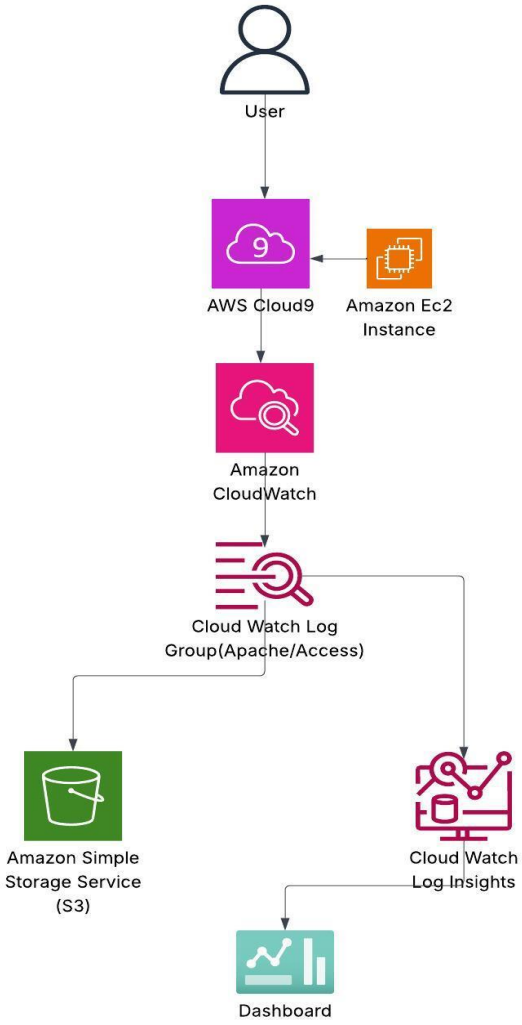
PHASE1:

Task 1: Creating an architectural diagram



In this architecture, a user connects to an Apache Web Server running on an Amazon EC2 instance through AWS Cloud9. The CloudWatch Agent installed on the EC2 instance collects application and system logs, sending them to a CloudWatch Log Group (specifically for Apache access logs). These logs are then analyzed using CloudWatch Logs Insights for deep querying and troubleshooting. Data from the Log Group is visualized through a CloudWatch Dashboard, providing key operational metrics. Additionally, important log data is backed up to an Amazon S3 bucket for long-term storage and disaster recovery purposes, ensuring data durability and accessibility.

Task 2: Developing a cost estimate

AmazonEC2(cloud 9 instance)-0.21USD

Amazon S3(log storage) for 5GB -1.39USD

Amazon Athena 10 GB queries per month-0.05USD

IAM Role Free(project):0USD

Amazon cloud watch logs (1 dashboard) -2.55USD

Amazon cloud9 charged only for EC2 usage

Amazon EC2 Instance configuration:

*We should select shared instance

*Select Instance type t3.mcro

*Pricing: On-demand

Amazon S3 configuration:

Region:us-east-1
Storage class: S3 standard

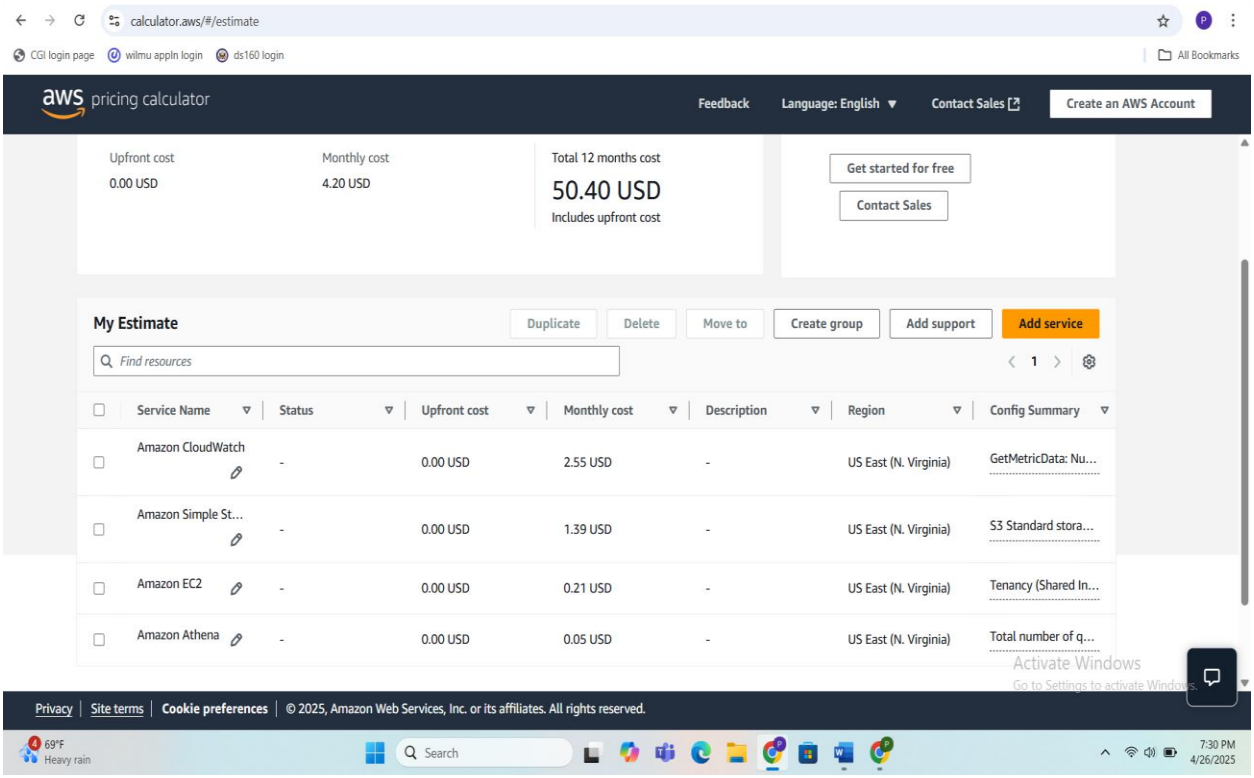
PUT, COPY, POST, LIST requests:10,000 per month

GET, SELECT and all other requests:10,000 per month

Data transfer: Minimal (0-1)GB

Cloud Watch configuration:

Get metric data API:1000 per month
Number od dashboard:1
Number of standard resolution alarms:0



PHASE2: Analyzing the website and confirming weblog data

Task 1: Analyzing and understanding the lab environment

VPC Setup: We have a Lab VPC with a Public Subnet connected to the internet via an Internet Gateway, allowing access from the cloud.

AWS Cloud9: A Cloud9 instance (running as an EC2) in the public subnet provides an integrated IDE for editing code and running AWS CLI commands.

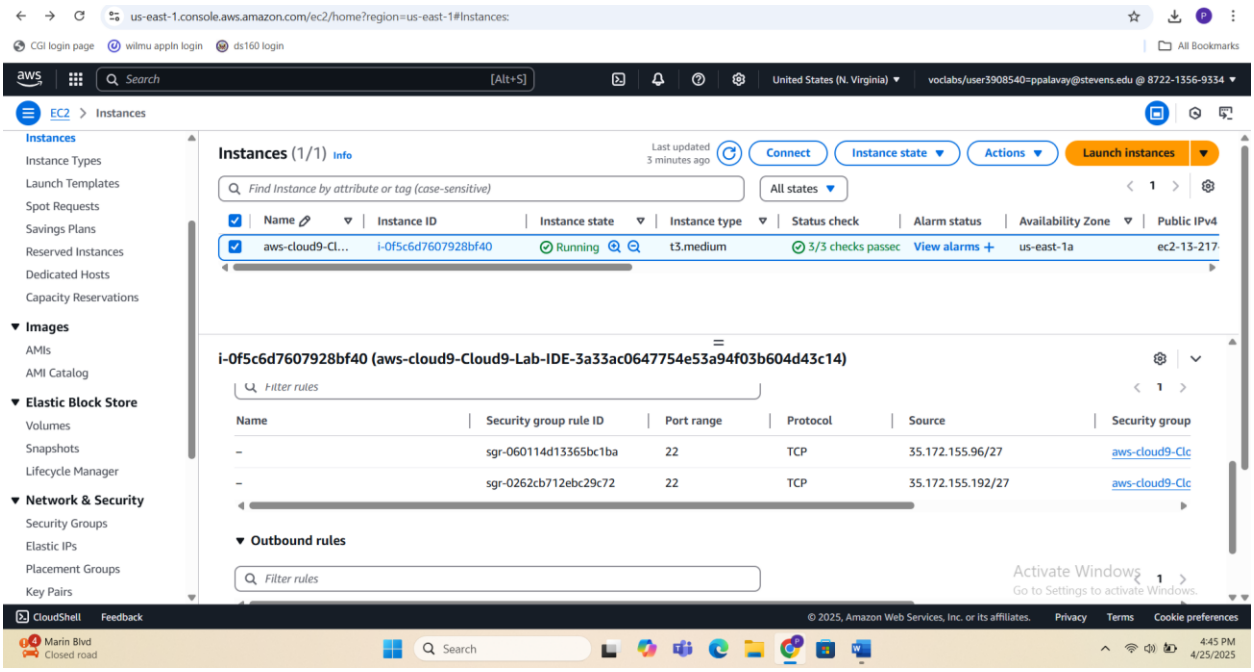
Security Group: The EC2 instance has a security group controlling access, ensuring only necessary traffic (e.g., HTTP, HTTPS, MySQL) can reach the web application.

IAM Role (CafeRole): The CafeRole IAM role is attached to the EC2 instance, granting it access to other AWS services like Amazon S3, CloudWatch, and Systems Manager.

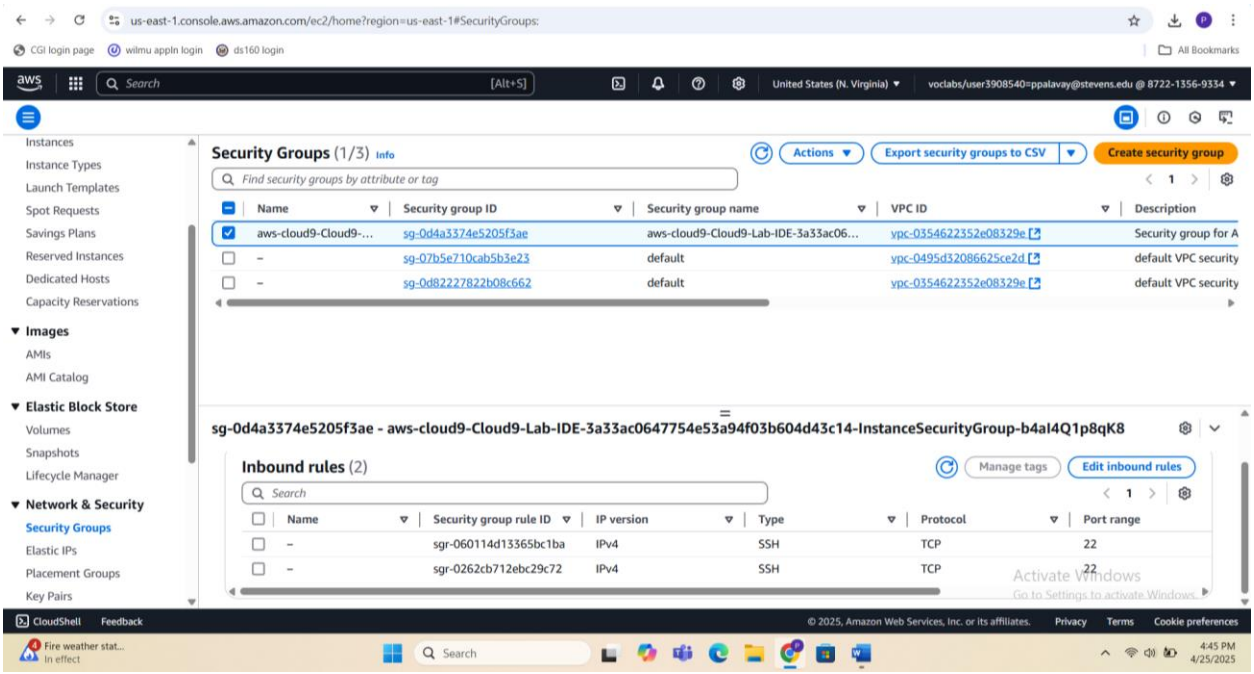
Web Application: The café web app runs on Apache (HTTP server) and PHP, with a MariaDB database hosted on the same EC2 instance for storage and data management.

