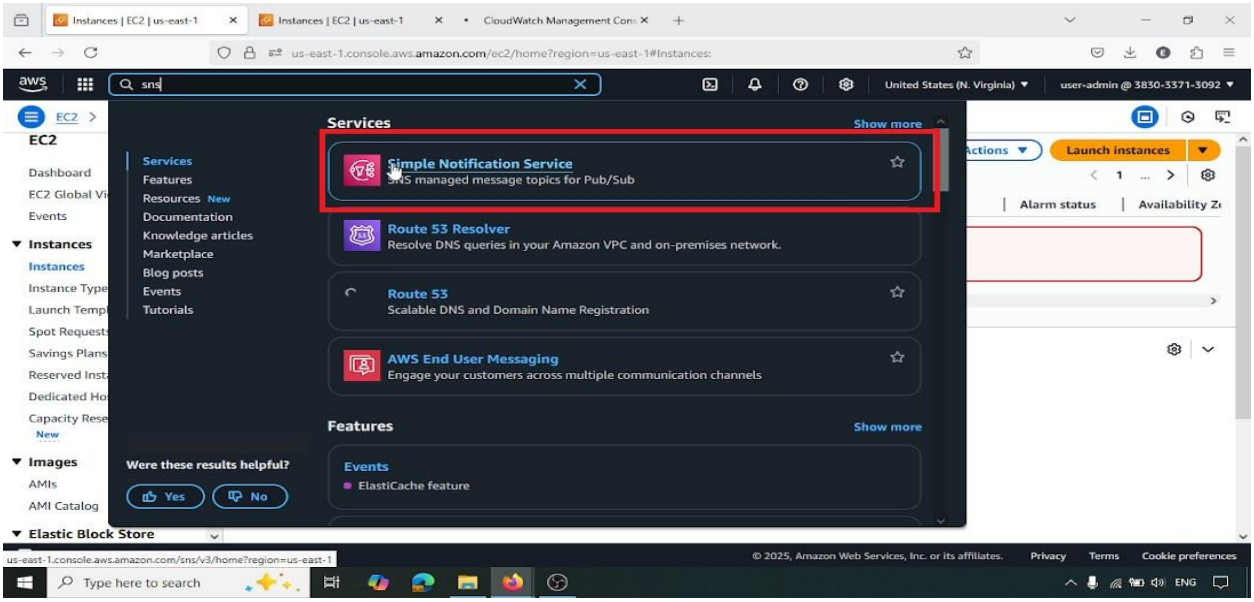
24: As You Can See in Userdata This image provides a reference back to the user data script, showing the code that was executed during instance launch. This visual reminder reinforces how the "Hello from EC2 with NGINX" message was generated, linking the script directly to the successful web server output.



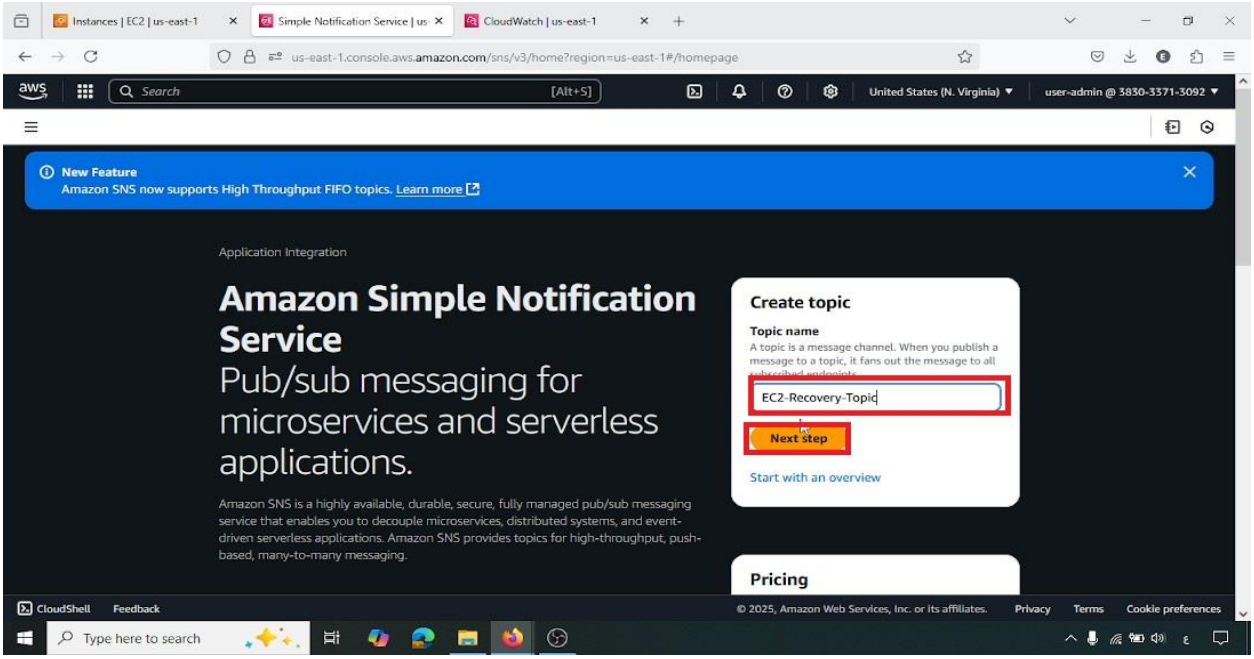
BY : EbRam

3. Configuring SNS Topics

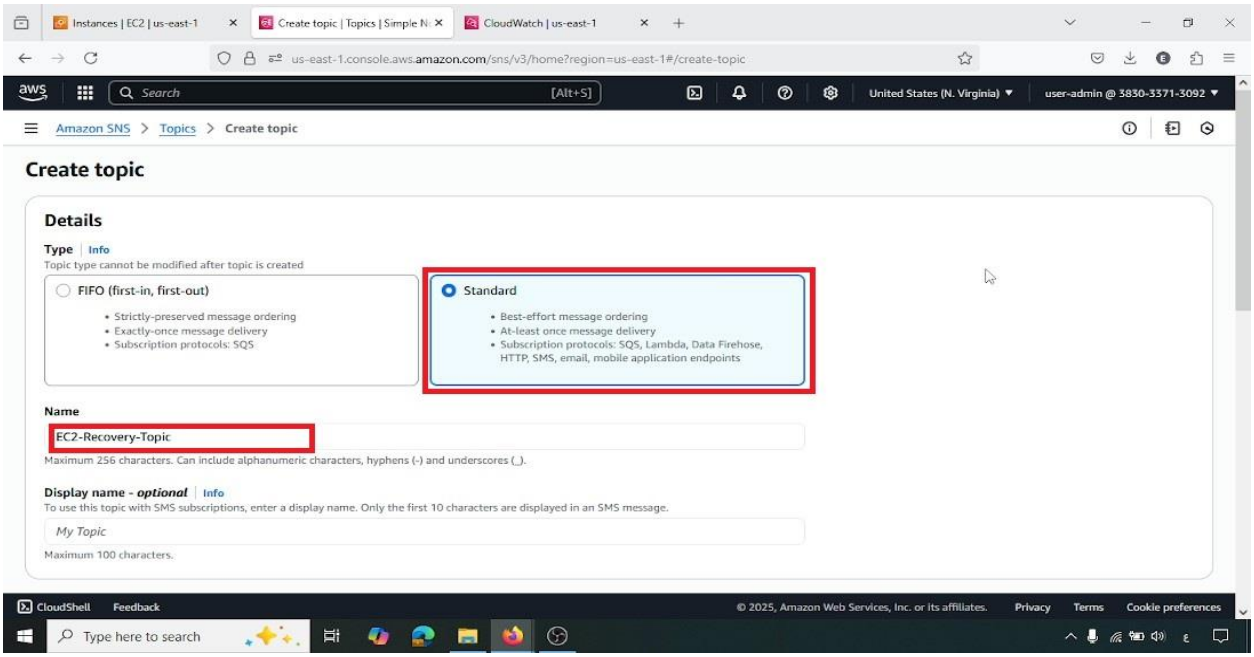
25: Go to SNS This screenshot shows the AWS console's search bar being used to locate the Simple Notification Service (SNS). Navigating to SNS is the first step in setting up notification channels for CloudWatch alarms.



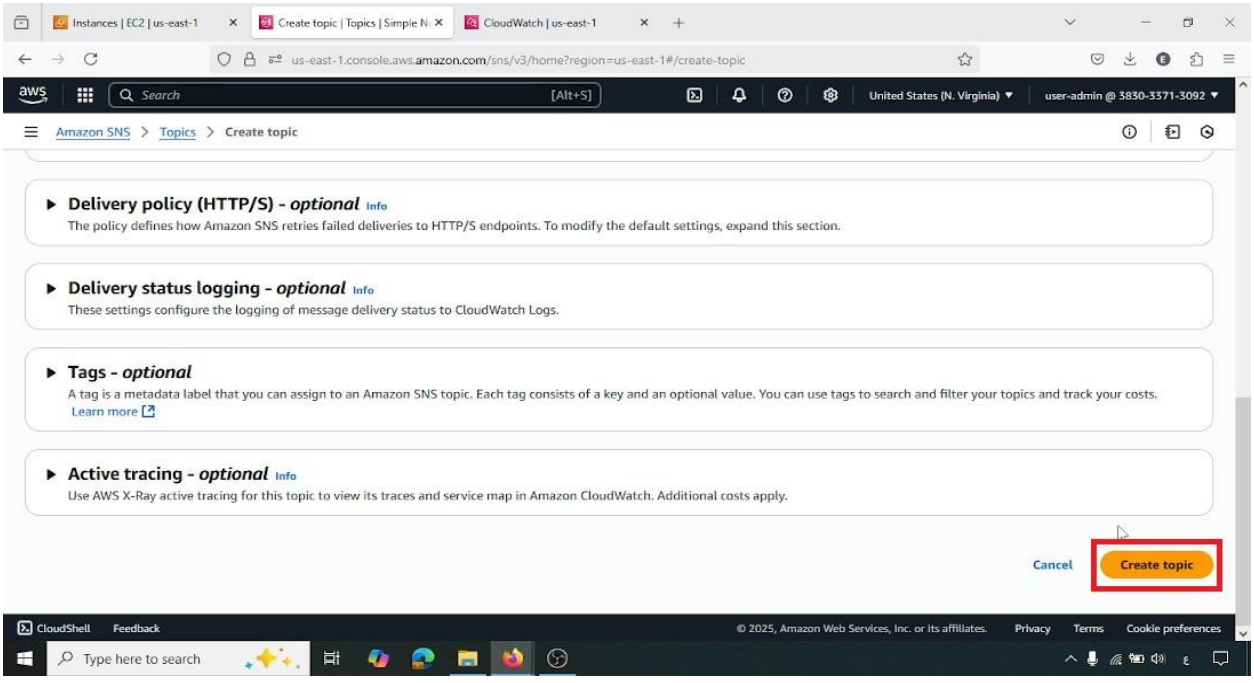
26: Type Topic Name and Click Create Here, the process of creating a new SNS topic is initiated by entering a desired topic name, EC2-Recovery-Topic, into the designated field. This topic will serve as a communication channel for instance recovery alarms.



27: Select Standard Option This image shows the selection of the "Standard" topic type when creating an SNS topic. Standard topics are suitable for most general-purpose messaging, including notifications from CloudWatch alarms.

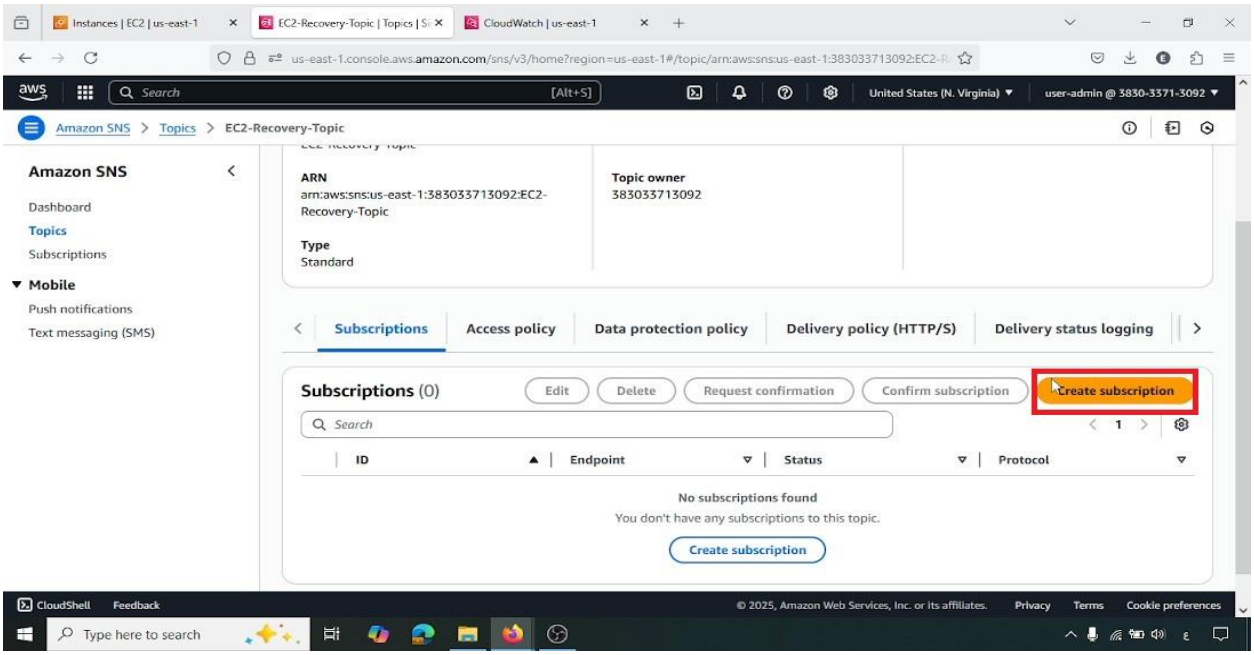


28: Then Create Topic This screenshot shows the final step of creating the SNS topic by clicking the "Create topic" button. This action provisions the EC2-Recovery-Topic, making it ready to receive and distribute messages.