Task 2: Modifying a security group and verifying that the web application loads

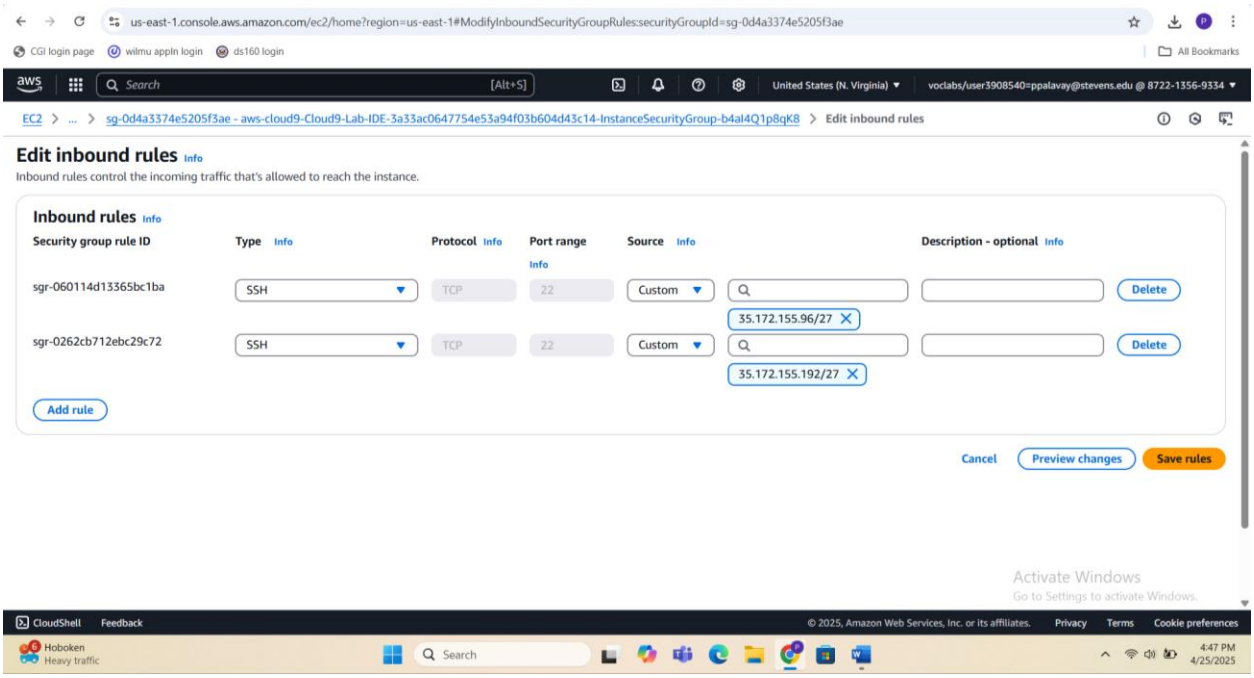
Go to Amazon EC2 Dashboard.



Select your instance and click on the **Security group** linked to it



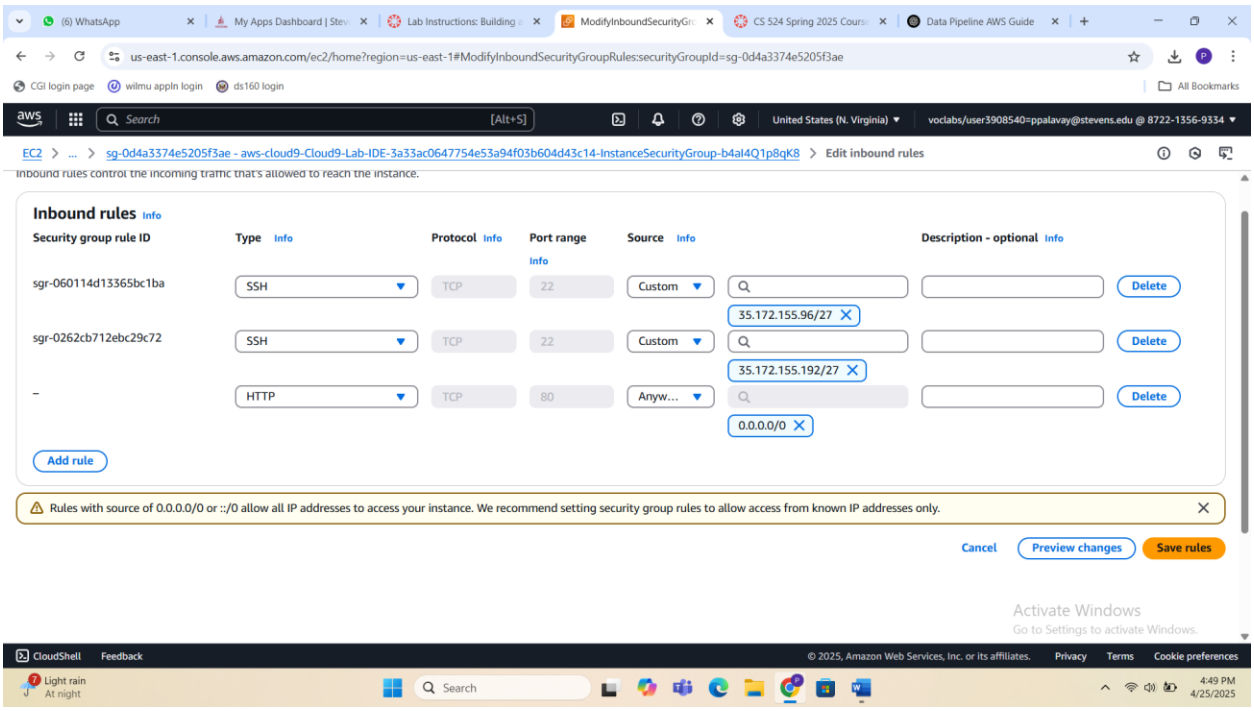
Choose the **Inbound rules** tab.



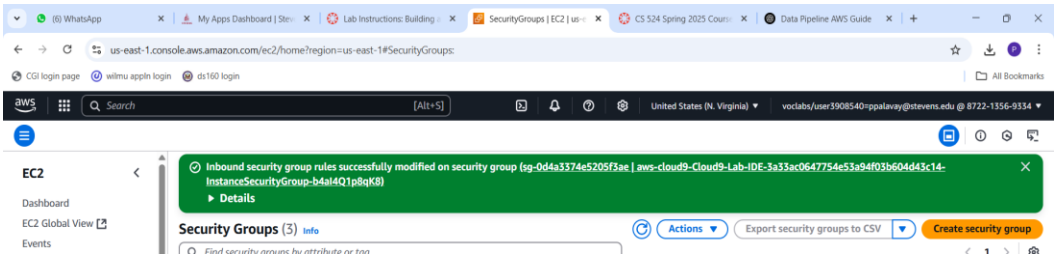
Click **Edit Inbound Rules**.

Add this rule:

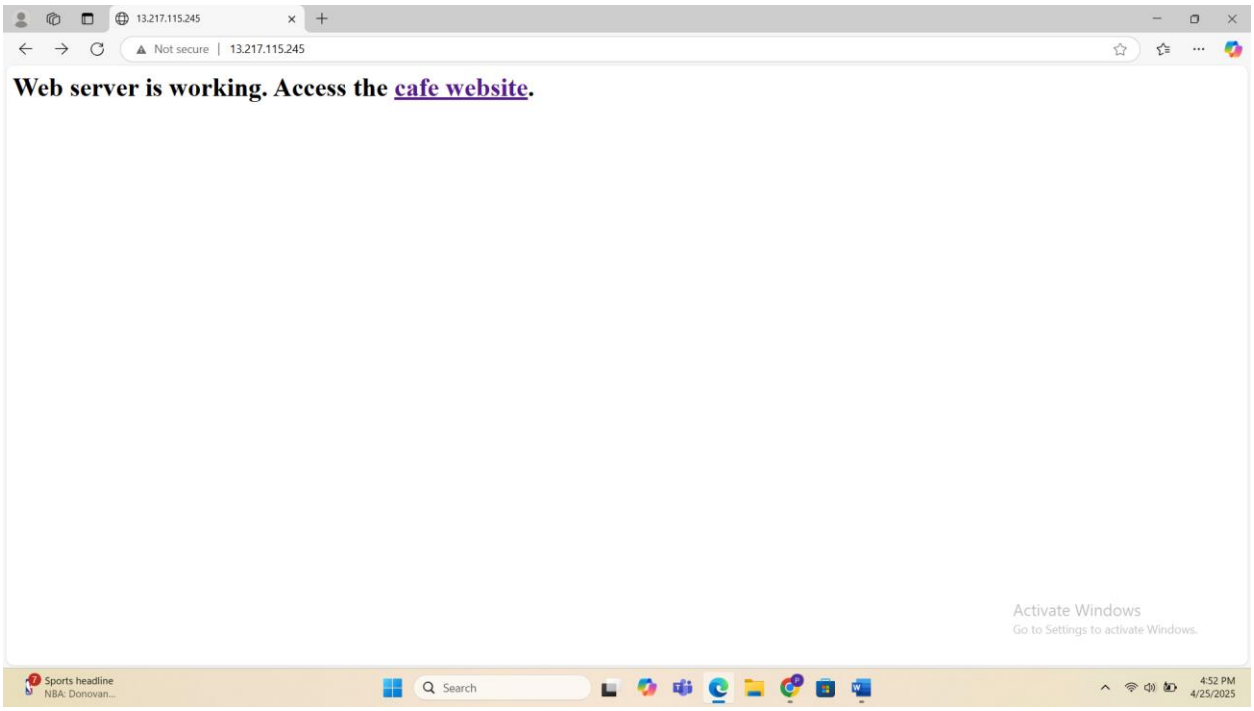
- **Type**: HTTP
- **Protocol**: TCP
- **Port Range**: 80
- **Source**: Anywhere (0.0.0.0/0)



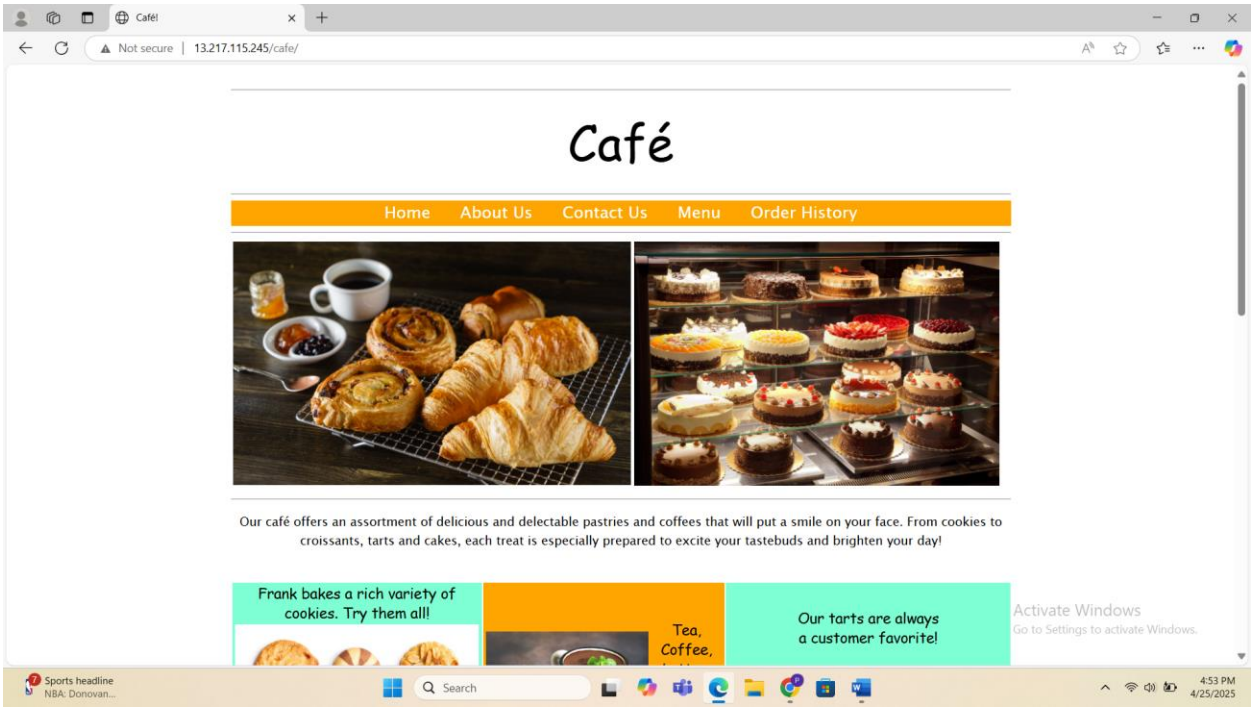
Save the rule.



Go to **EC2 Instances** and get the **Public IPv4 address** of the running instance. Open a browser and give the IP address to **CafeWebsite**.



Click on the website to see menu, order history.

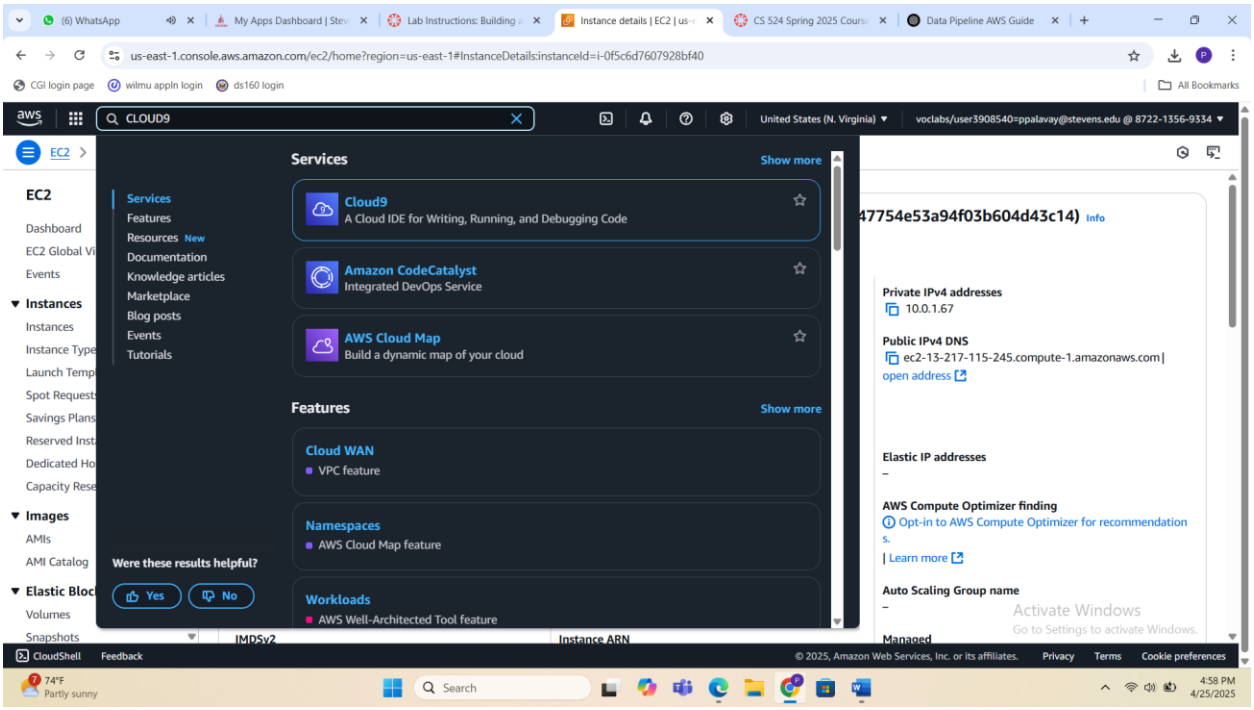


As we can see the **café website**, we can confirm :

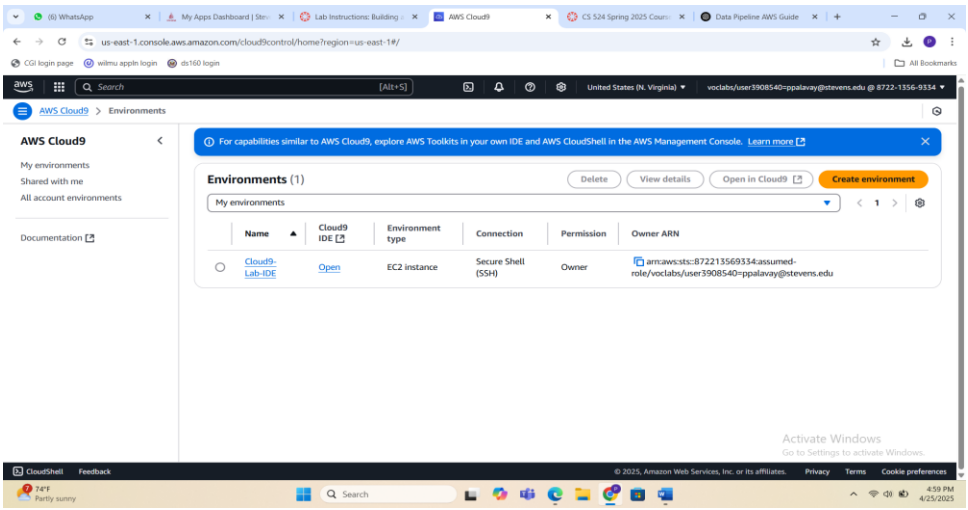
EC2 instance is running
Apache web server is serving the site
Security group is allowing HTTP traffic

Task3: Monitor access logs in real time

Open AWS Cloud9.



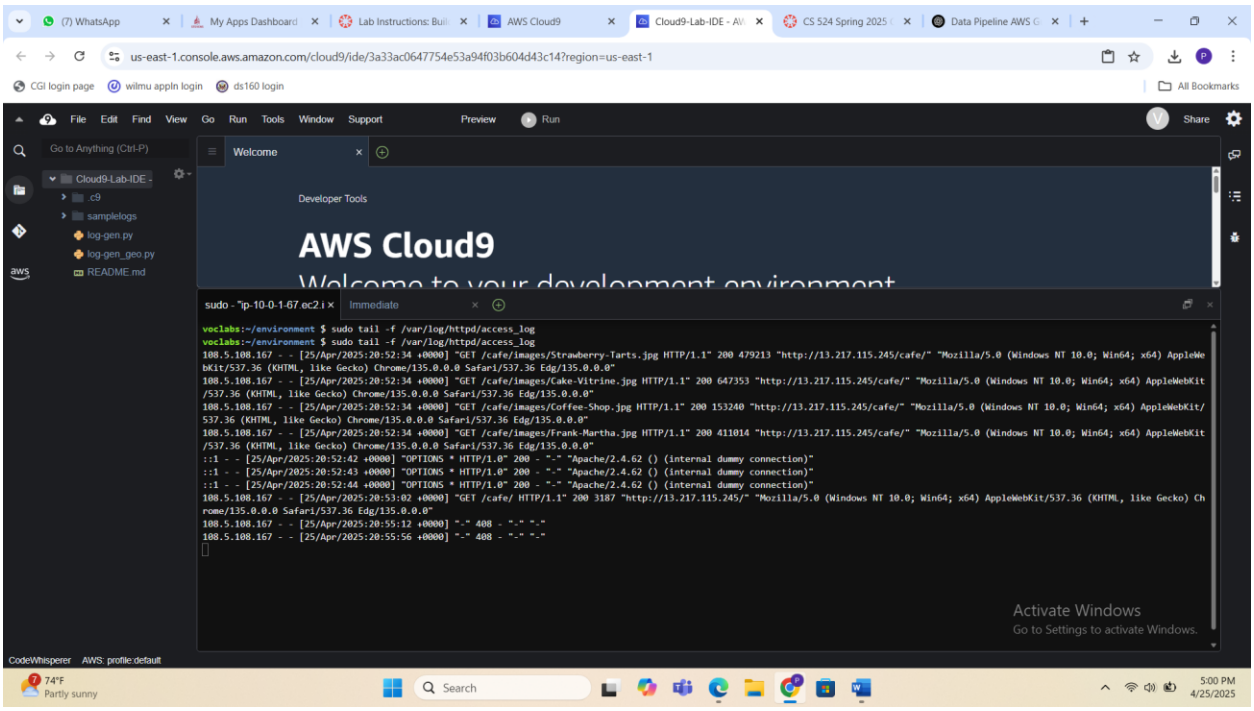
Open the terminal from cloud 9 environment



Run this command

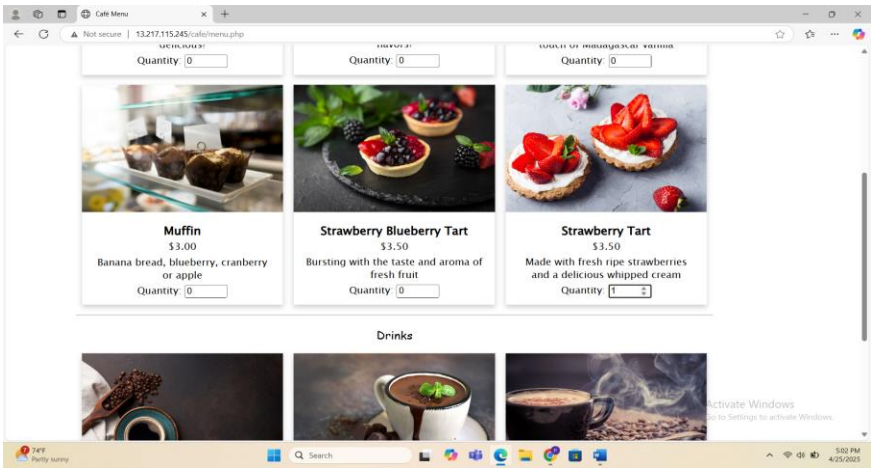
sudo tail -f /var/log/httpd/access_log

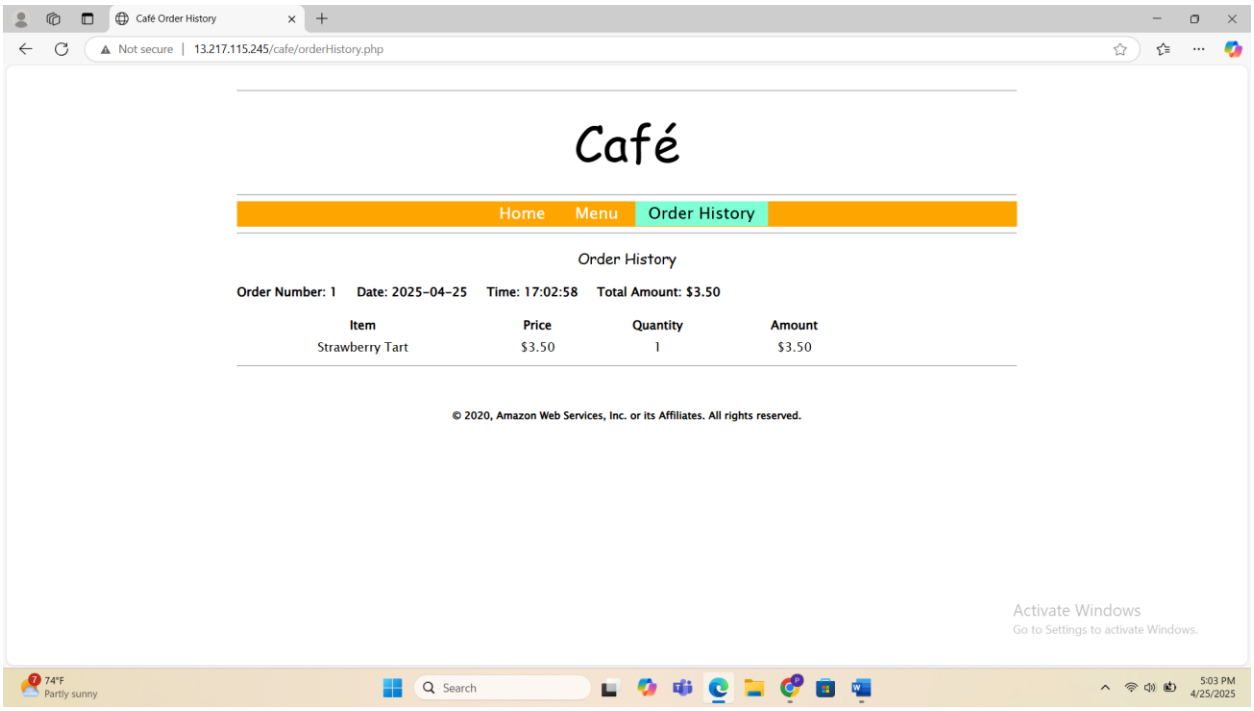
This will display new lines added to the access log **in real-time**