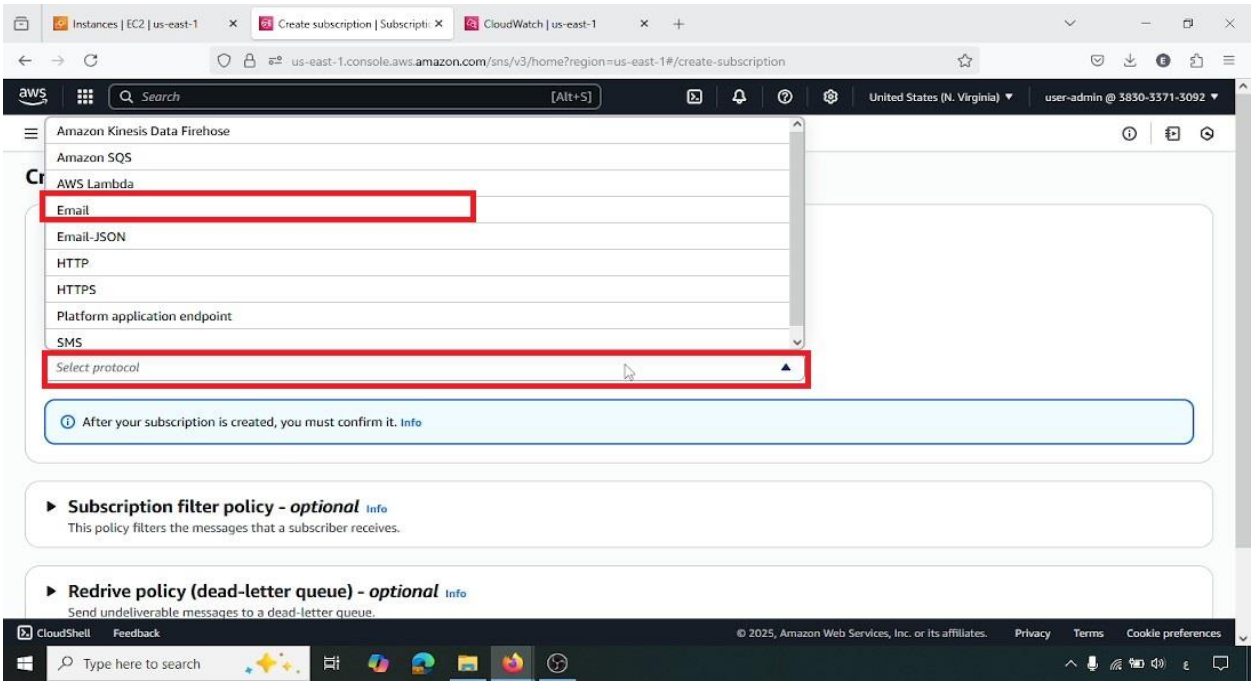


BY : EbRam

29: Create Subscription After the EC2-Recovery-Topic is created, this image shows the next step: creating a new subscription for it. Subscriptions define the endpoints (e.g., email addresses) that will receive messages published to the topic.

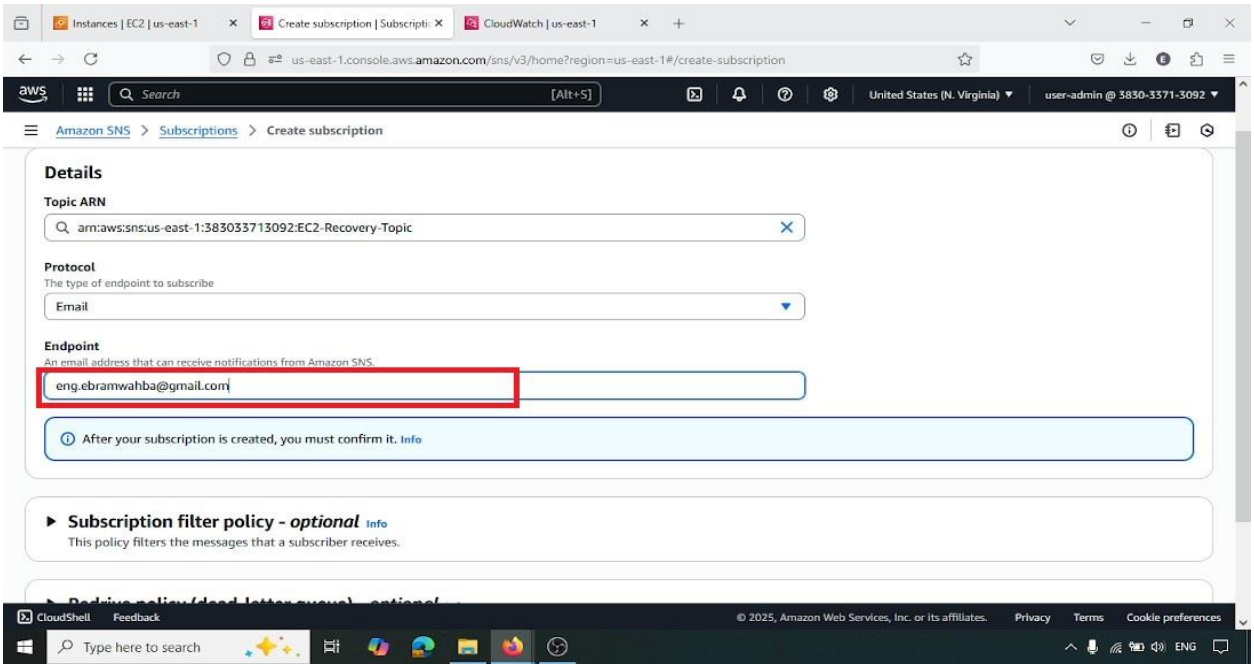


30: Choose Email Protocol This screenshot illustrates the selection of "Email" as the protocol for the SNS subscription. This choice ensures that when an alarm triggers and sends a message to the topic, an email notification will be sent to the specified address.

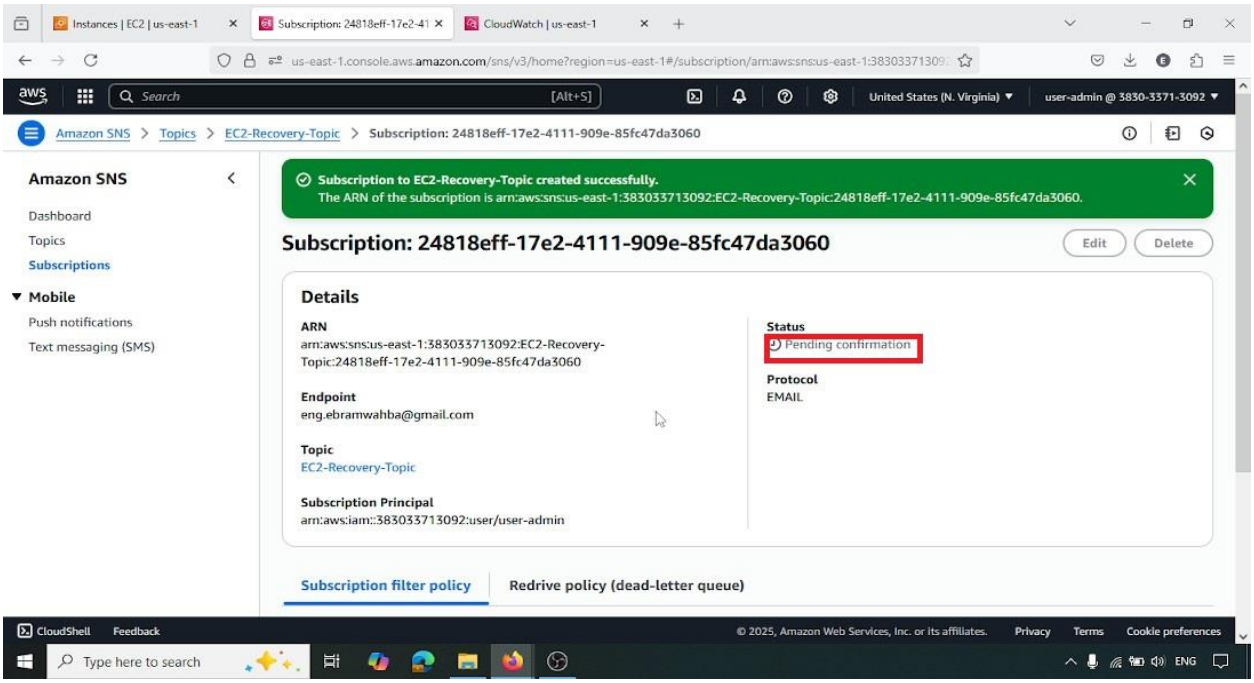


BY : EbRam

31: Enter Email Address Here, the recipient's email address is entered into the subscription configuration. This email address will receive all notifications and alerts published to the EC2-Recovery-Topic once the subscription is confirmed.

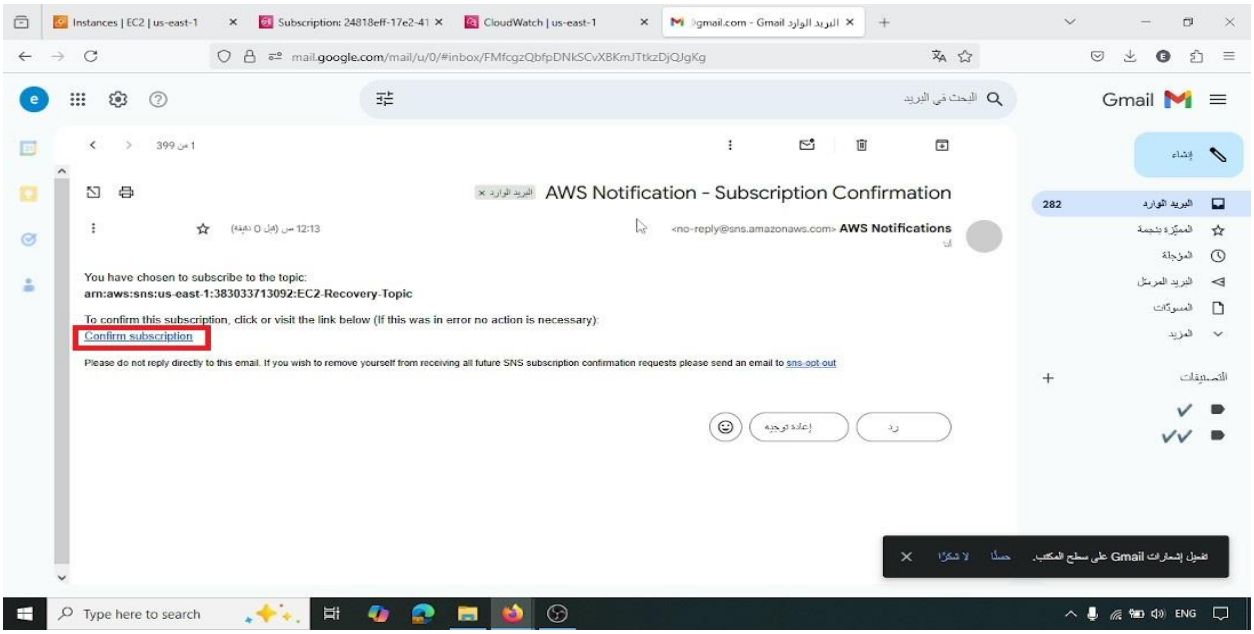


32: Pending Confirmation This image displays the status of the newly created email subscription as "Pending confirmation." Before notifications can be sent, the recipient must confirm their subscription by clicking a link sent to their email address.

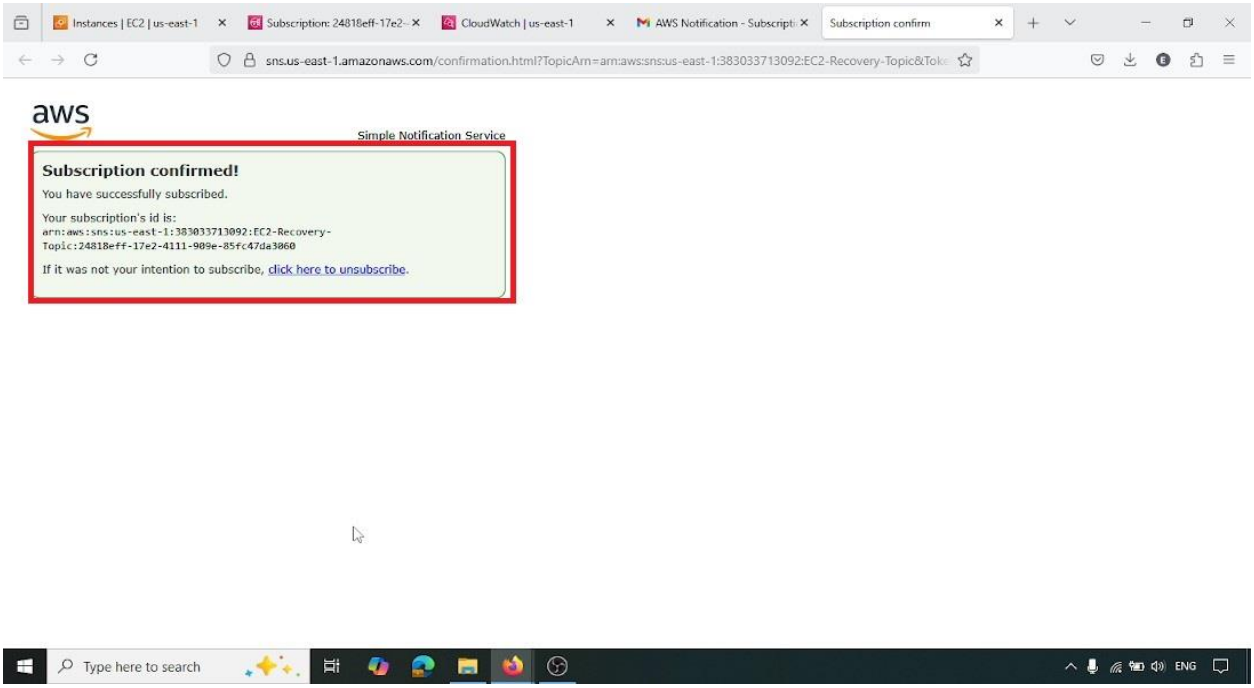


BY : EbRam

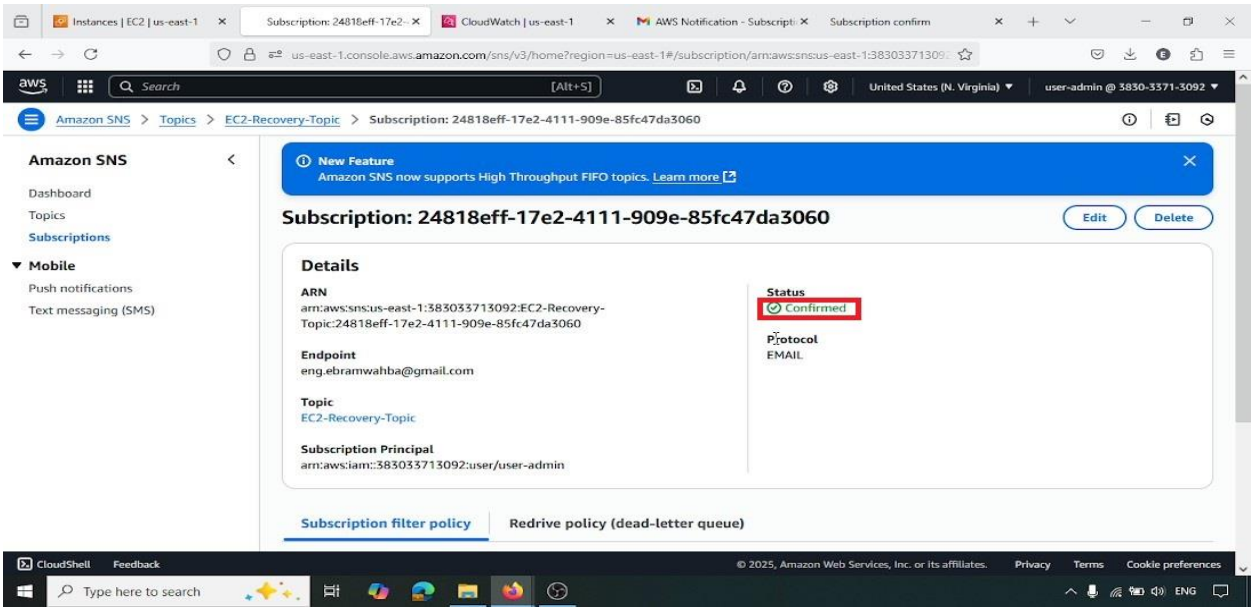
33: Confirm Subscription in Email This screenshot shows the email received by the subscriber, containing a link to "Confirm subscription." Clicking this link is a necessary step to activate the SNS subscription and begin receiving notifications.