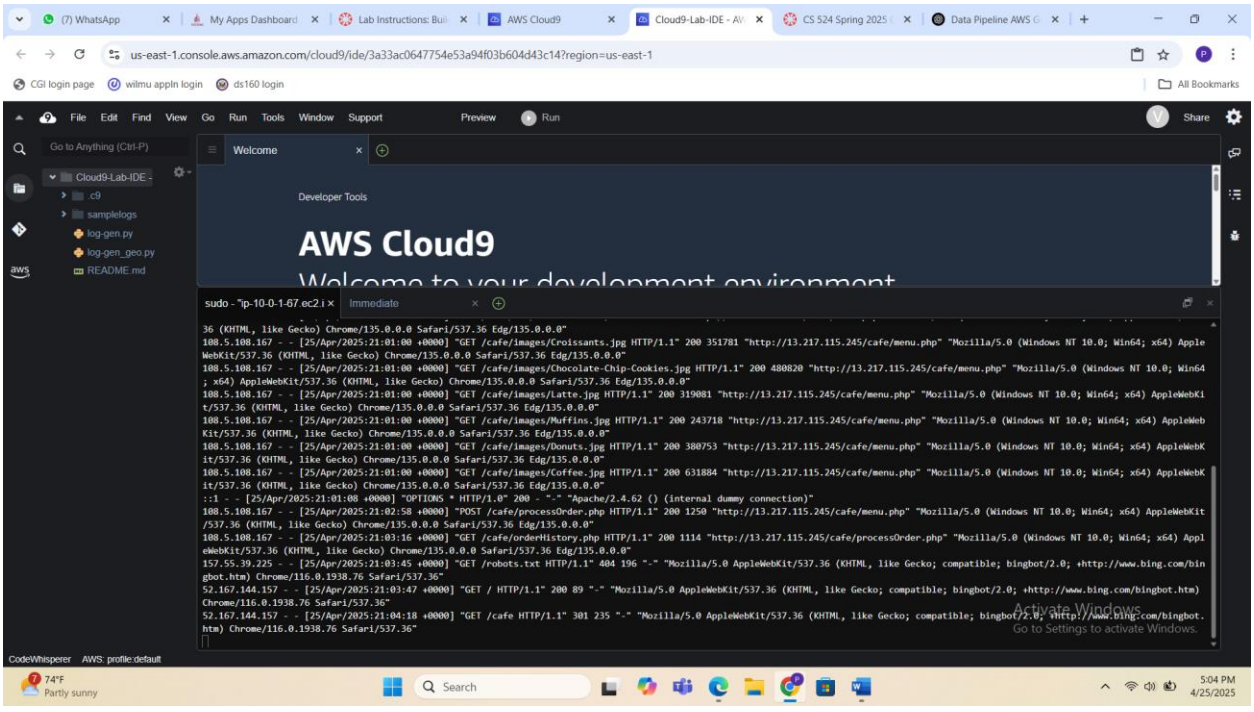
Going back to the browser and interacting with the website:

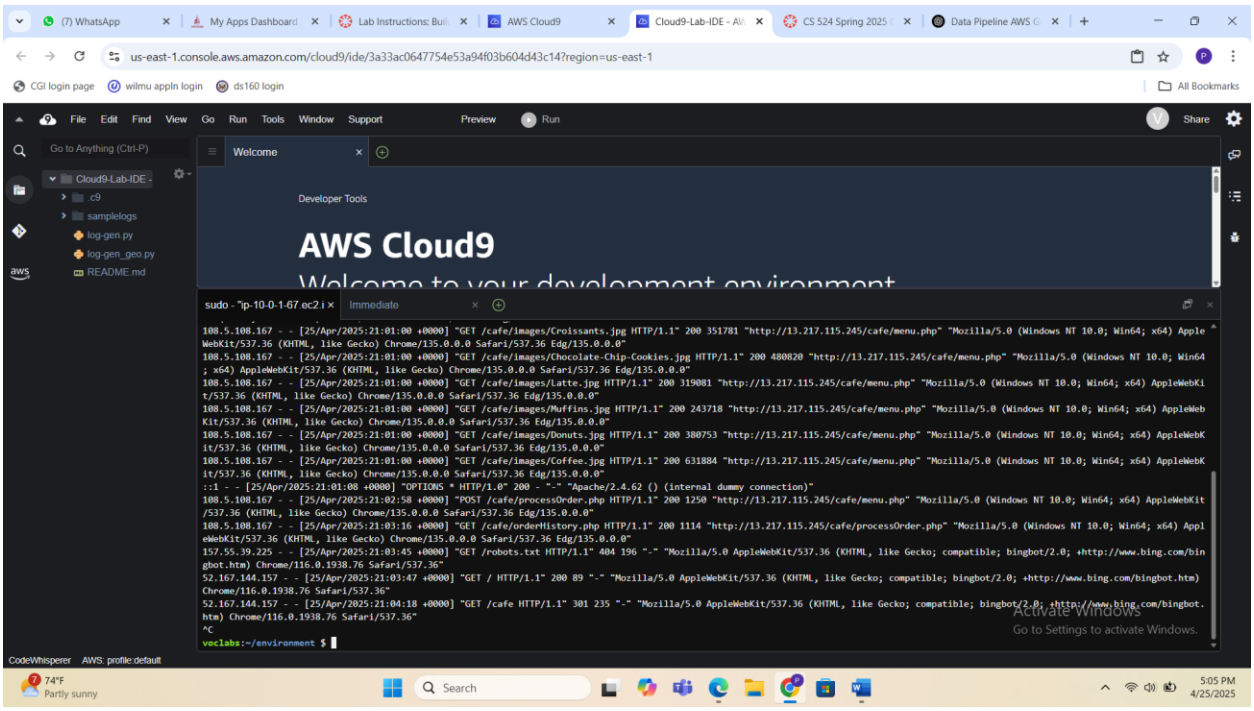
- Visit pages, Submit an order, Refresh the homepage





Each request made to the site (like visiting /cafe/menu.php or /cafe/processOrder.php) will appear as a line in the log file.



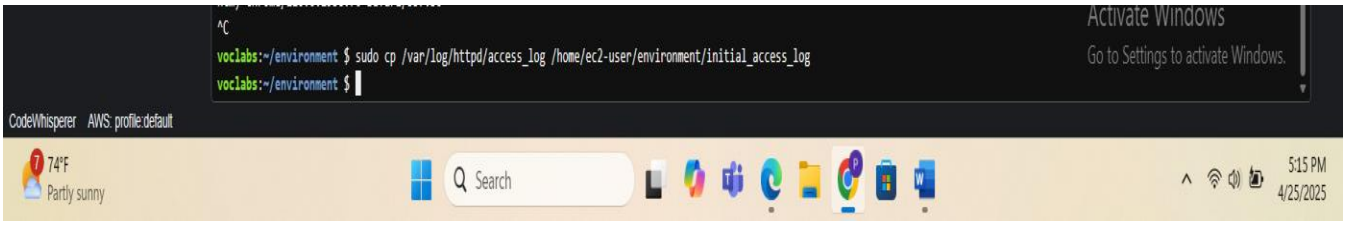


Stop the logs by Ctrl+C

Task4: Backup the Current Log File

Use this command

sudo cp /var/log/httpd/access_log /home/ec2-user/environment/initial_access_log



Keeping a backup gives us a snapshot of the **original raw data**

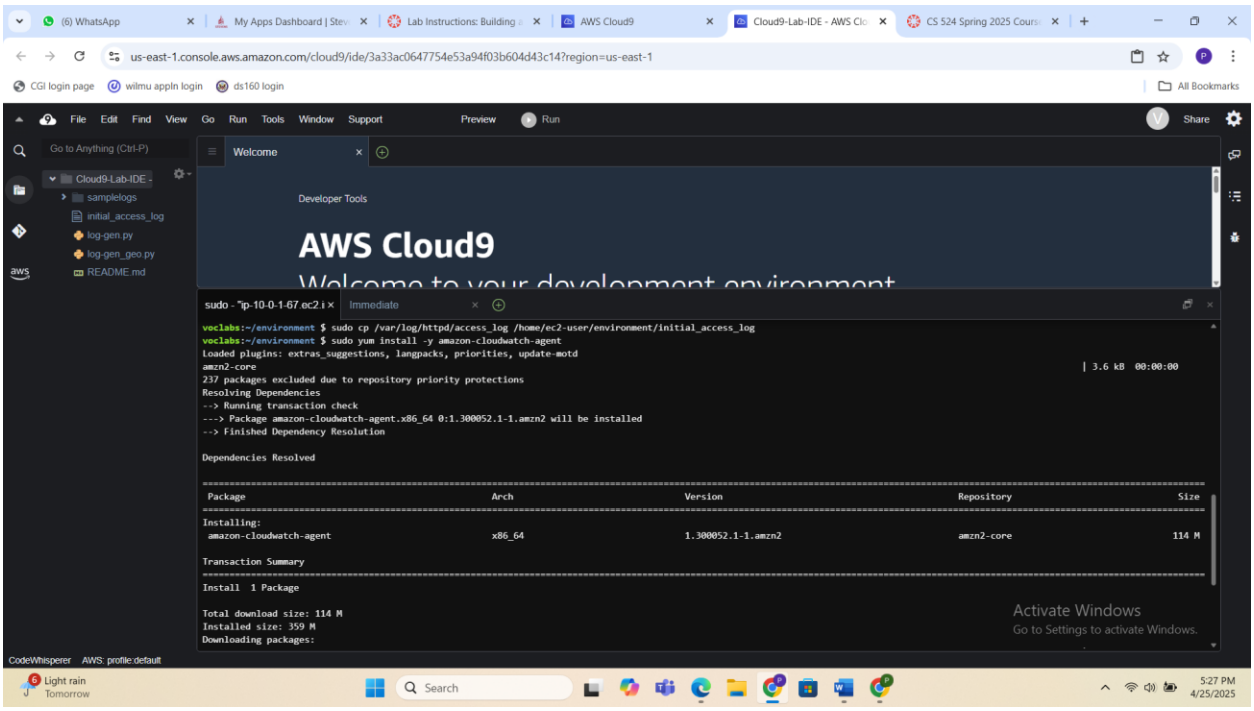
PHASE 3

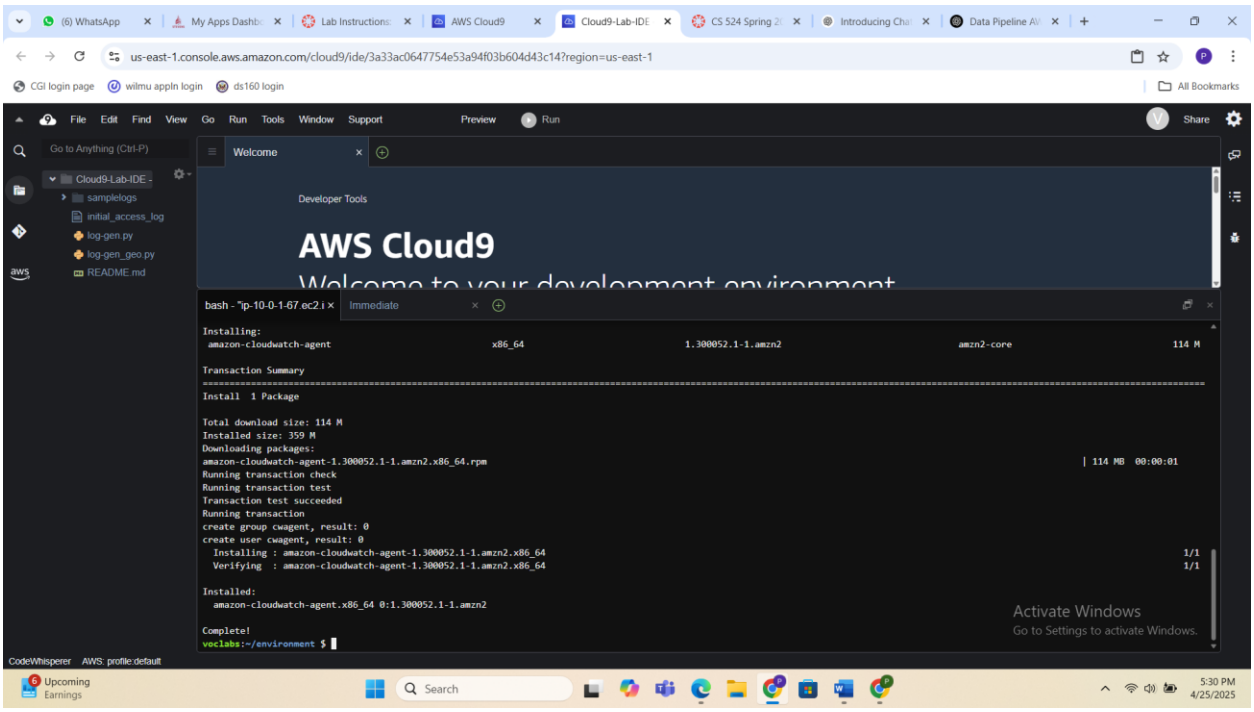
Task1: Install & Configure the CloudWatch Agent

Run this command

sudo yum install -y amazon-cloudwatch-agent

This installs the agent that will send log data from your EC2 instance to Amazon CloudWatch.