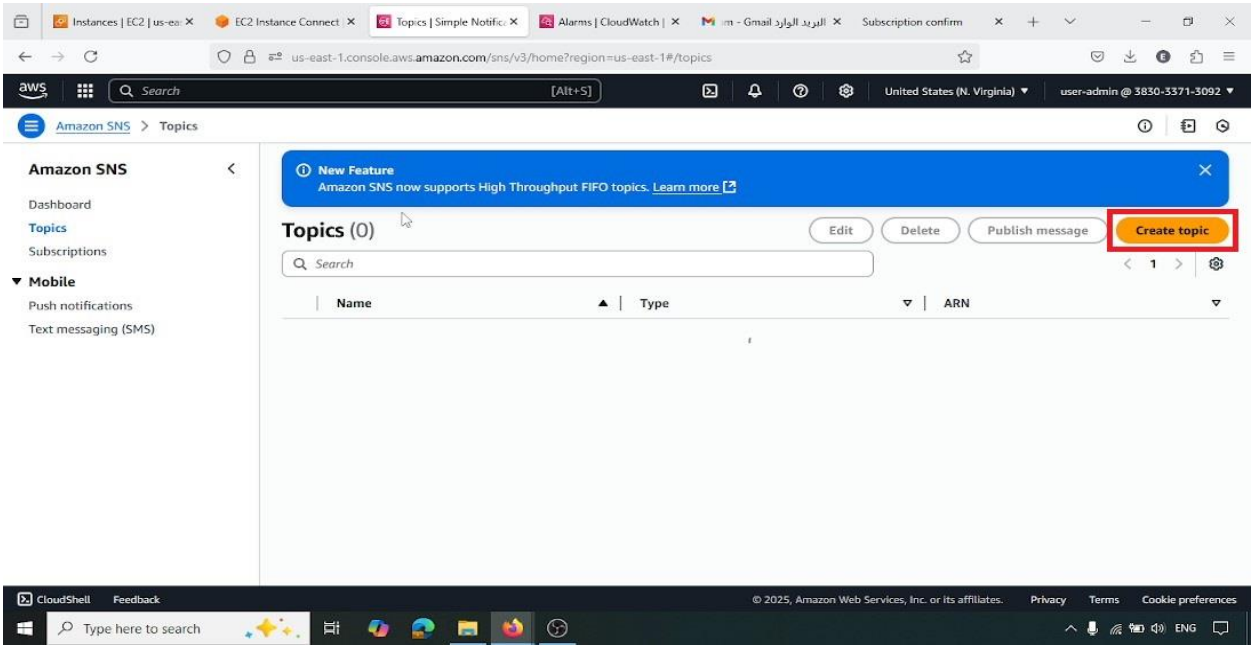
34: Subscription Confirmed This image shows the success message displayed after a user clicks the confirmation link in the email. This verifies that the subscription has been successfully activated and is now ready to receive messages from the SNS topic.



35: The Status Is Confirmed This screenshot from the AWS SNS console confirms that the subscription status for EC2-Recovery-Topic has changed from "Pending confirmation" to "Confirmed." This indicates that the email address is now actively subscribed and will receive notifications.

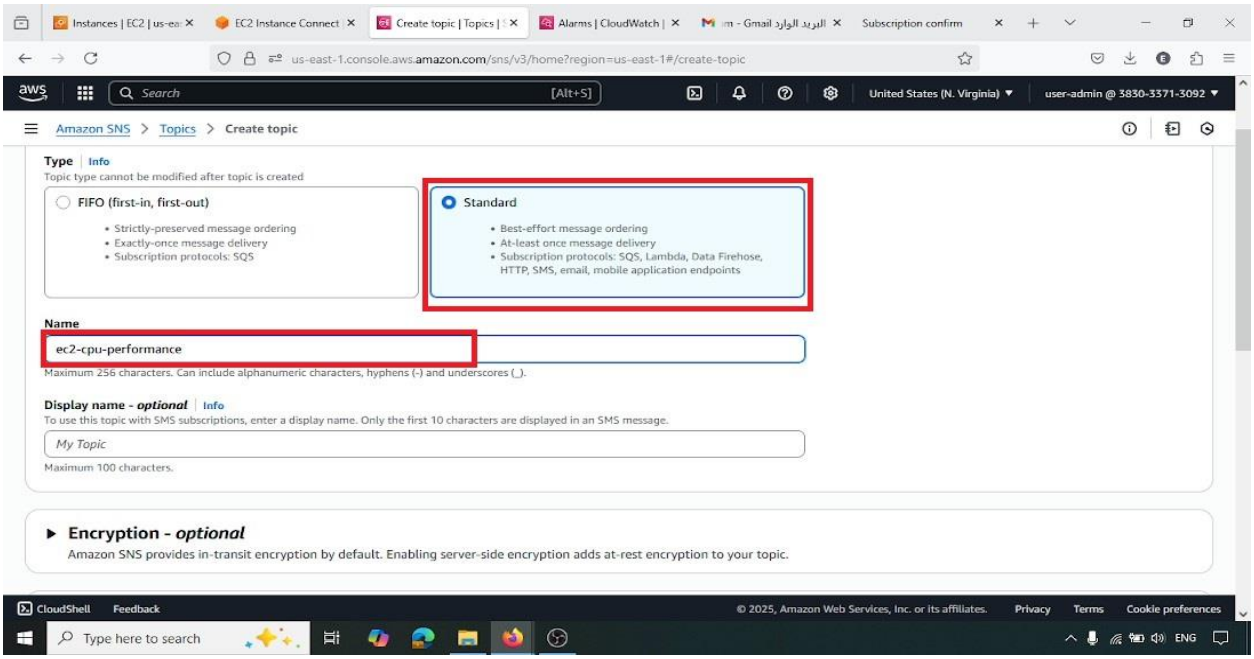


36: In Topic of SNS Click Create Topic This image, somewhat out of sequence, reiterates the process of creating a new SNS topic, specifically for the ec2-cpu-performance alarm. It emphasizes returning to the SNS dashboard to initiate the creation of another notification channel.

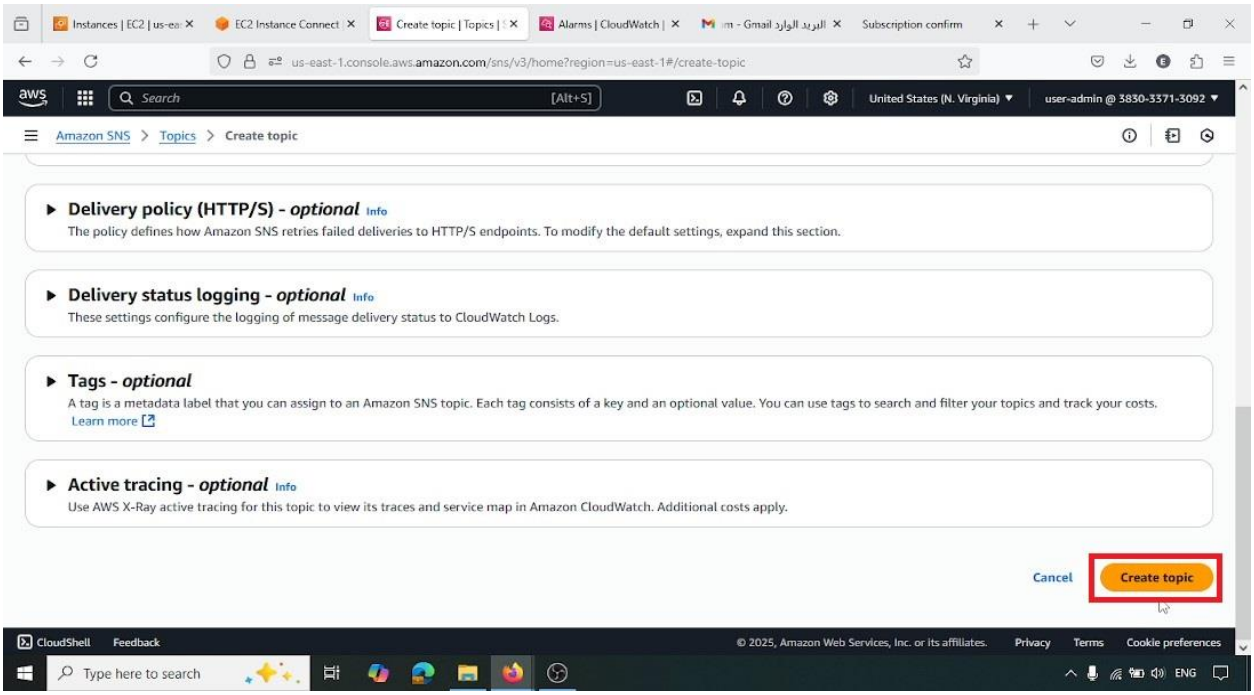


BY : EbRam

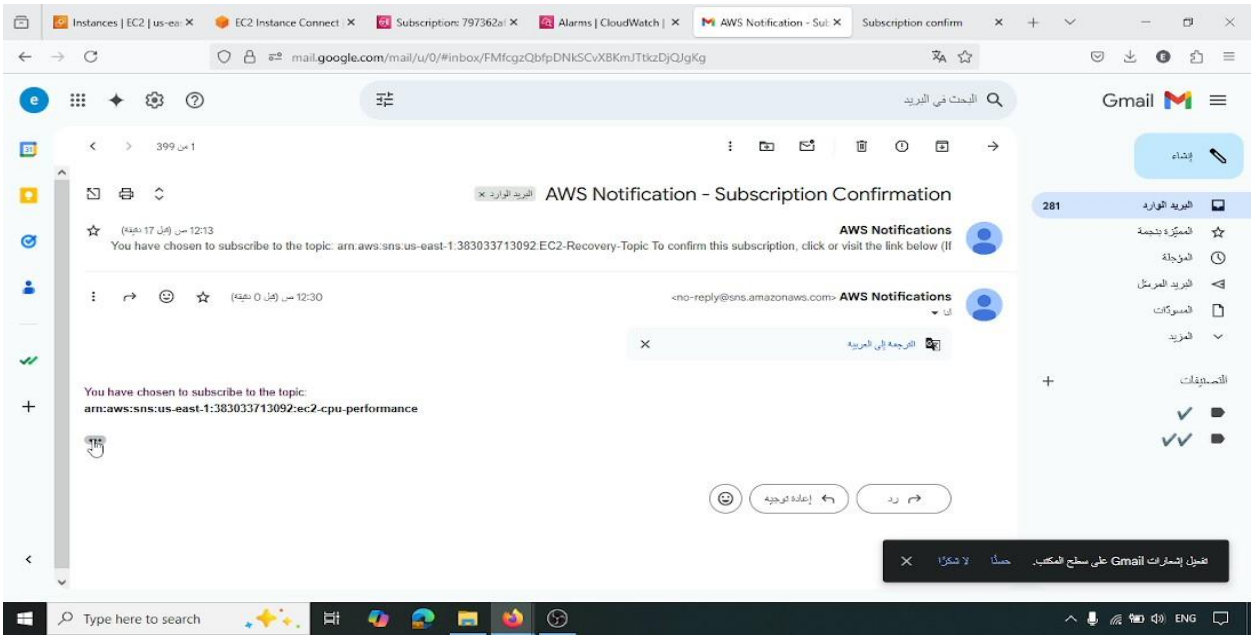
37: Write the Name of Topic Here, the name ec2-cpu-performance is entered for the second SNS topic. This topic will be dedicated to alerts related to high CPU utilization of the EC2 instance, separating it from recovery-related notifications.



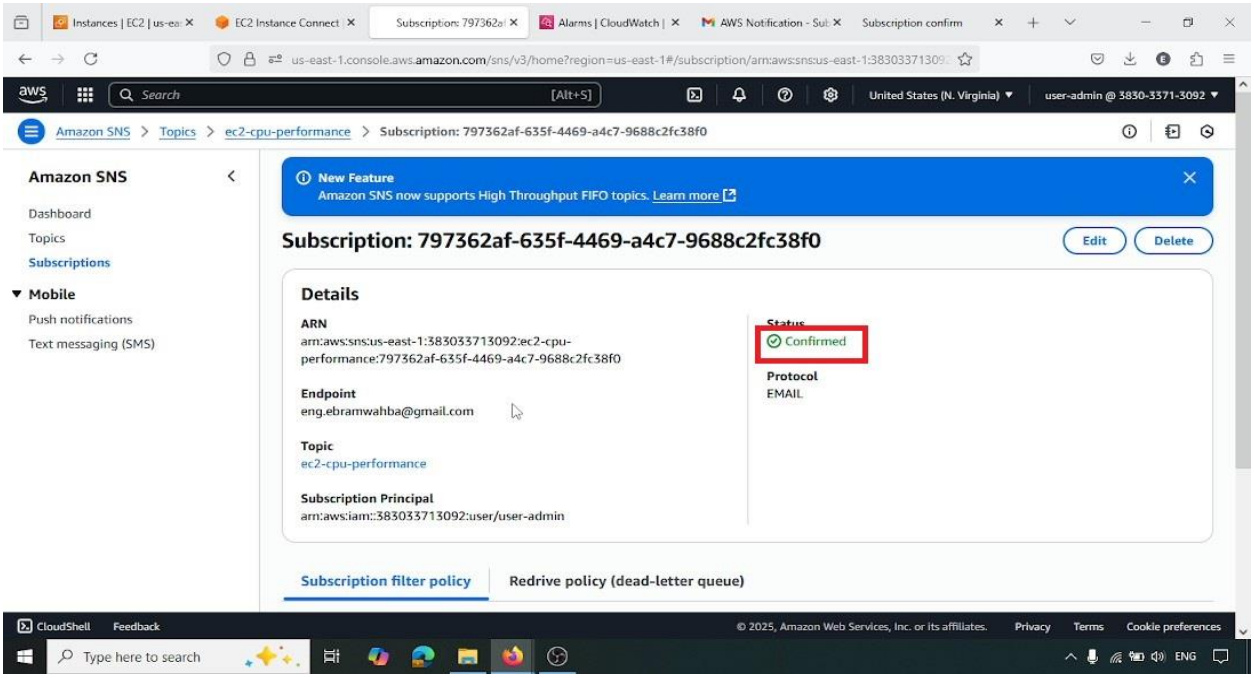
38: Then Create Topic This screenshot shows the final action to create the ec2-cpu-performance SNS topic. Once created, this topic will serve as the destination for CloudWatch alarms monitoring the instance's CPU usage.



39: Confirm Subscribe IB Usage This image refers to the confirmation process for the ec2-cpu-performance subscription. Similar to the recovery topic, an email is sent to the subscribed address, requiring a click on a confirmation link to activate the subscription.