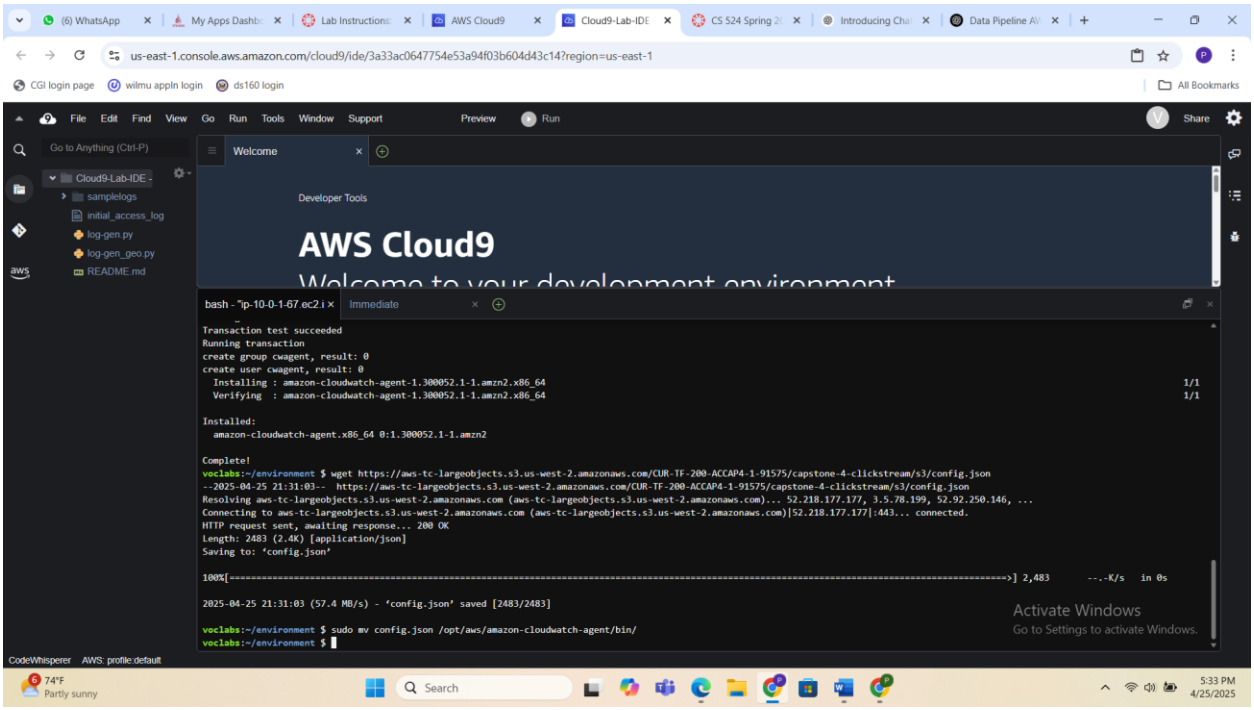


Download the Prebuilt Config File

wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCAP4-1-91575/capstone-4-clickstream/s3/config.json

Run the command

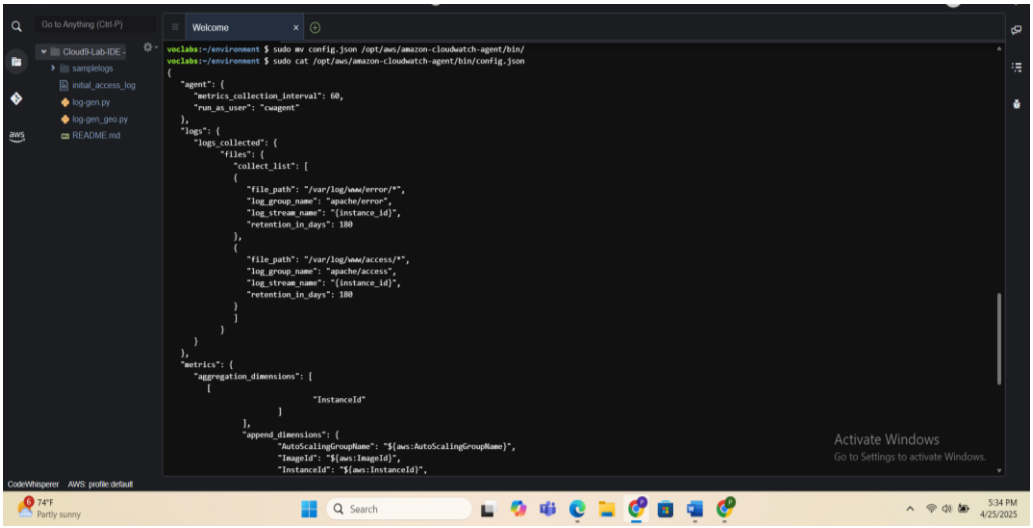
sudo mv config.json /opt/aws/amazon-cloudwatch-agent/bin/

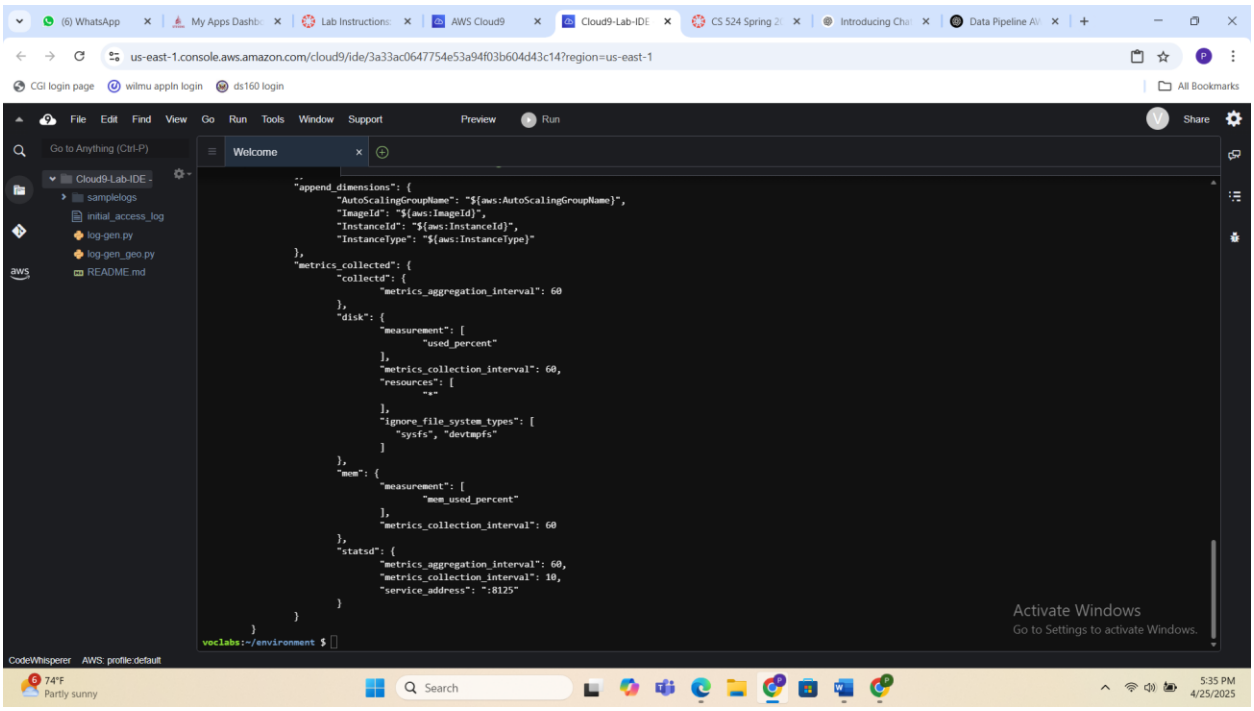


Review the Config File

sudo cat /opt/aws/amazon-cloudwatch-agent/bin/config.json

This file tells CloudWatch which logs to collect, where to send them, how long to keep them, etc.



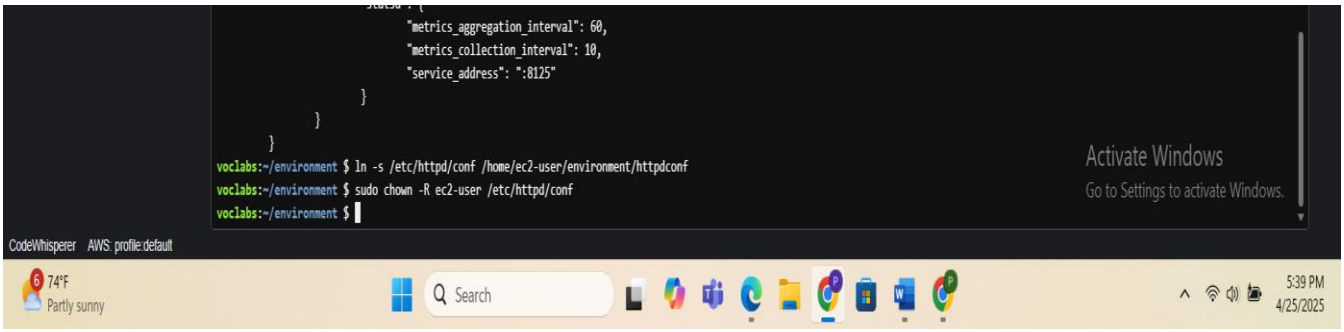


Task 2: Update httpd.conf to Log in JSON Format

ln -s /etc/httpd/conf /home/ec2-user/environment/httpdconf

sudo chown -R ec2-user /etc/httpd/conf

This makes a shortcut (symlink) to the config file and gives you permission to edit it in Cloud9.



In Cloud9 IDE (left sidebar), open: httpdconf → httpd.conf

Find:

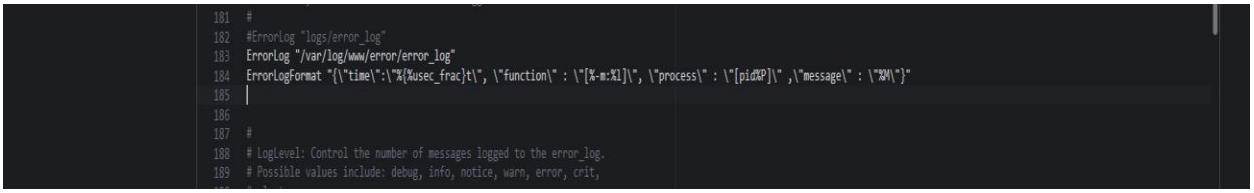
ErrorLog "logs/error_log"

ErrorLog "logs/error_log"(comment it)

Paste these 2 commands below

ErrorLog "/var/log/www/error/error_log"

ErrorLogFormat "{\"time\":\"%{%usec_frac}t\", \"function\" : \"[%-m:%l]\", \"process\" : \"[pid%P]\" ,\"message\" : \"%M\"}"



Find:

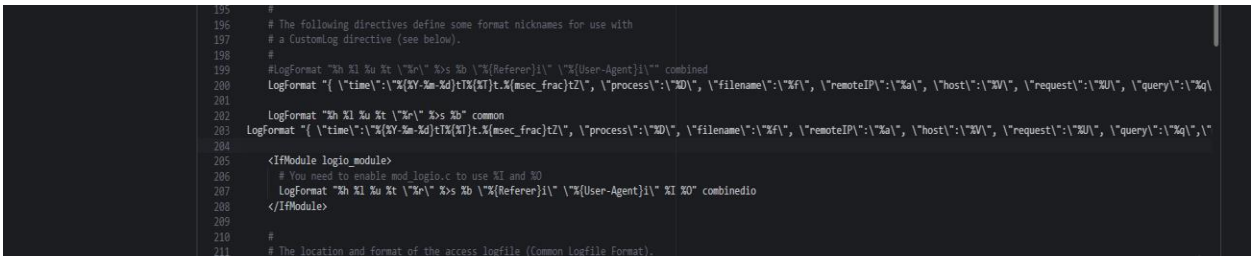
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined

Comment it out:

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined

After that, **paste this new JSON format** below the common one:

LogFormat "{ \"time\":\"%{%Y-%m-%d}tT%{%T}t.%{msec_frac}tZ\", \"process\":\"%D\", \"filename\":\"%f\", \"remoteIP\":\"%a\", \"host\":\"%V\", \"request\":\"%U\", \"query\":\"%q\",\"method\":\"%m\", \"status\":\"%>s\", \"userAgent\":\"%{User-agent}i\",\"referer\":\"%{Referer}i\"}" cloudwatch

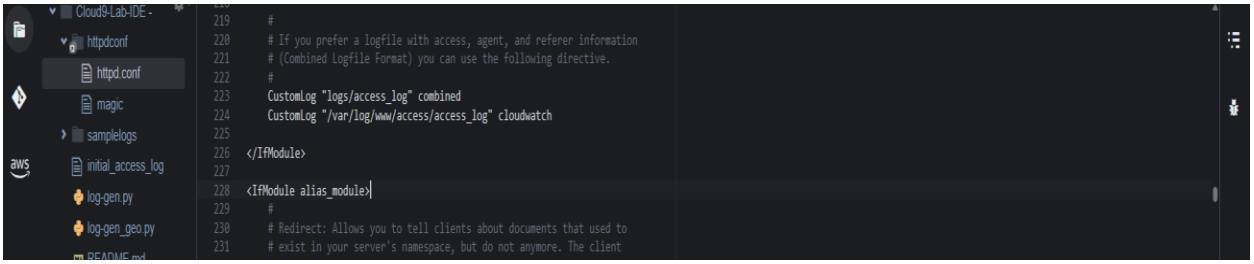


Find:

CustomLog "logs/access_log" combined

Keep that line, but **paste this below** it:

CustomLog "/var/log/www/access/access_log" cloudwatch



Create new folders for Apache to save logs:

Now let's **start the CloudWatch Agent** to begin shipping your logs to CloudWatch.