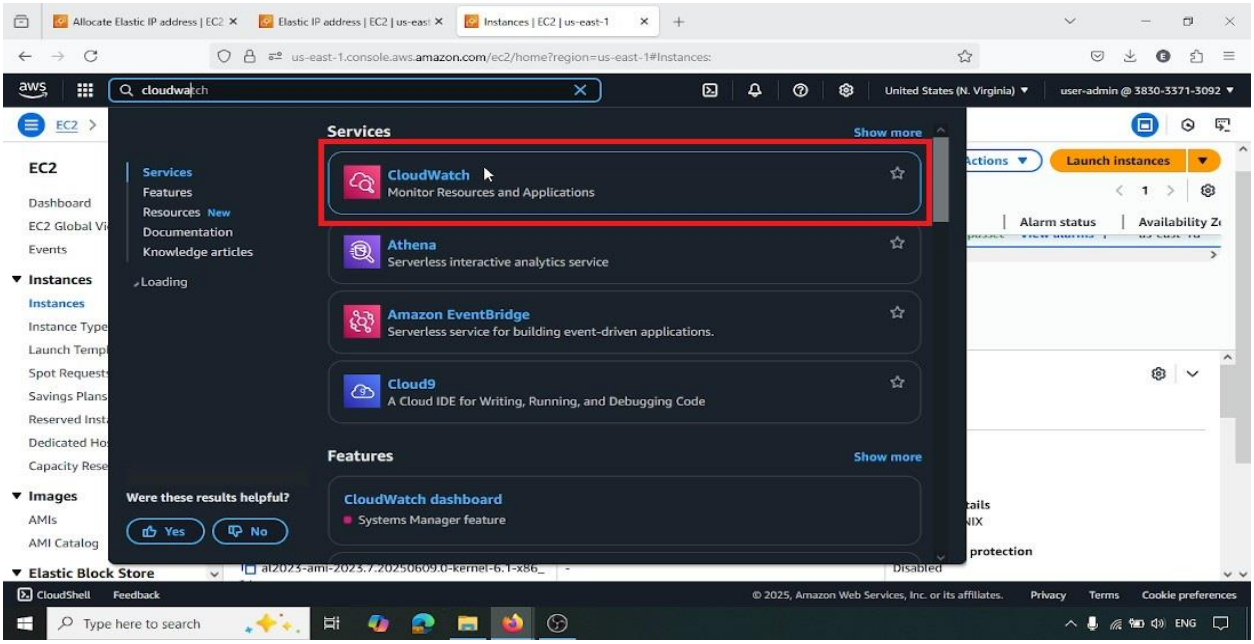


40: The Status Is Confirmed This screenshot verifies that the subscription for the ec2-cpu-performance SNS topic has also been successfully confirmed. Both necessary notification channels are now active and ready to receive alerts.

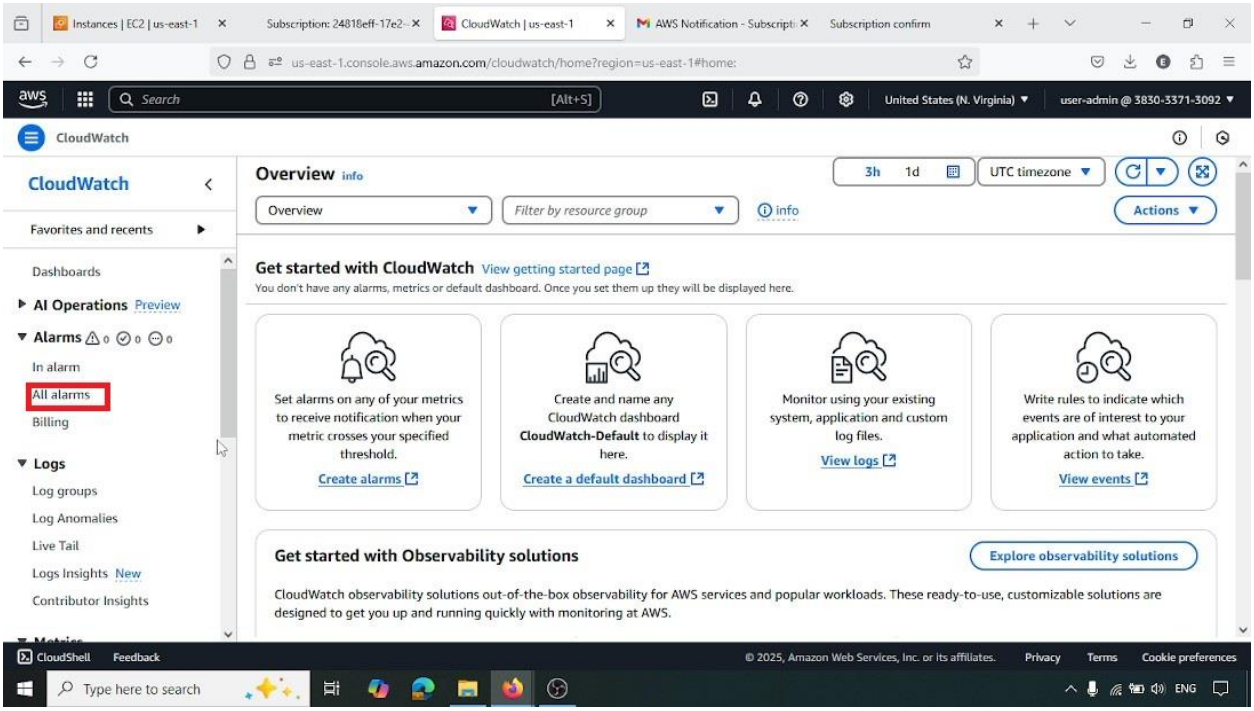


BY : EbRam

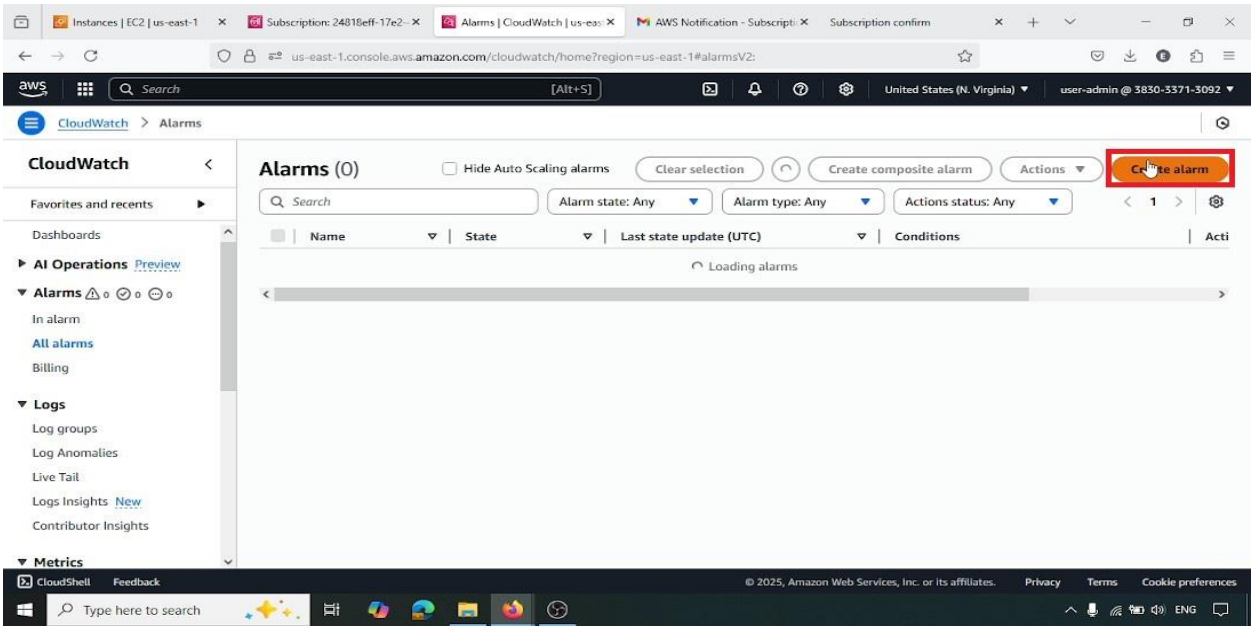
4. Setting up CloudWatch Alarms



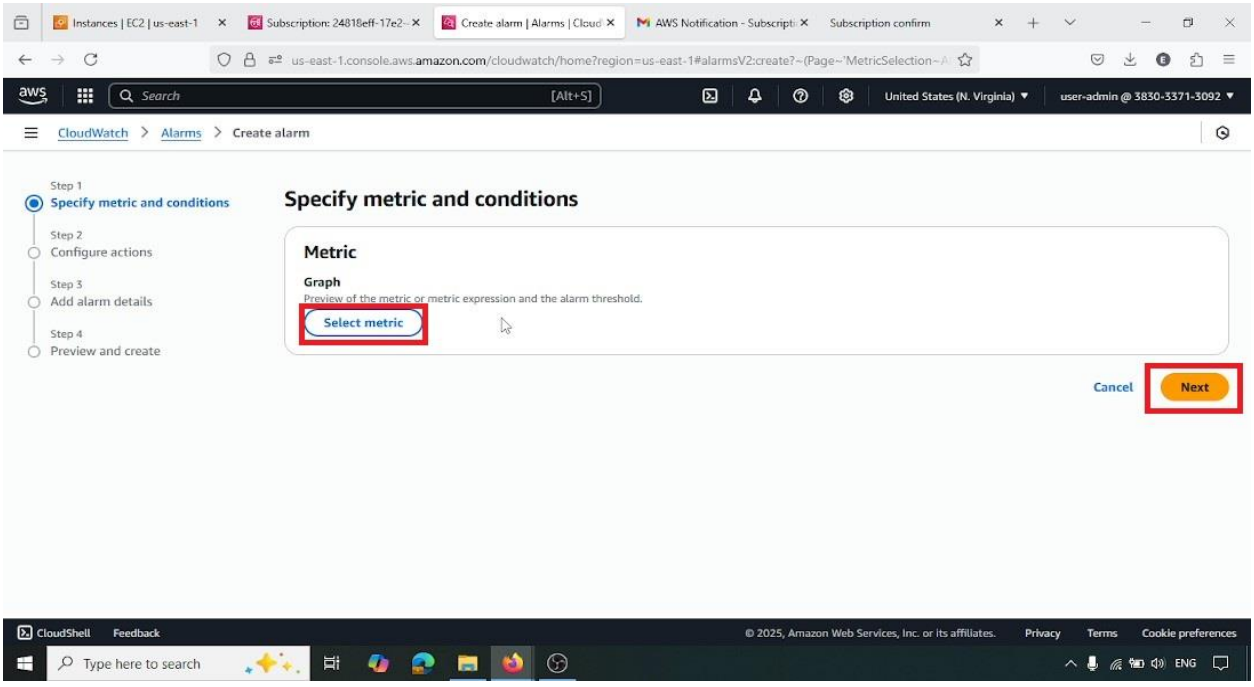
41: In CloudWatch Click All Alarm in the Left Side This screenshot shows the CloudWatch dashboard, with the "Alarms" section and "All alarms" option selected in the left navigation pane. This is the central location for viewing and managing all defined CloudWatch alarms.



42: Click Create Alarm Within the CloudWatch "All alarms" view, this image highlights the "Create alarm" button. Clicking this button initiates the wizard for configuring a new monitoring alarm, which can be based on various AWS metrics.utilization or system health.

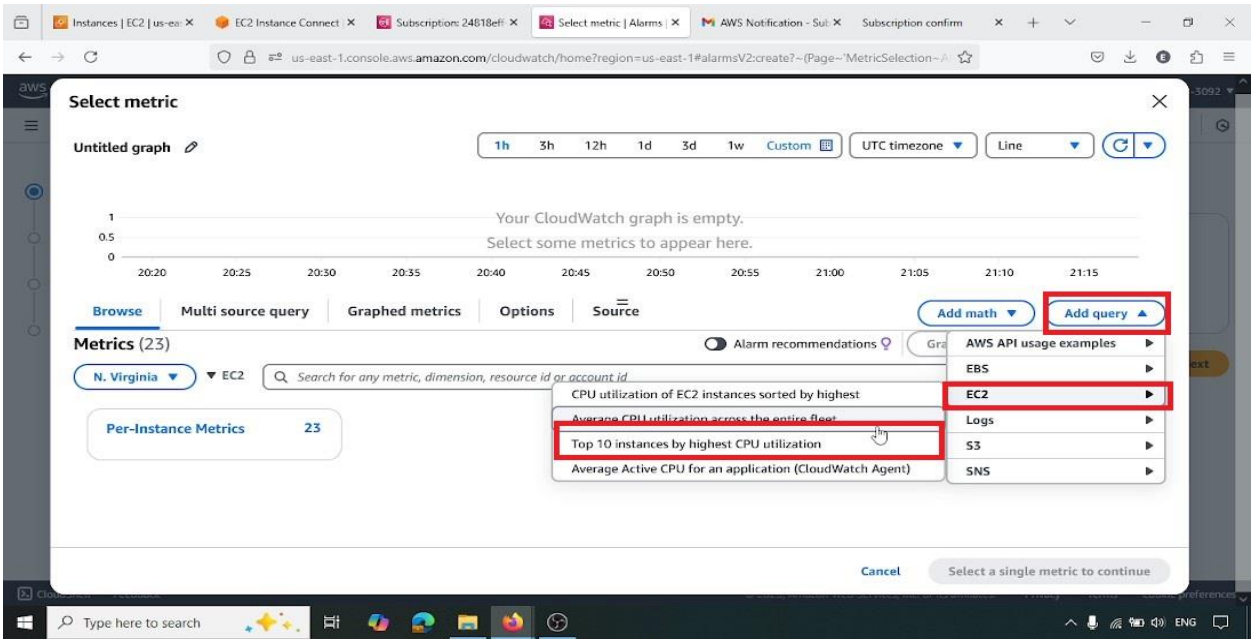


43: Choice Select Metric This screenshot shows the first step in creating a new CloudWatch alarm, which is to "Select metric." This involves choosing the specific data point or performance indicator that the alarm will monitor, such as CPU utilization or system health.points

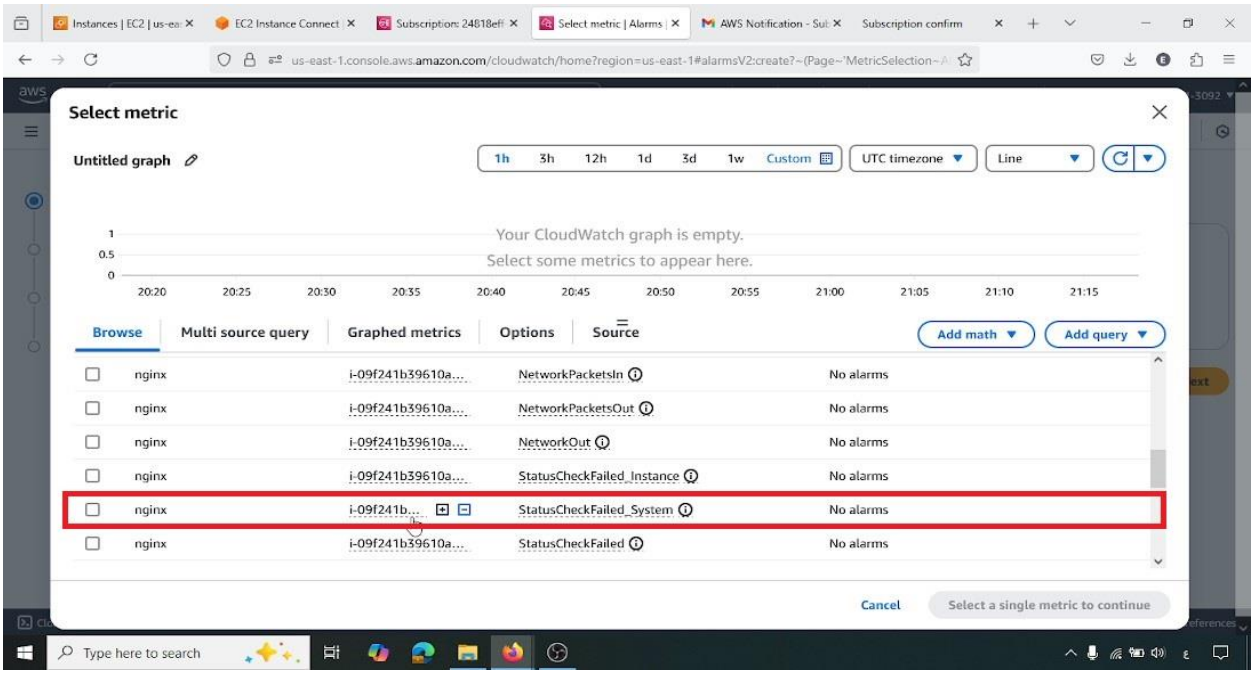


BY : EbRam

44: From Add Query Choice EC2 Then To narrow down the metrics, this image shows selecting "EC2" from the available services. This filters the metric list to display only those relevant to Amazon EC2 instances, making it easier to find desired monitoring points.