Run this command:

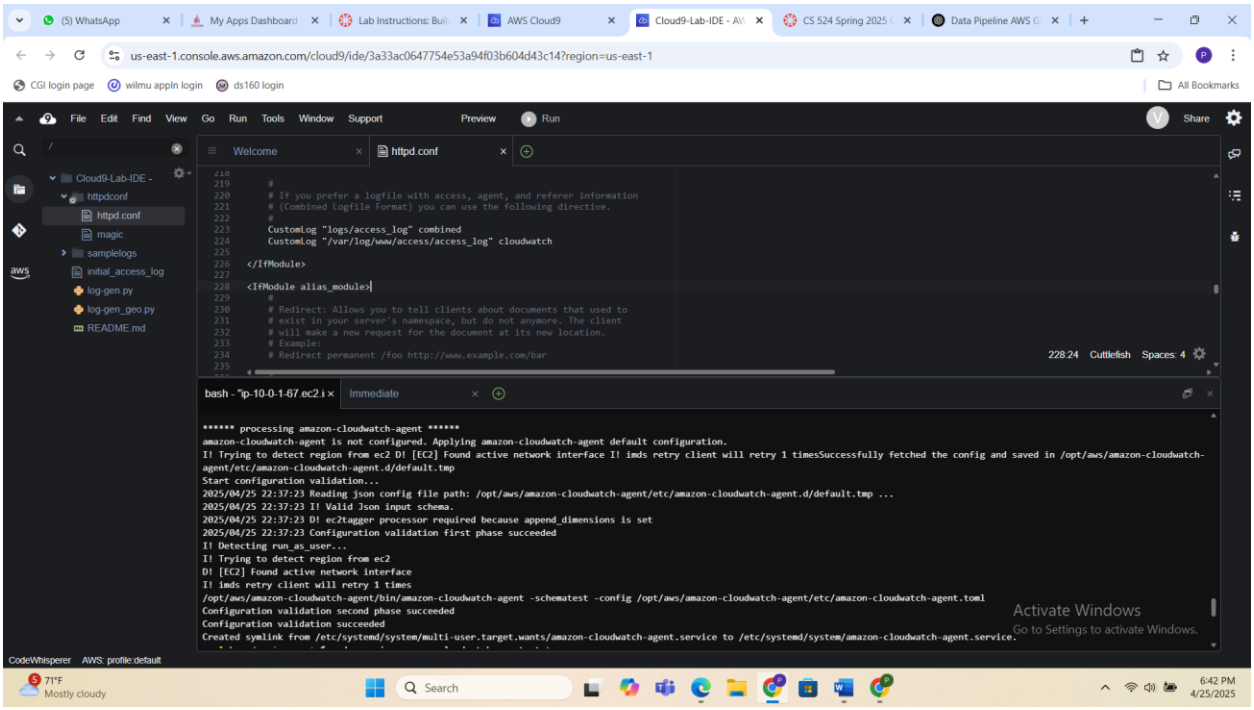
sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl \

-a start \

-m ec2 \

-c file:/opt/aws/amazon-cloudwatch-agent/bin/config.json \

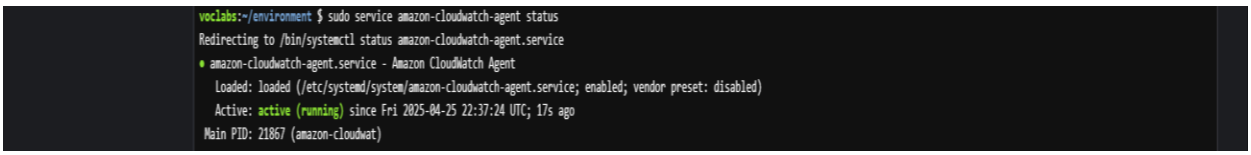
-s



To check if the agent is running properly

Run this:

sudo service amazon-cloudwatch-agent status

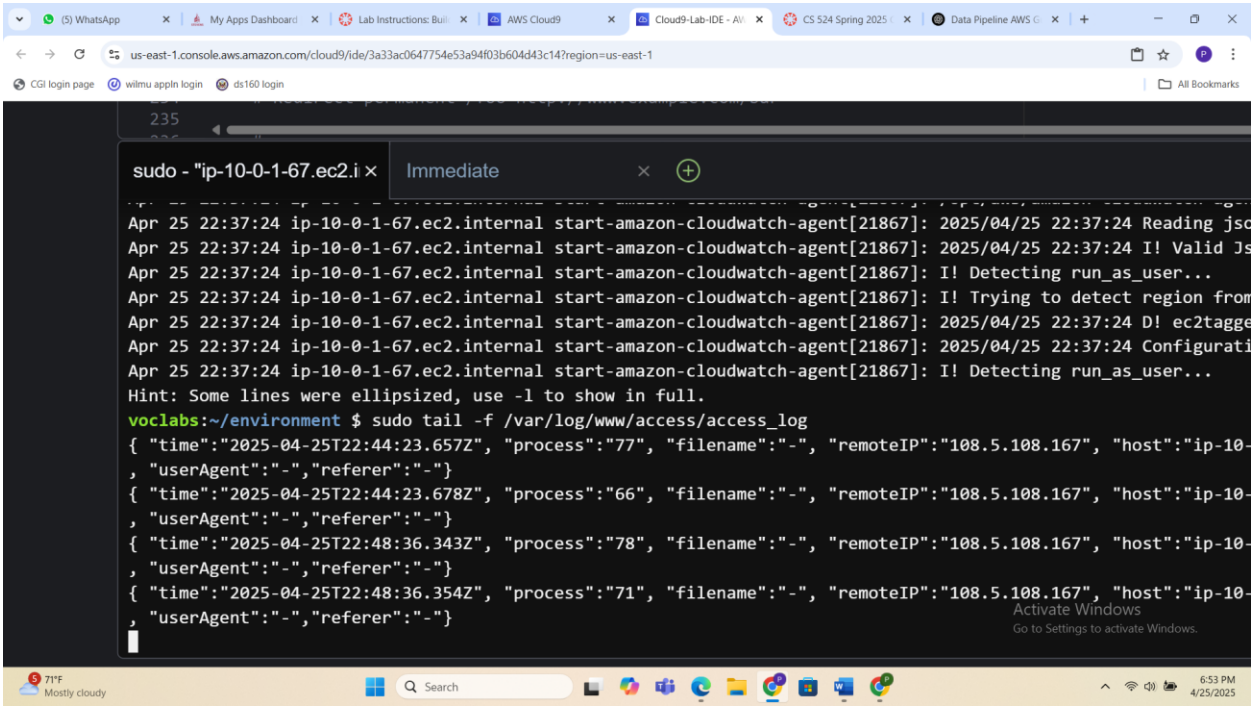


We can see active running

PHASE4: Test CloudWatch Agent & Log Flow

Task 1: Confirm Web Logs Are Being Written

sudo tail -f /var/log/www/access/access_log

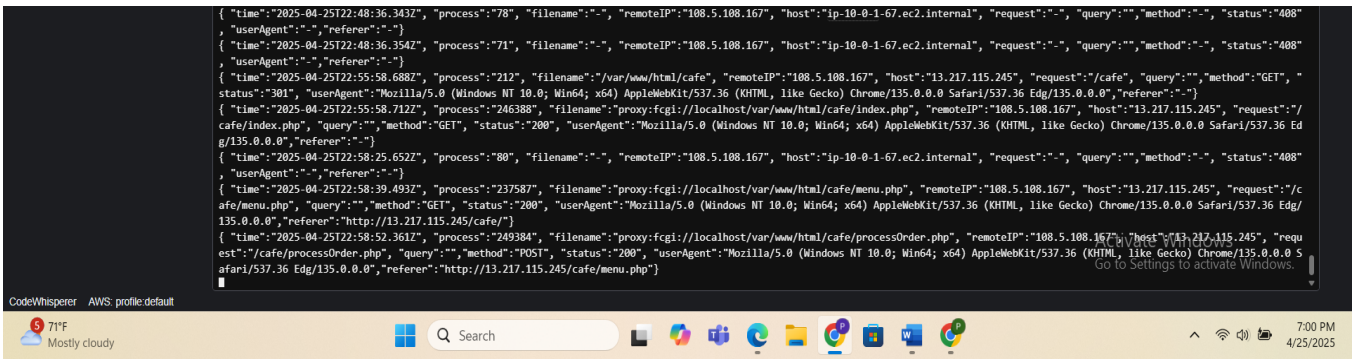


This will show real-time log entries as people visit the website

Visiting pages like:

- /menu.php
- /processOrder.php

Makes changes in terminal —we can see new log lines appear, and they should look like JSON

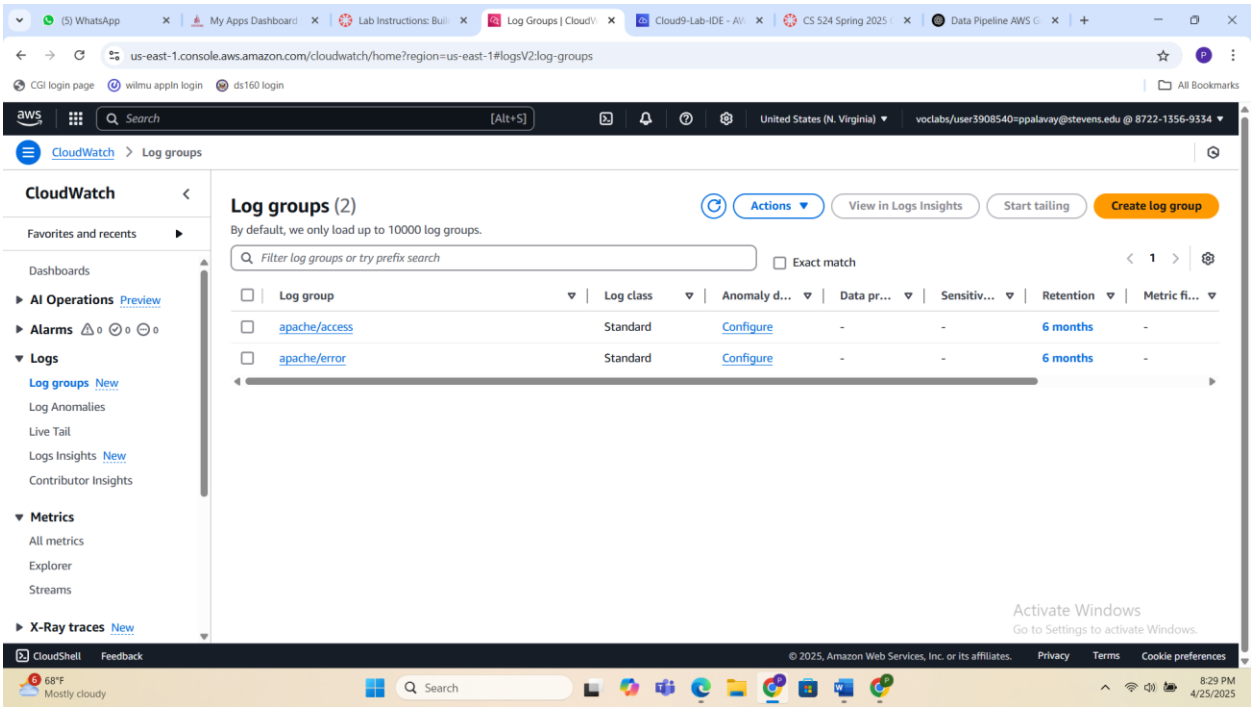


Task2: Verify in AWS Console (CloudWatch Logs)

Now go to the AWS Console:

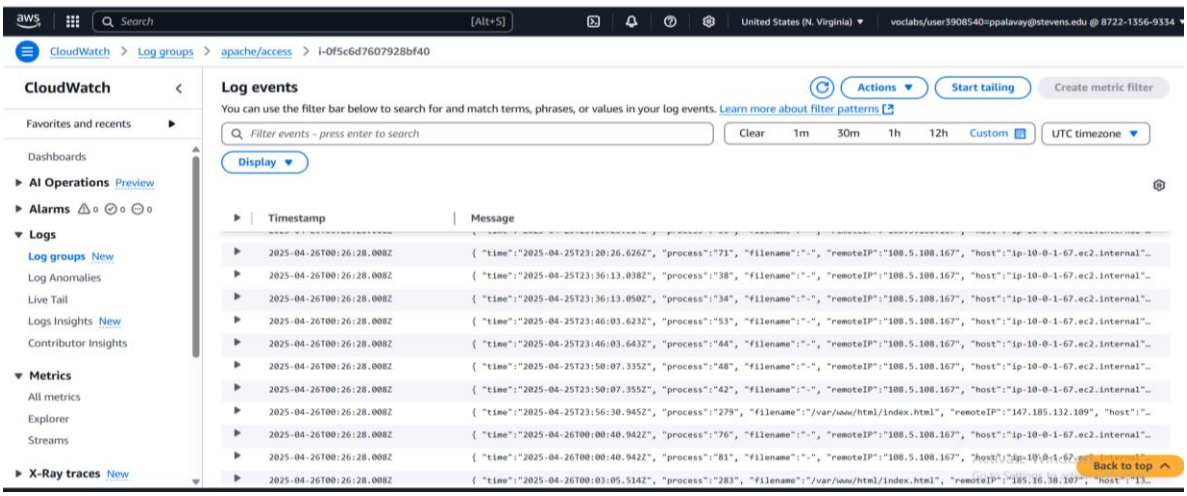
Navigate to CloudWatch → Logs → Log Groups

We should see apache/access log groups



Click into apache/access log group open the latest Log Stream

We should see entries in JSON format showing page visits!



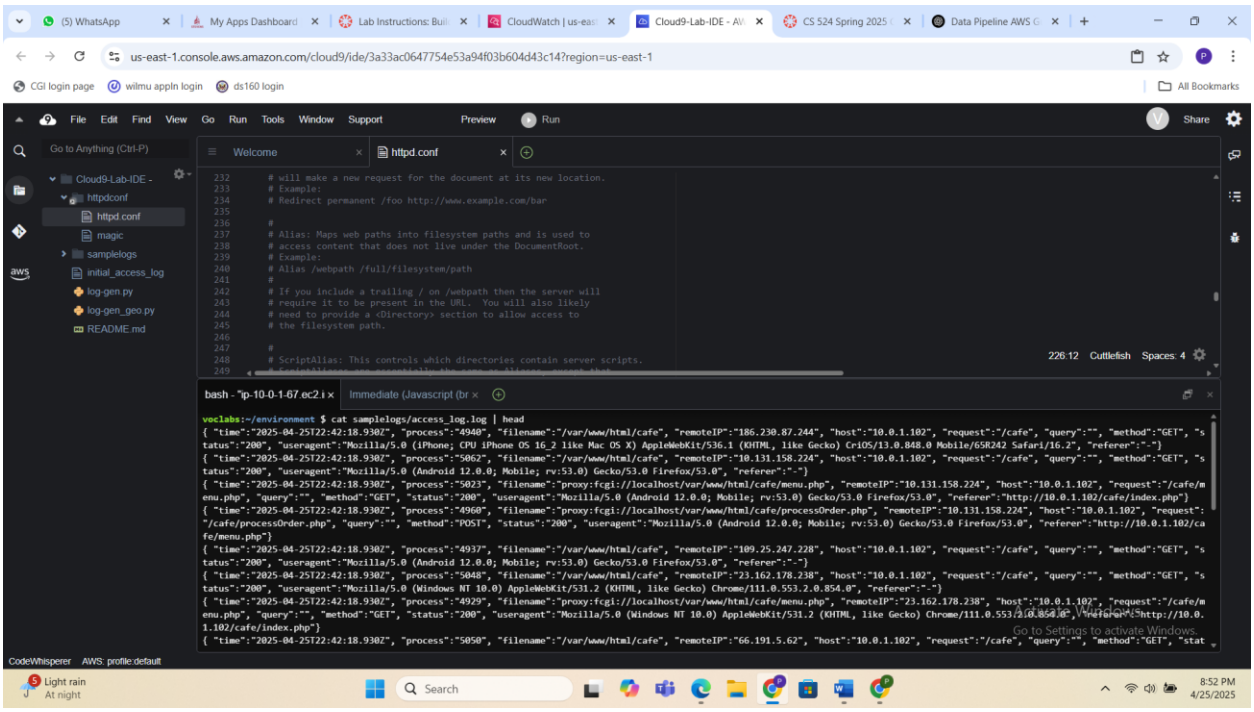
Phase 5: Use Simulated Logs

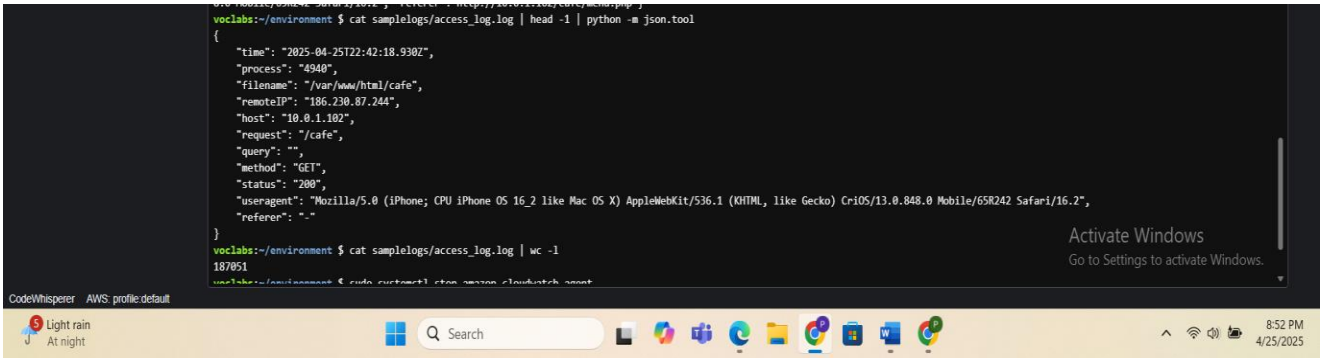
Run the commands

cat samplelogs/access_log.log | head

cat samplelogs/access_log.log | head -1 | python -m json.tool

cat samplelogs/access_log.log | wc -l

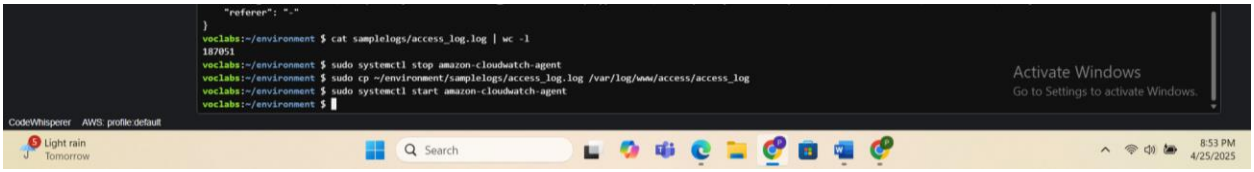




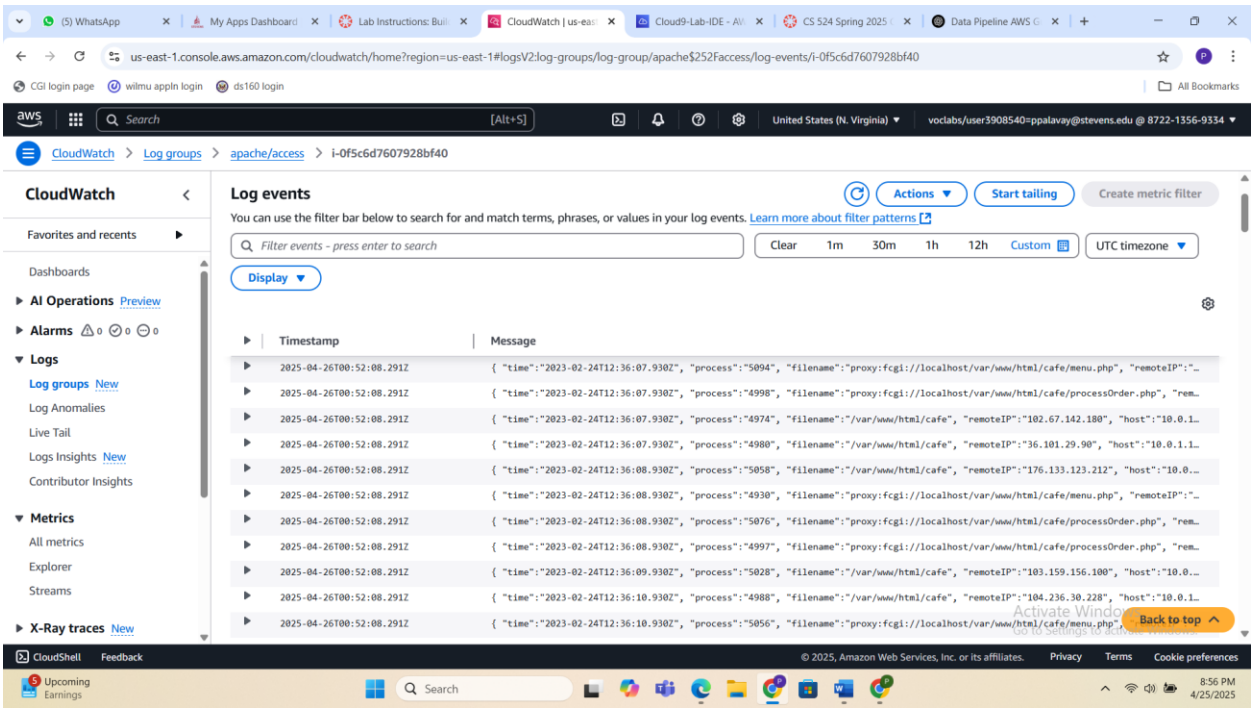
sudo systemctl stop amazon-cloudwatch-agent

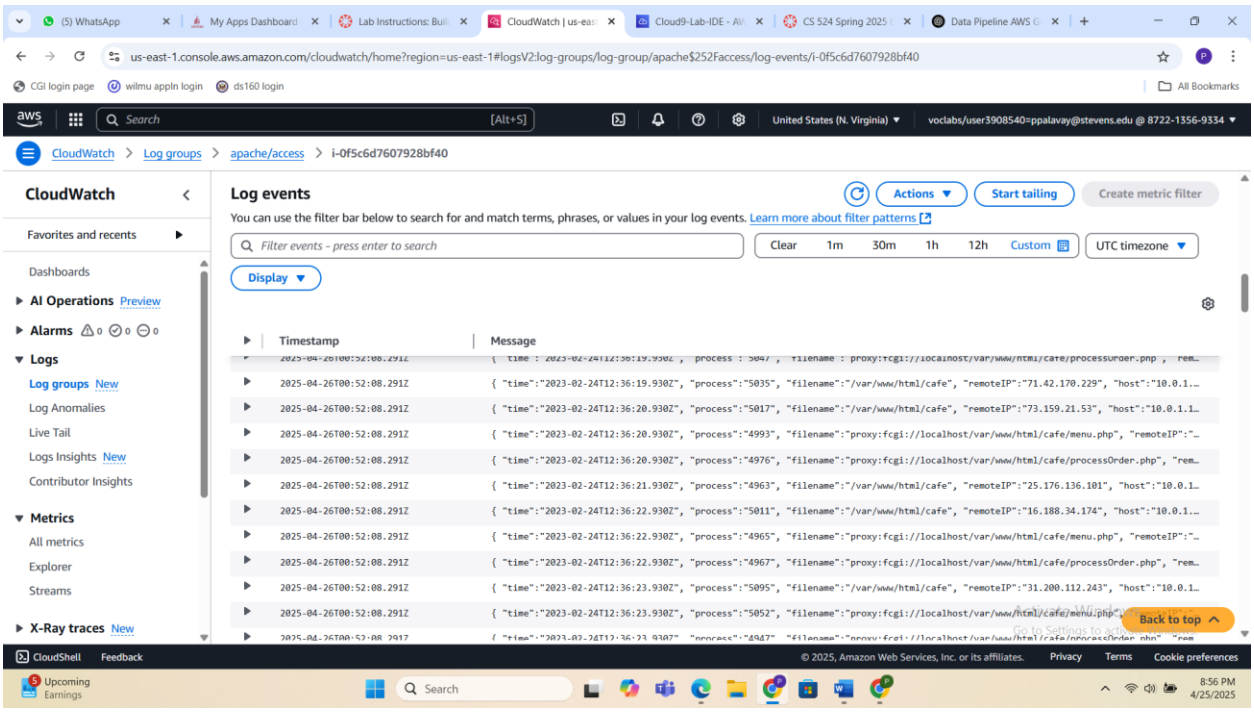
sudo cp ~/environment/samplelogs/access_log.log /var/log/www/access/access_log