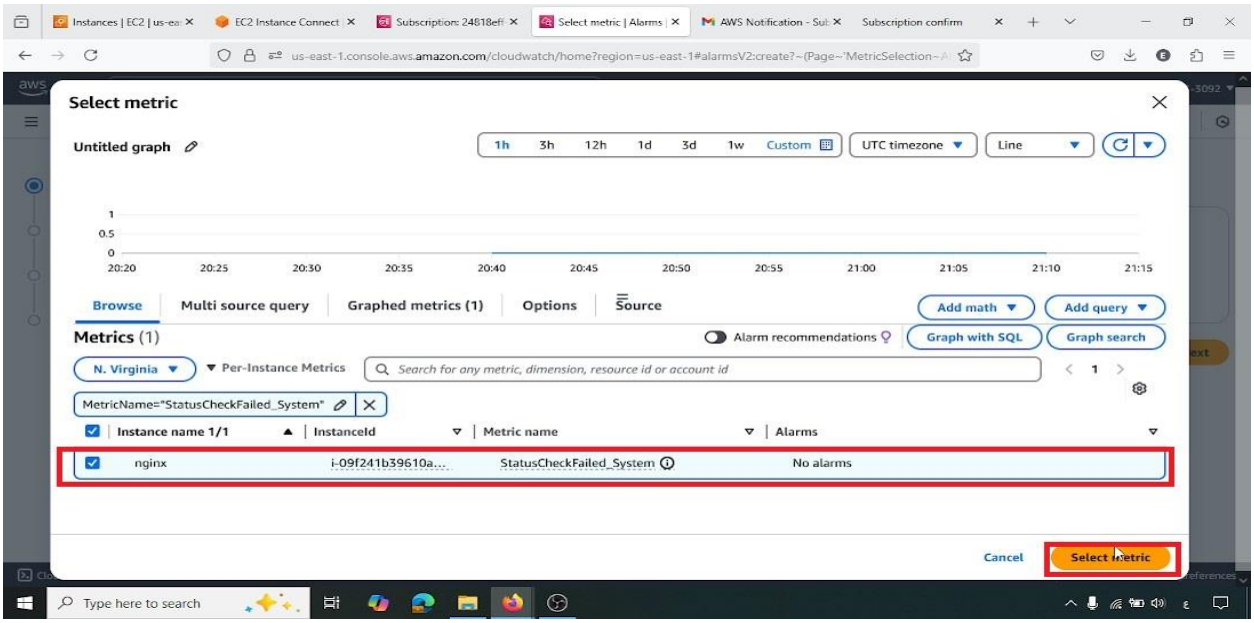


45: Choose System Fail Alarm Within the EC2 metrics, the StatusCheckFailed_System metric is chosen. This metric specifically monitors system-level failures that could indicate underlying hardware issues or problems with the EC2 host, prompting the need for recovery.

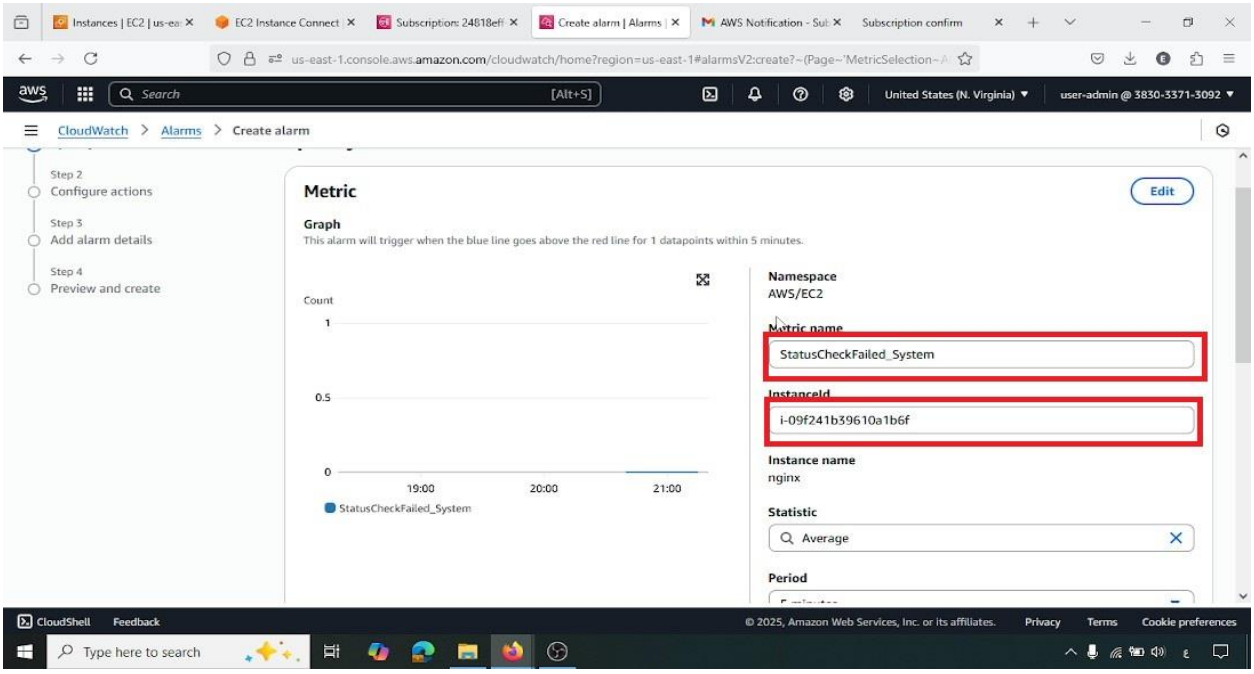


BY : EbRam

46: Then Click Select Metric After selecting the StatusCheckFailed_System metric, this image shows clicking "Select metric" to confirm the choice and proceed to configure the alarm conditions based on this specific metric.

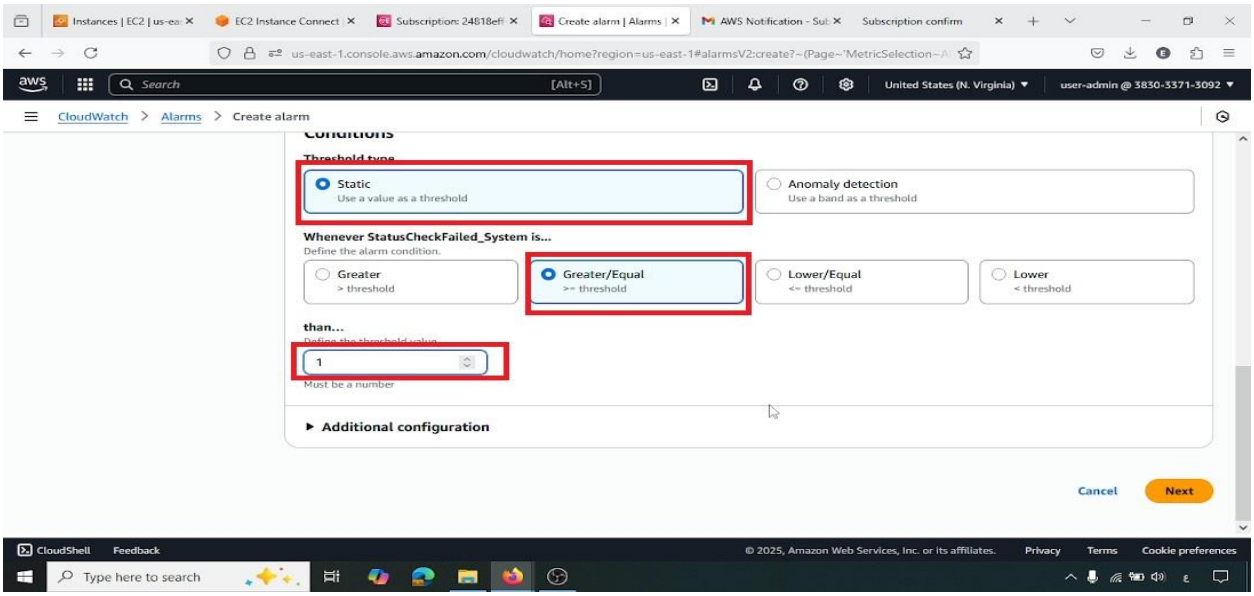


47: Now We Choice the Instance That We Need to Make This Alarm for It This screenshot illustrates the selection of the particular NGINX EC2 instance for which the StatusCheckFailed_System alarm will be configured. This ensures that the alarm monitors the health of the correct server.

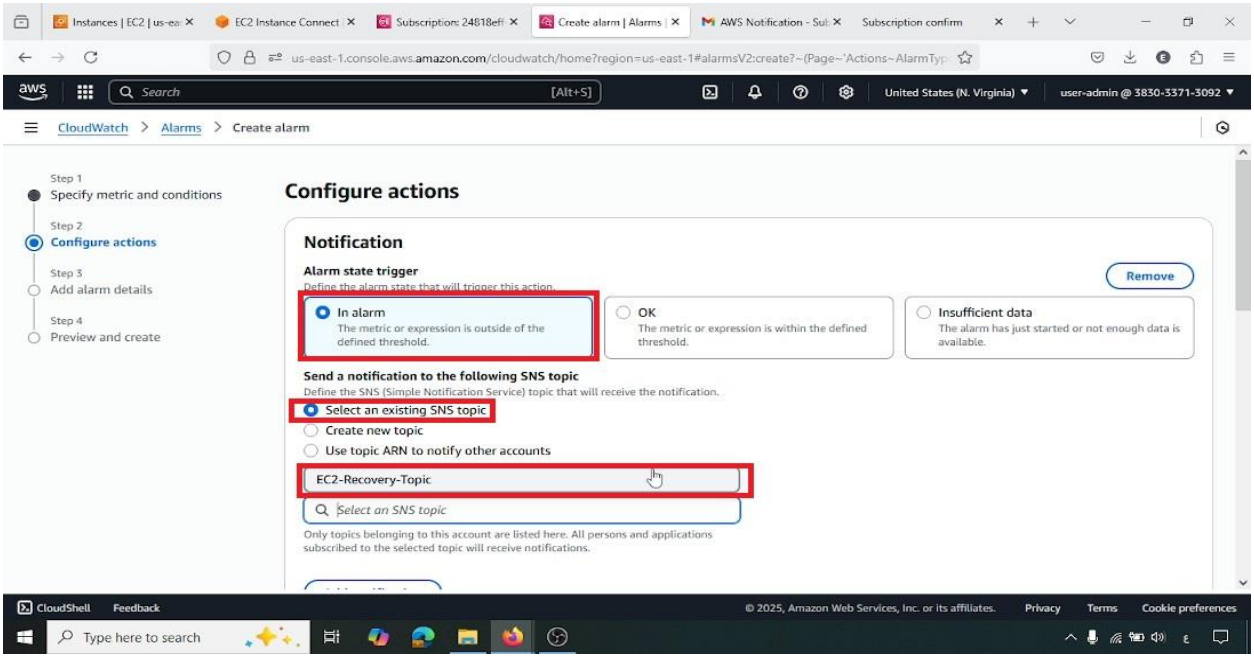


BY : EbRam

48: Select Static Then Equal or Greater Than 1 Threshold Here, the alarm condition for StatusCheckFailed_System is configured. It is set to a "Static" threshold, triggering when the metric is "Greater/Equal" to "1" over a specified period, indicating a system check failure has occurred.

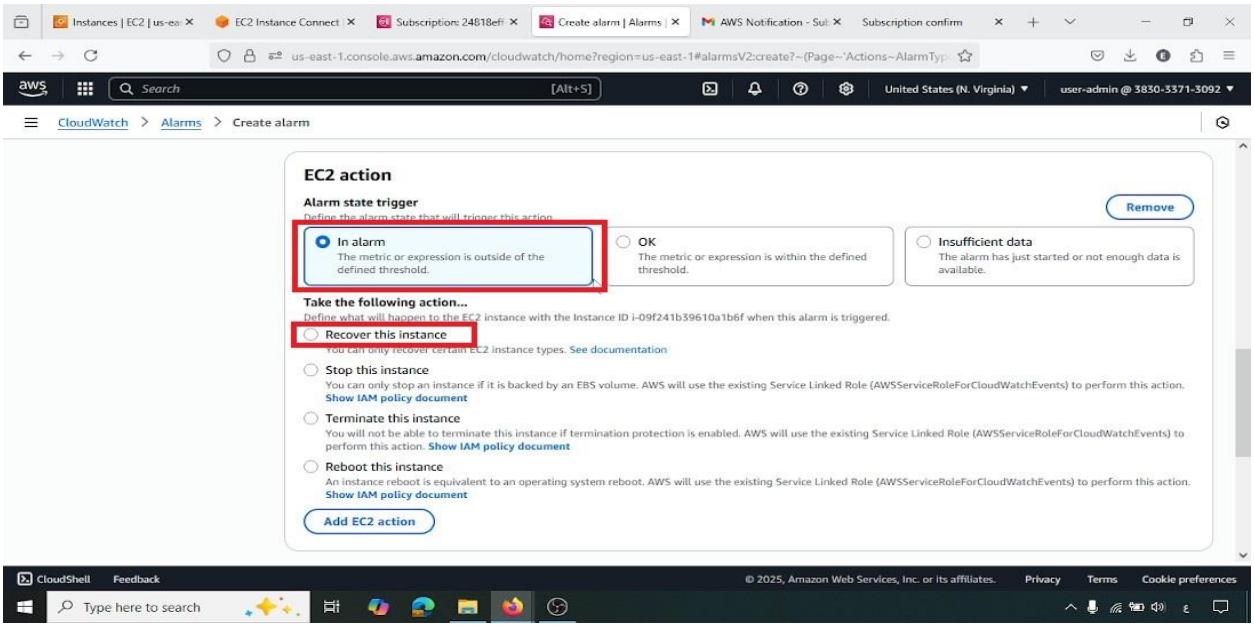


49: Select Notif and Put the Topic That We Created Before This image shows the configuration of the alarm actions. When the alarm transitions to an "In alarm" state, it is configured to send a notification to the previously created EC2-Recovery-Topic SNS topic, alerting administrators.

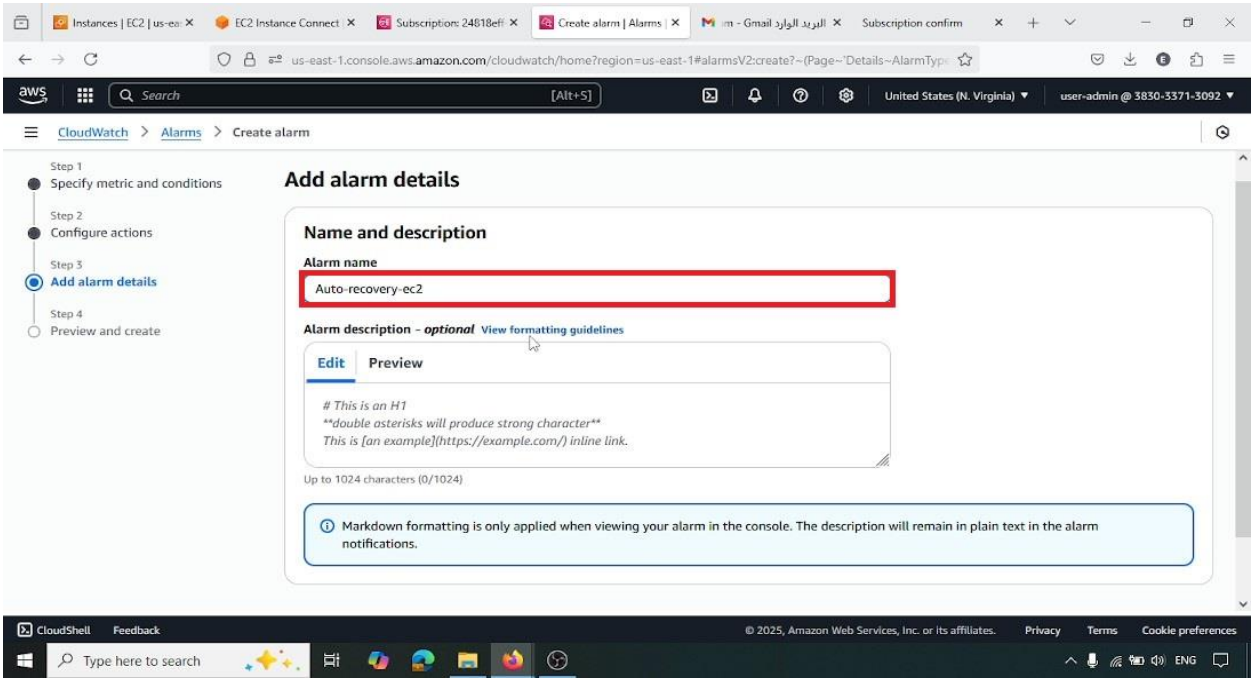


BY : EbRam

50: In EC2 Action Choose Instance Recovery Crucially, this screenshot shows configuring an EC2 action for the Auto-recovery-ec2 alarm. When the alarm is triggered, it will automatically perform an "Instance recovery" action, attempting to fix the impaired instance by moving it to healthy underlying hardware.

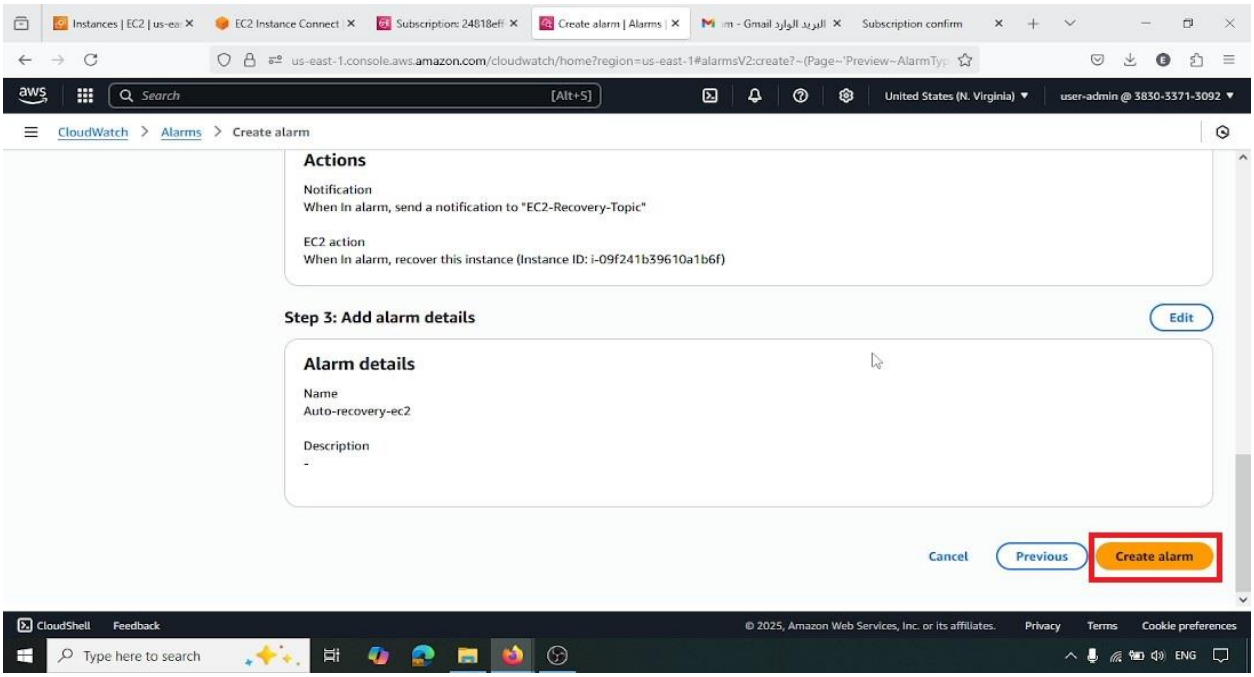


51: Give the Alarm Name The alarm is named Auto-recovery-ec2, a descriptive name that clearly indicates its purpose: to automatically recover the EC2 instance upon a system check failure.

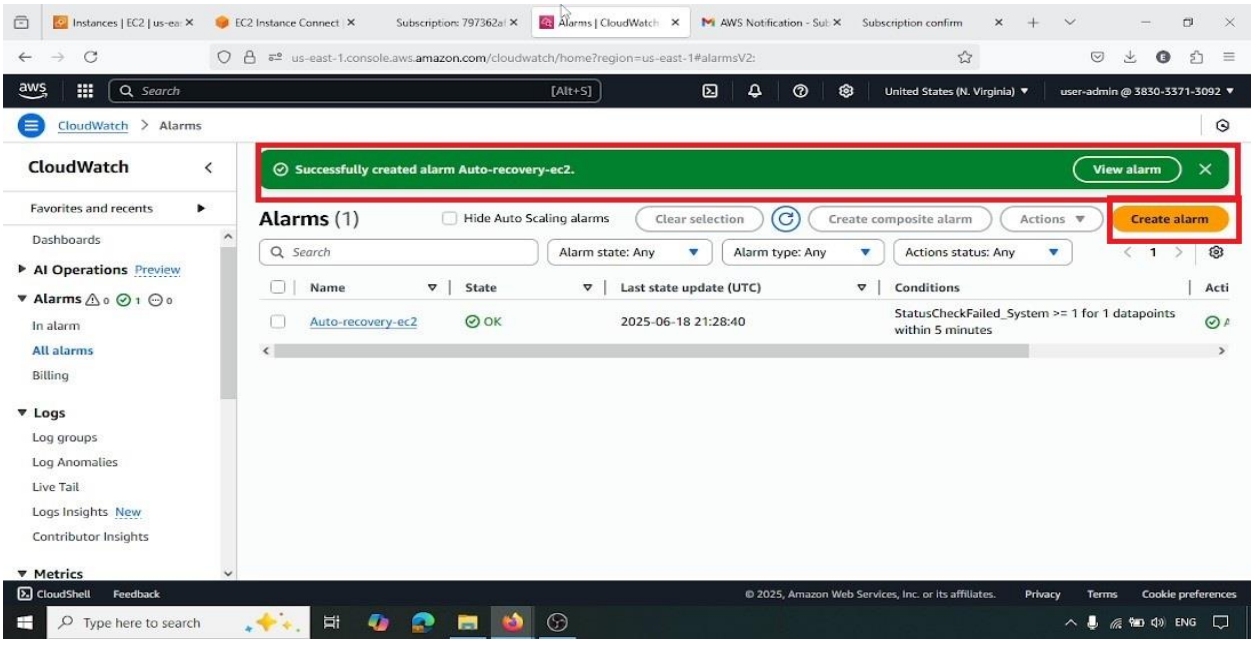


BY : EbRam

52: Then Click Create Alarm This image shows the final step in creating the Auto-recovery-ec2 CloudWatch alarm. Clicking "Create alarm" registers the alarm, activating its monitoring capabilities and recovery actions.

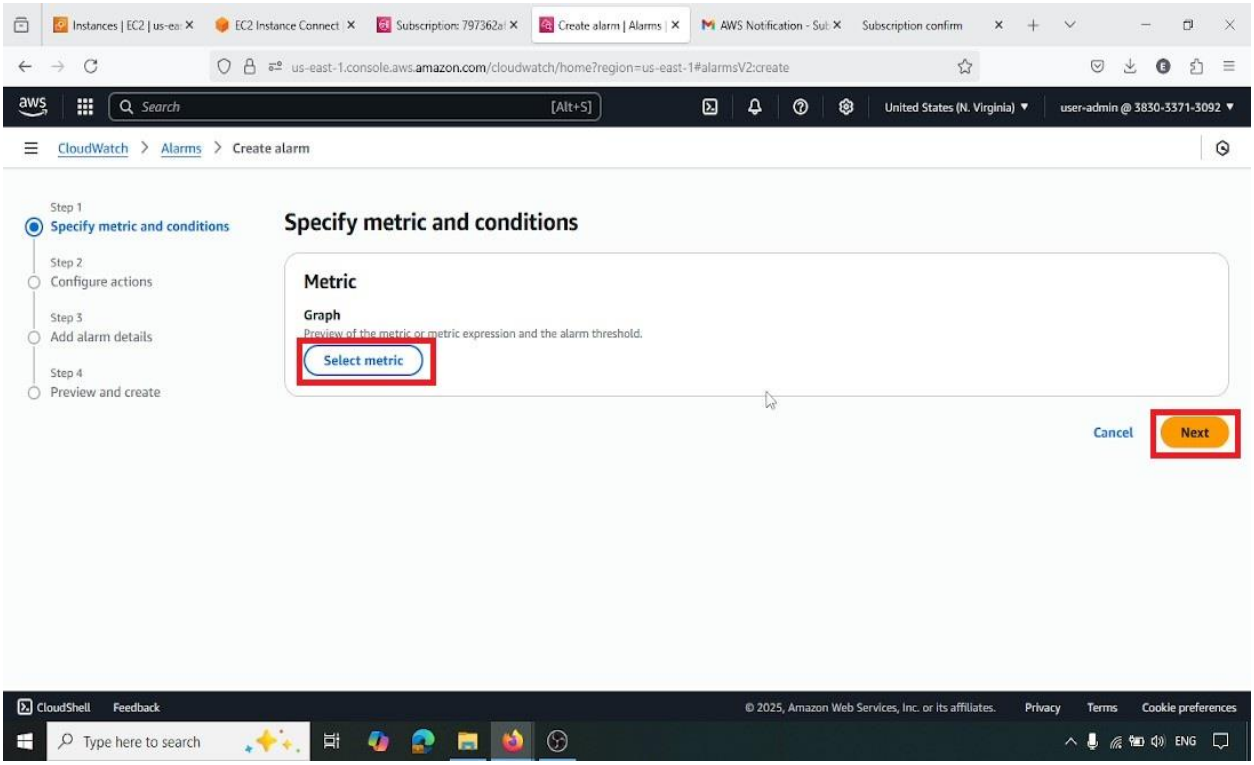


53: Go to CloudWatch and Click Create Alarm This image, positioned slightly out of general flow, again shows navigating back to CloudWatch and initiating the creation of another alarm, this time for CPU utilization.

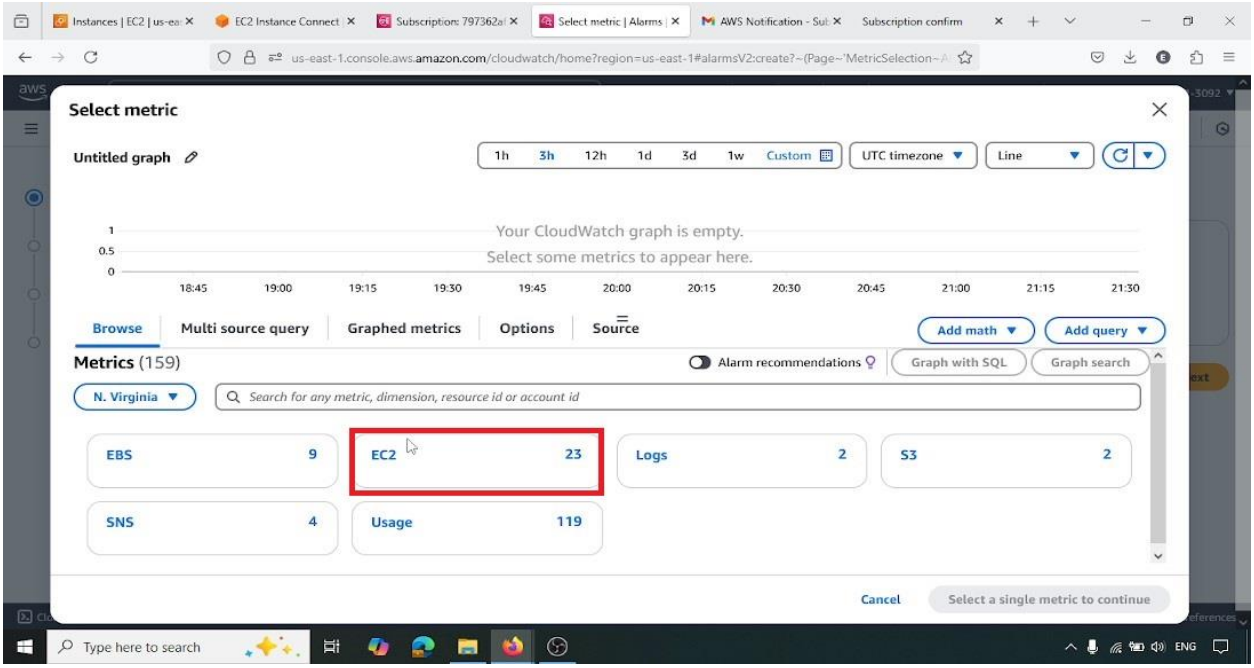


BY : EbRam

54: Select Metric Similar to the recovery alarm, the "Select metric" option is chosen to define the basis of the new CPU utilization alarm.

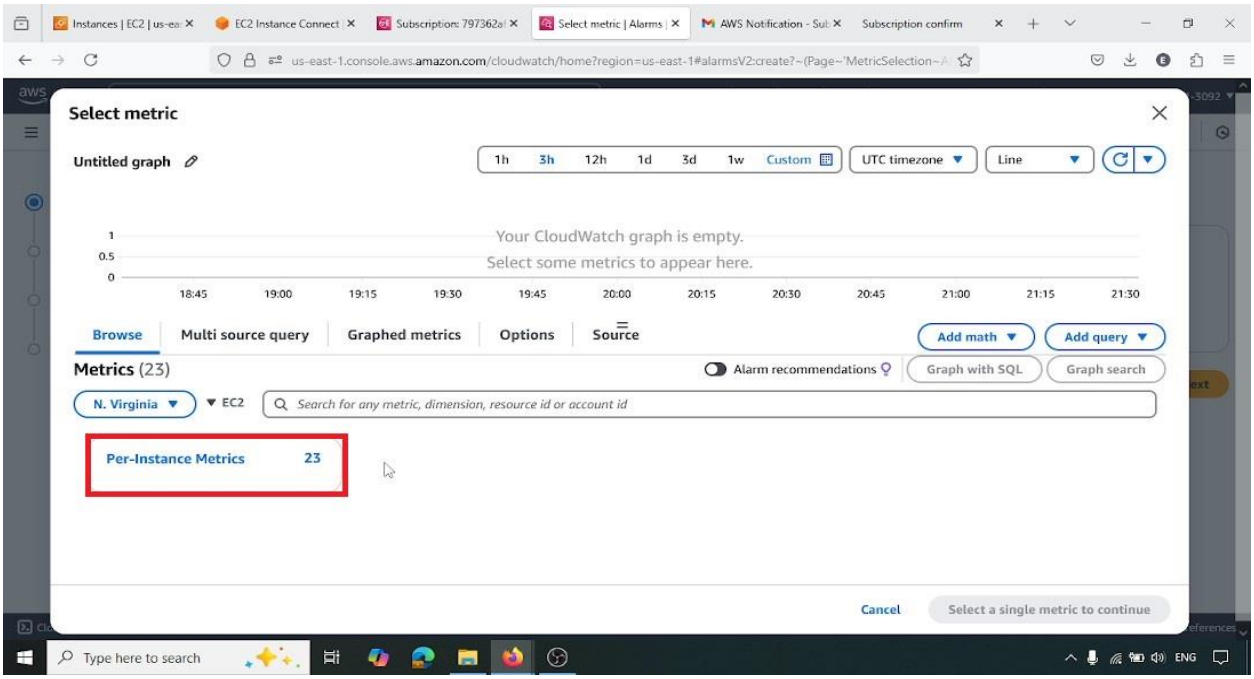


55: Choice EC2 The "EC2" service is selected from the metric categories, narrowing down the options to metrics relevant to EC2 instances.

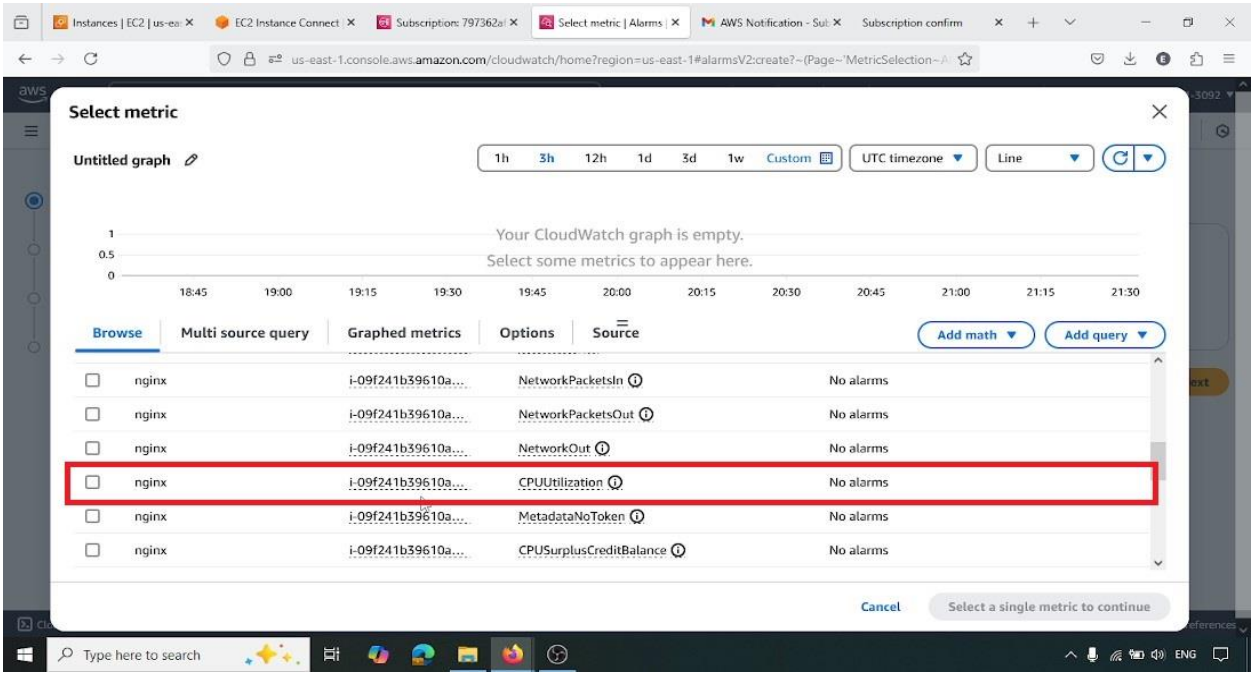


BY : EbRam

56: Choose-Per-Instance This image indicates selecting "Per-Instance Metrics" within the EC2 category, which allows monitoring of individual EC2 instance performance characteristics.

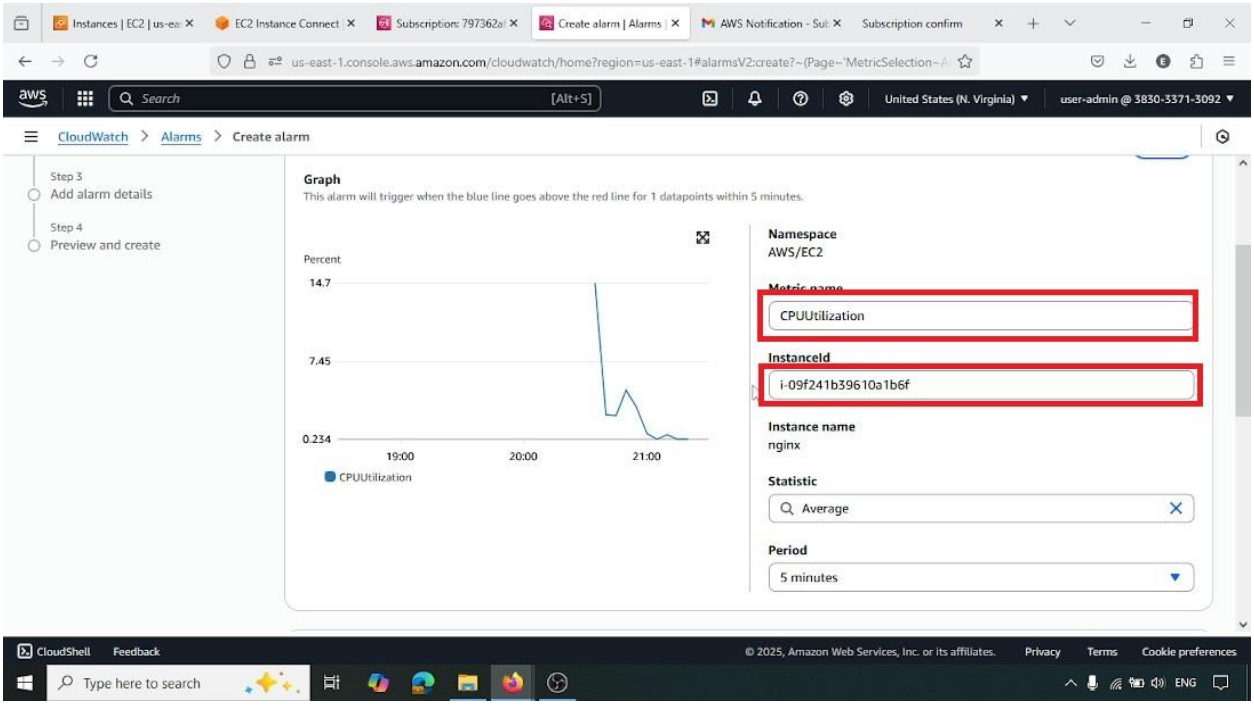


57: Choose CPU Utilization From the per-instance metrics, CPUUtilization is chosen as the specific metric for this new alarm. This will allow the alarm to monitor the percentage of allocated EC2 compute units that are currently in use.

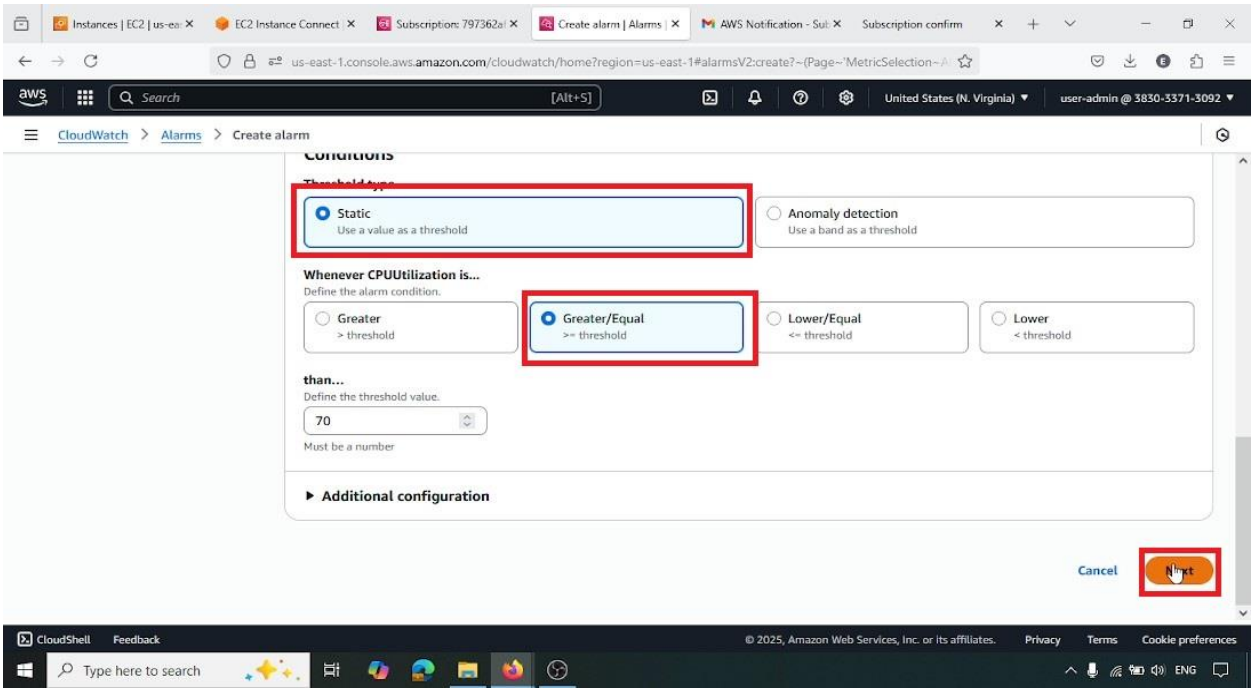


BY : EbRam

58: Select Instance ID The specific NGINX EC2 instance ID is selected to ensure that the CPU utilization alarm monitors the correct server's performance.

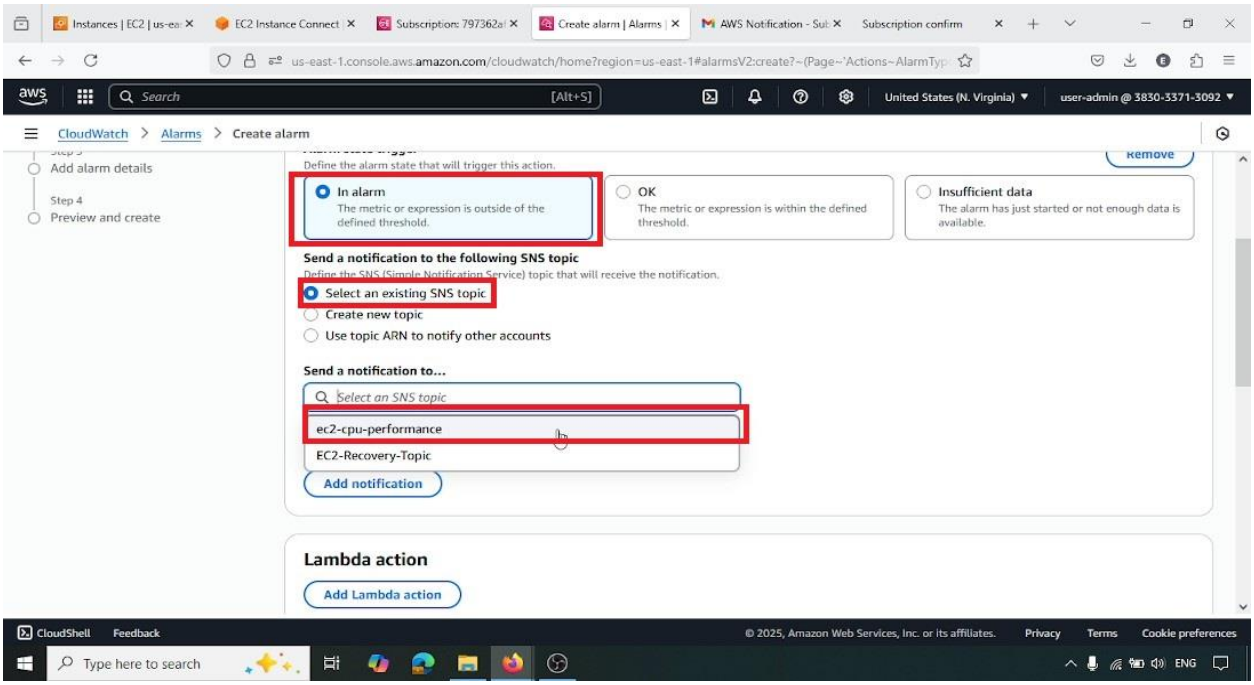


59: Select Static Then Greater or Equal and Usage Type 70% The alarm condition for CPUUtilization is set to "Static," triggering when the CPU utilization is "Greater or Equal" to "70%" over a defined period. This threshold indicates high resource consumption.

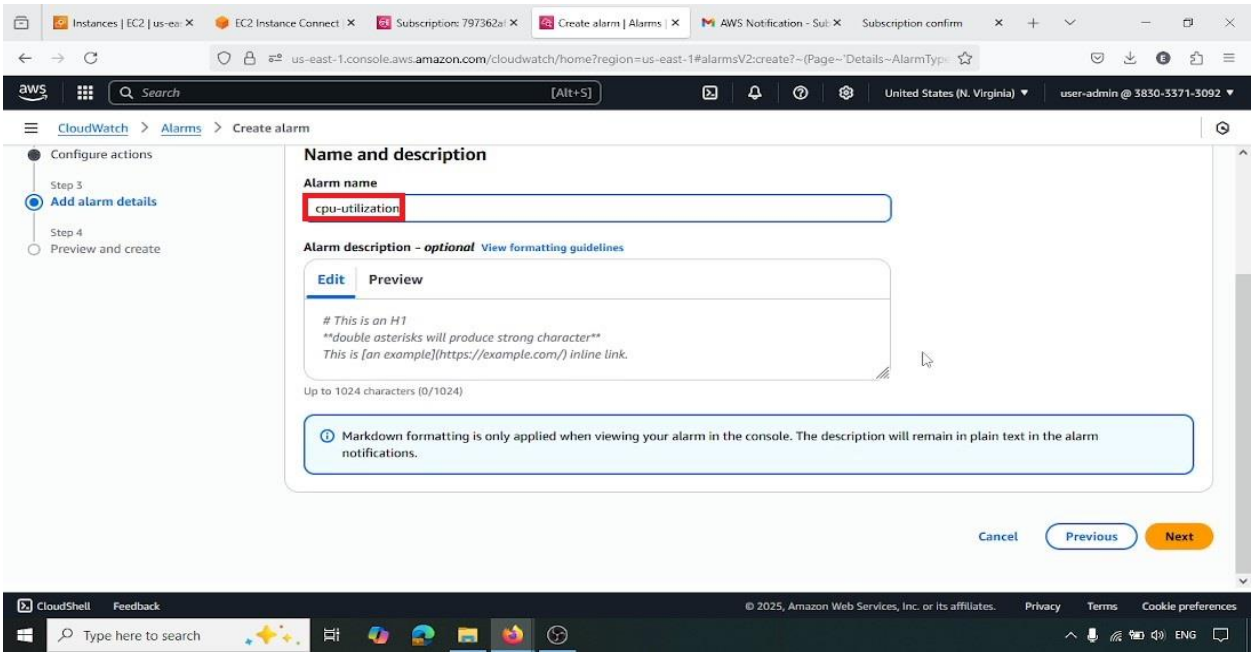


BY : EbRam

60: Choice In Alarm Then Select the Topic of SNS When the CPU utilization alarm transitions to the "In alarm" state, it is configured to send a notification. This image shows the selection of the previously created ec2-cpu-performance SNS topic as the destination for these alerts.

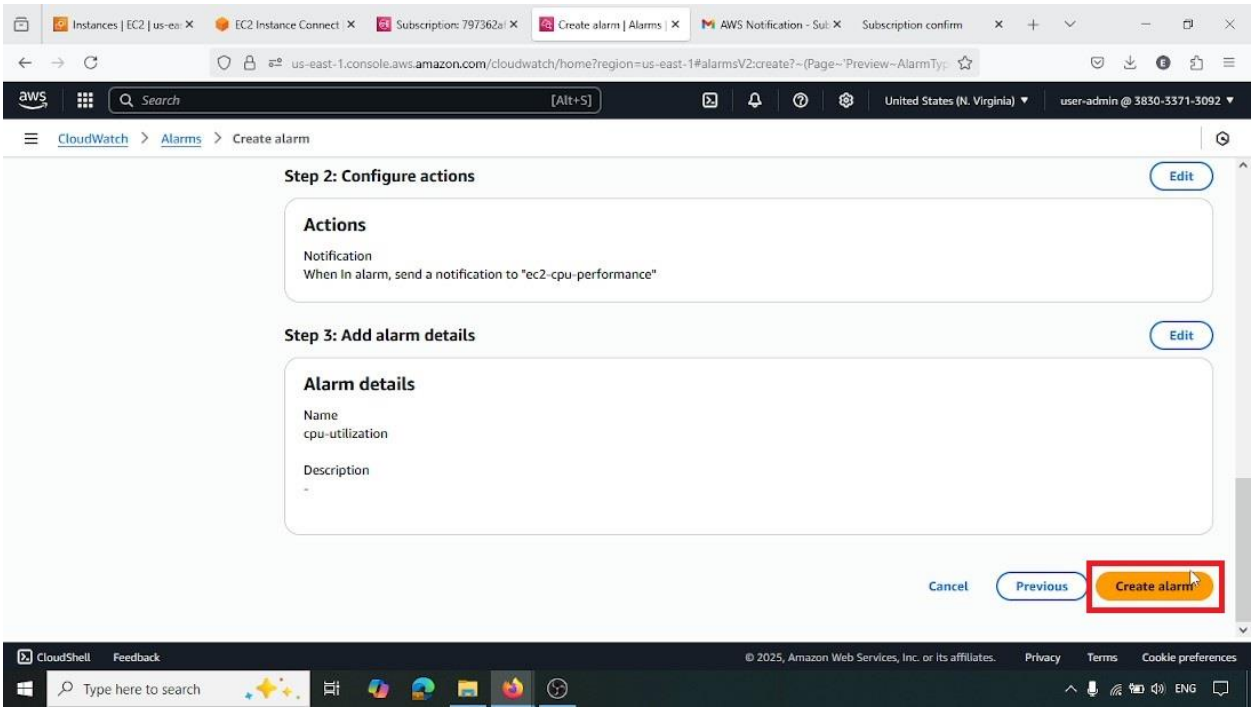


61: Write the Name of Alarm Then Next The CPU utilization alarm is named cpu-utilization, and the "Next" button is clicked to proceed with the alarm creation process.



BY : EbRam

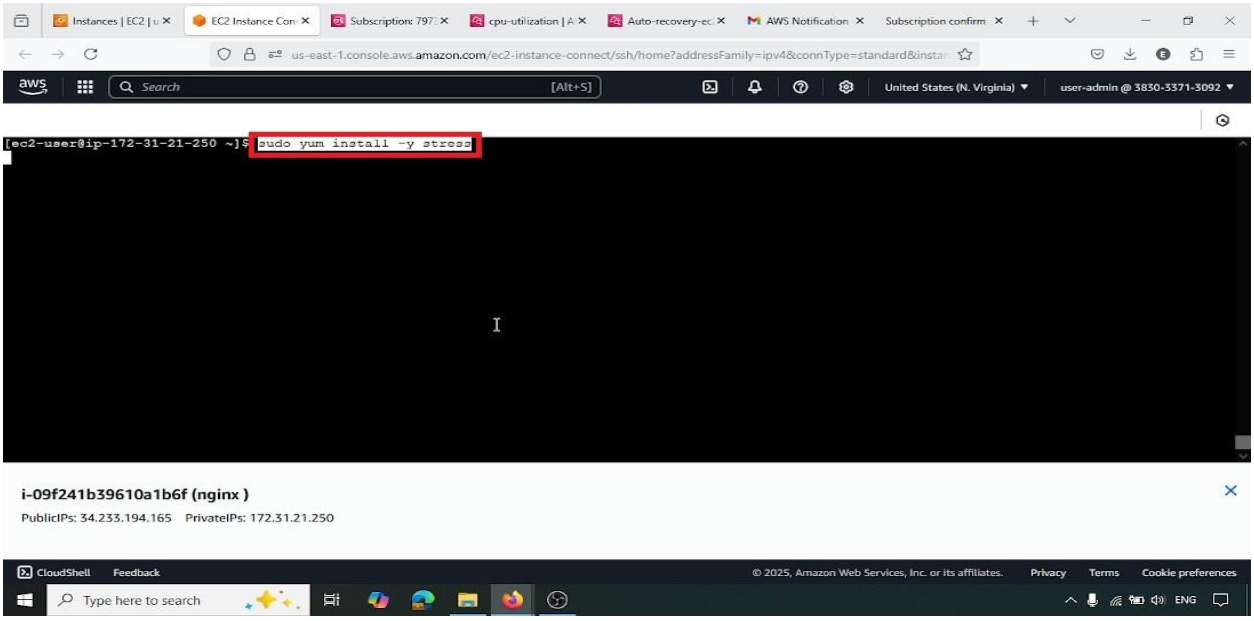
62: Click Create Alarm For... This final image shows the action of clicking "Create alarm," completing the setup for the cpu-utilization CloudWatch alarm. This alarm will now actively monitor the EC2 instance's CPU usage and notify administrators if it exceeds the specified threshold.



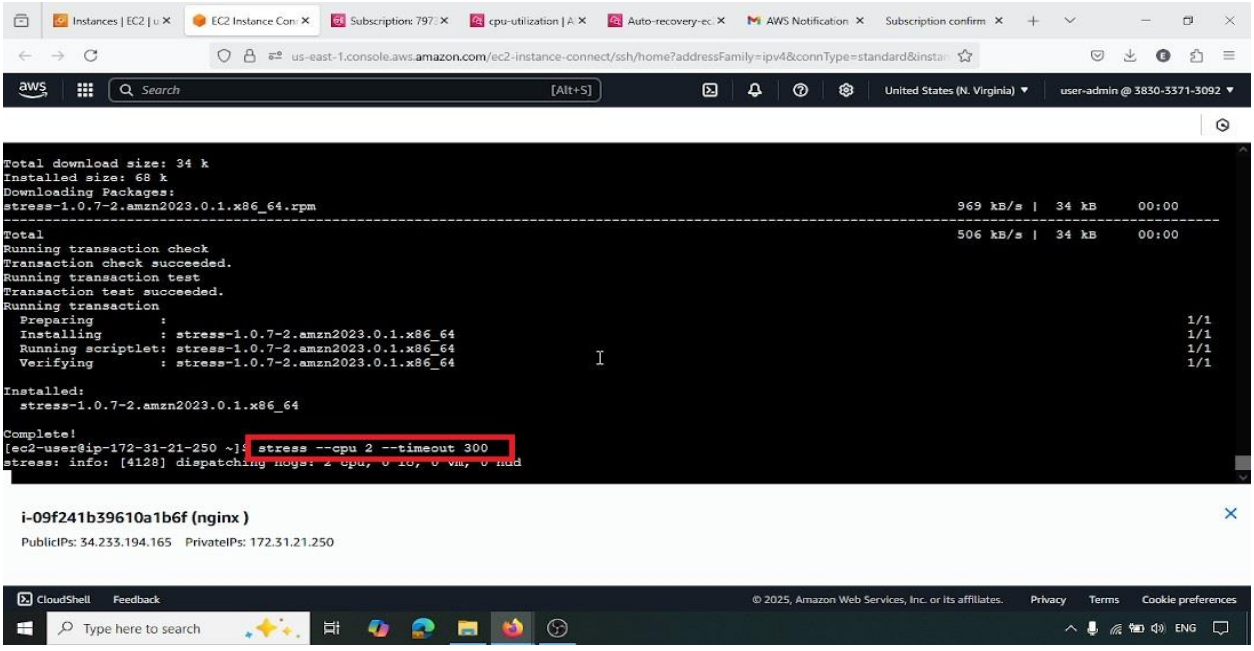
BY : EbRam

5. Testing Alarms

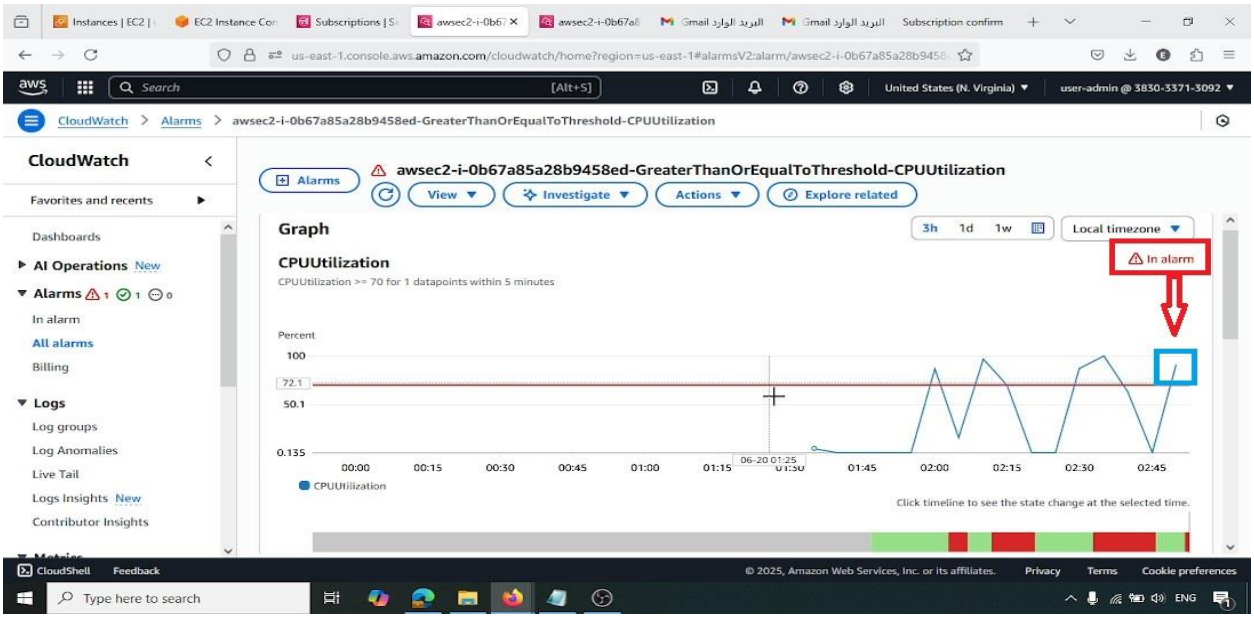
63: Install Stress Tool This screenshot shows the command sudo yum install -y stress being executed on the EC2 instance. The stress tool is a utility used to generate controllable workloads, which is essential for testing the CPU utilization alarm.



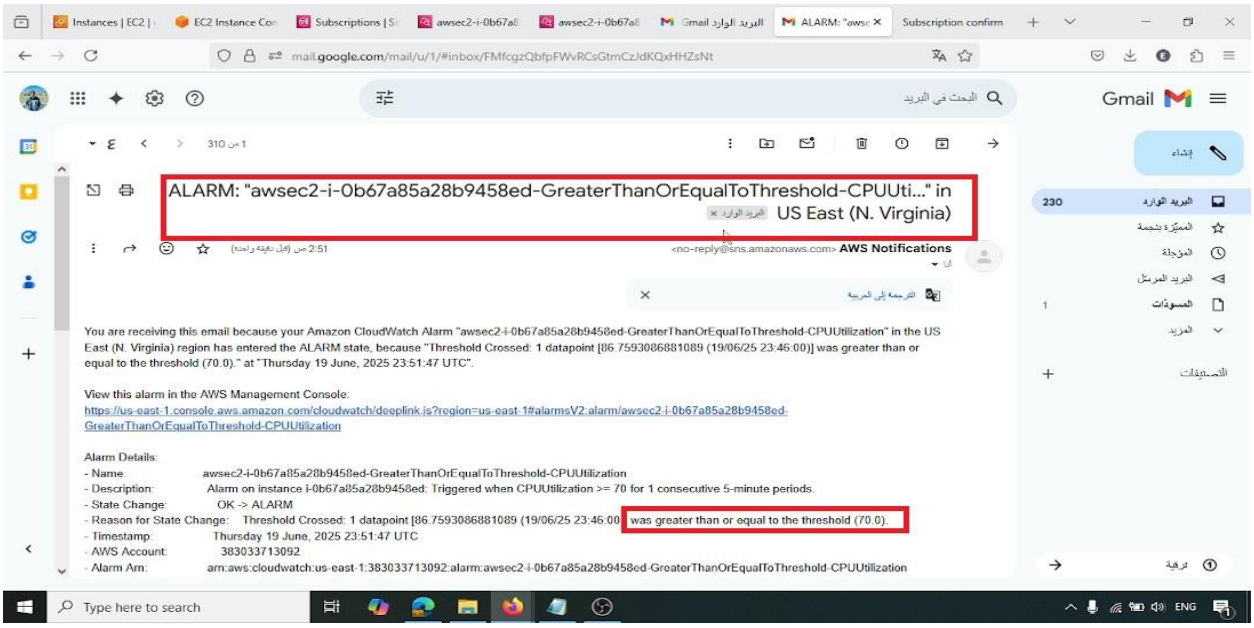
64: Stress the CPU The command stress --cpu 2 --timeout 300 is shown being run. This command intentionally puts a heavy load on two CPU cores for 300 seconds, simulating a high CPU usage scenario to trigger the cpu-utilization CloudWatch alarm.



65: CloudWatch Alarm State (In Alarm) This screenshot of the CloudWatch dashboard shows that the cpu-utilization alarm has successfully transitioned to the "In alarm" state. The graph visually confirms that the CPU utilization has exceeded the 70% threshold, validating the alarm's configuration.



66: Check Notification Email This image displays an email notification received from AWS, confirming that the cpu-utilization alarm has entered the "ALARM" state. This verifies that the SNS topic and email subscription are functioning correctly, providing timely alerts for performance issues.



BY : EbRam