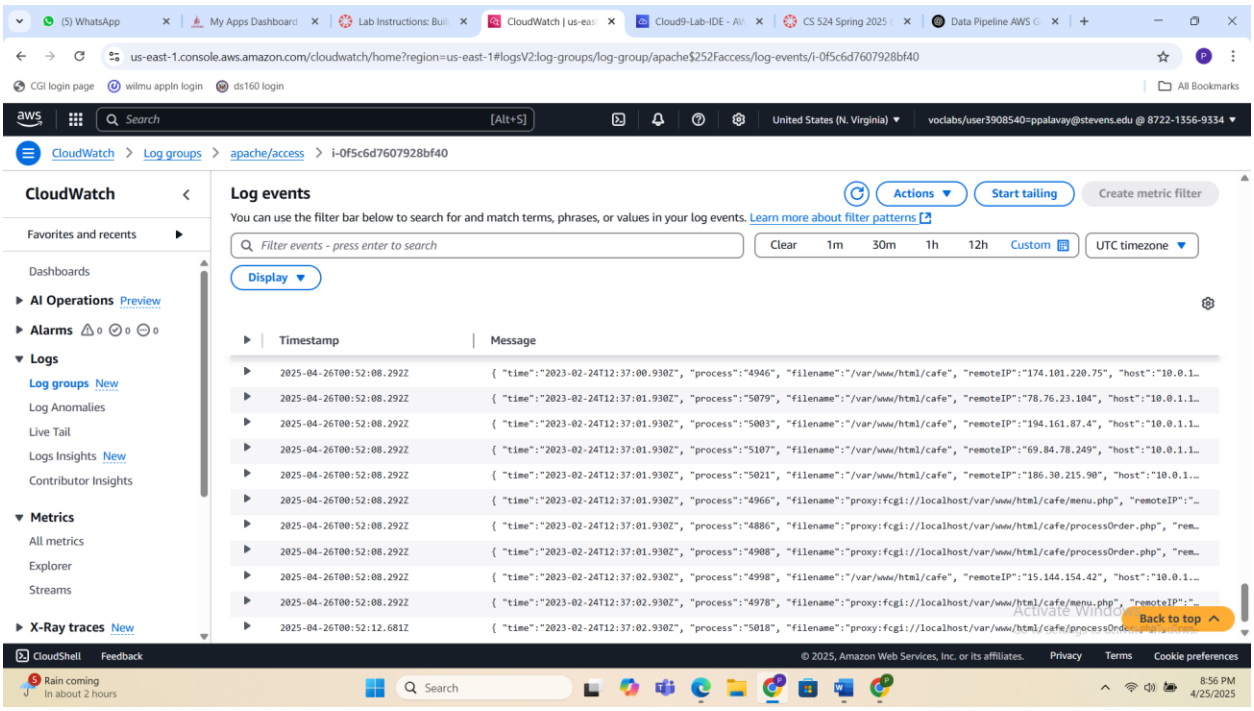
sudo systemctl start amazon-cloudwatch-agent



We can see more log events







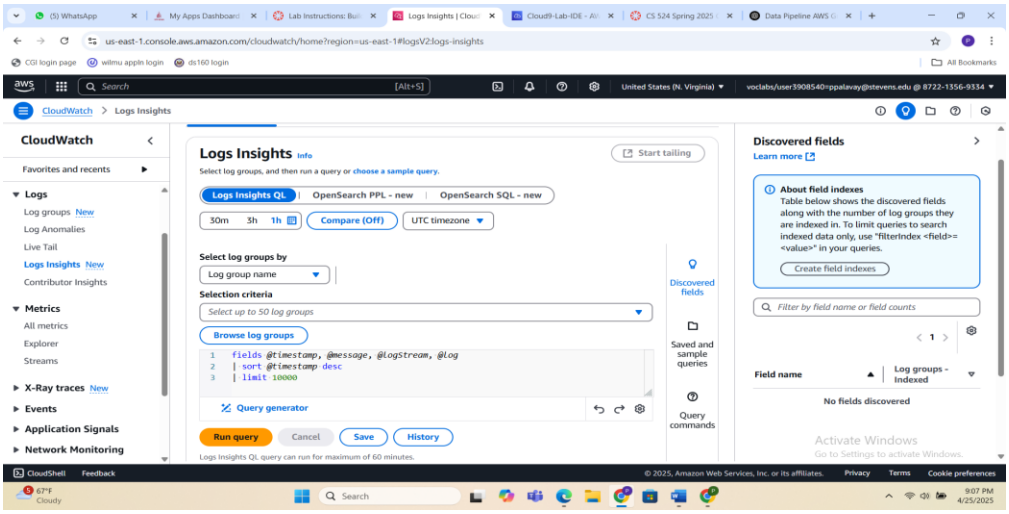
PHASE6: Analyze Data Using Logs Insights

Go to the CloudWatch console

In the sidebar, choose Logs Insights

At the top, choose the log group

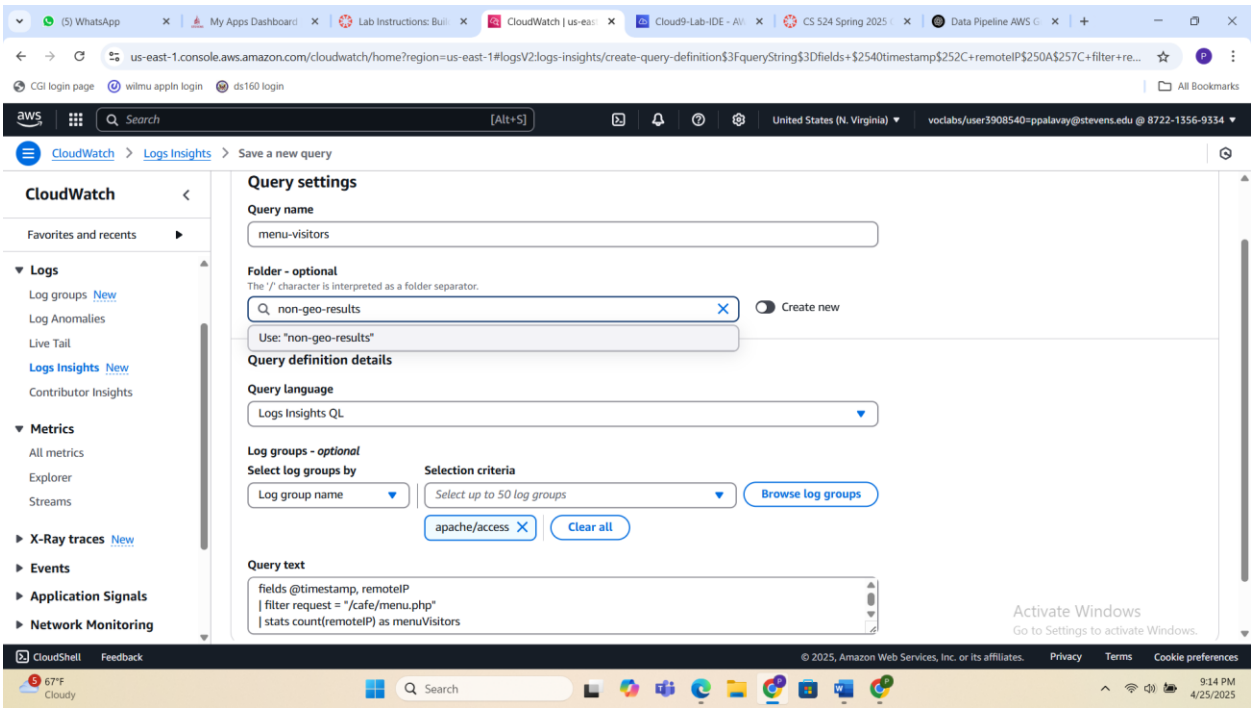
Selection criteria: apache/access

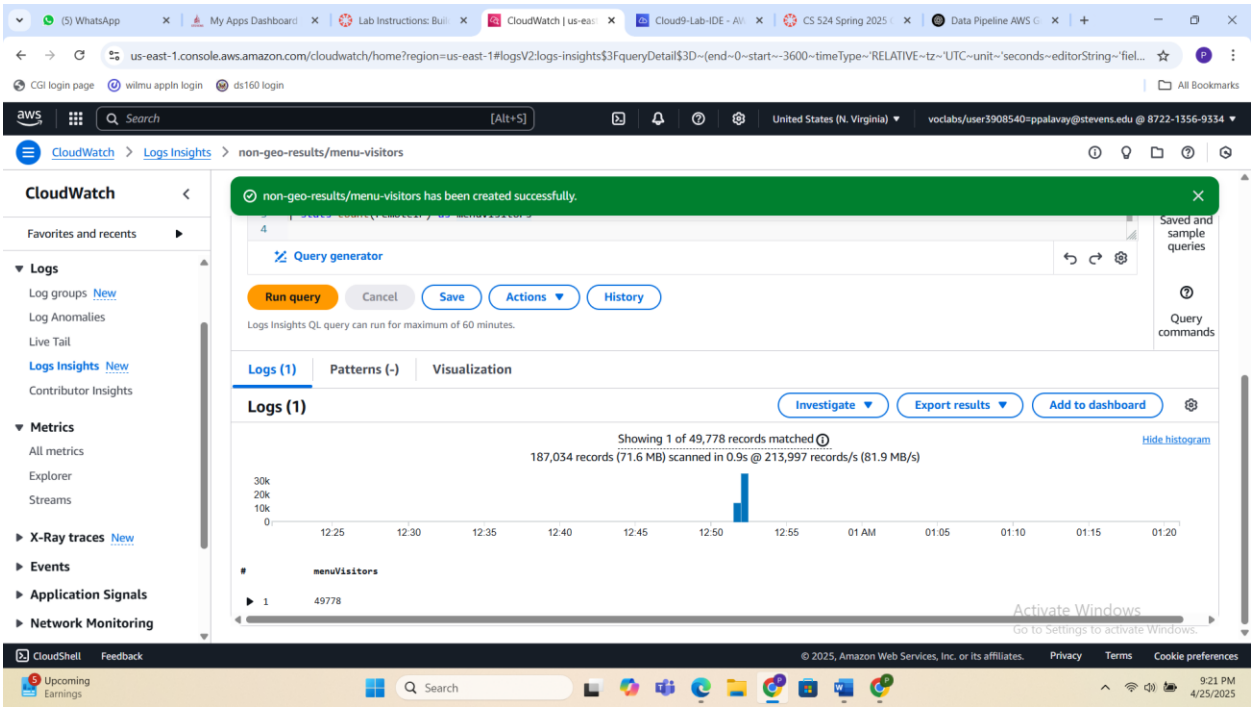


Click **Save query**

Name: menu-visitors

Folder: non-geo-results

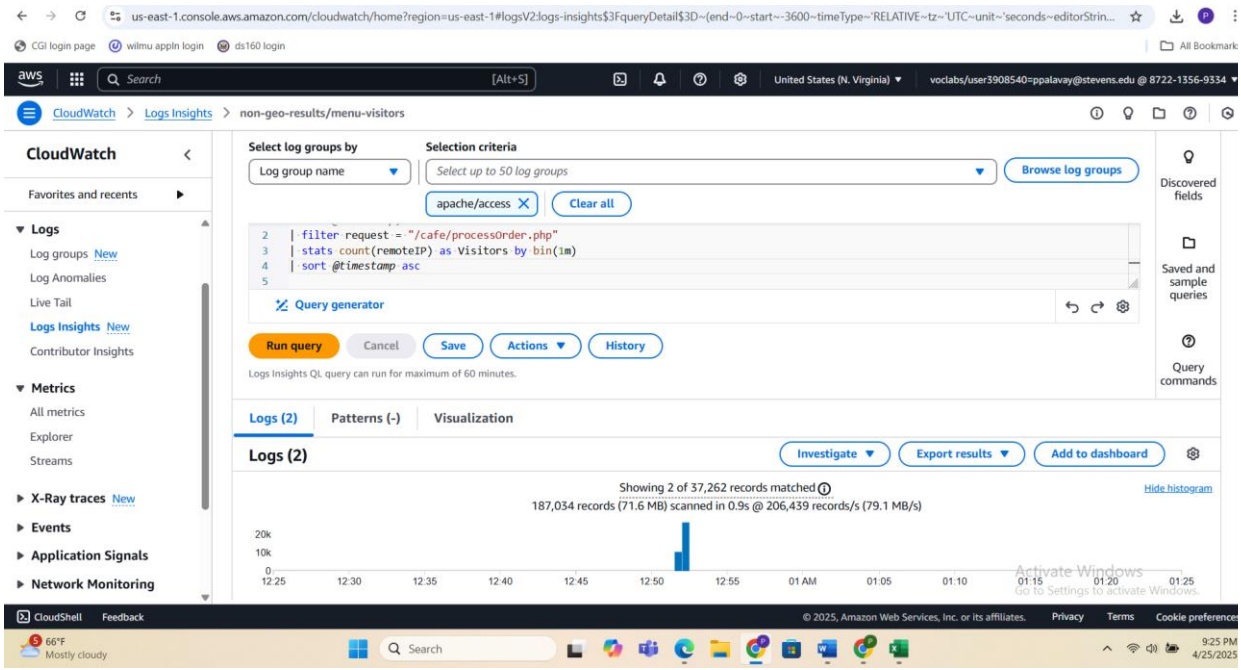




Menu visitors:49778

Run another query:

```
fields @timestamp, remoteIP
| filter request = "/cafe/processOrder.php"
| stats count(remoteIP) as Visitors by bin(1m)
| sort @timestamp asc
```



Purchase visitors:37262

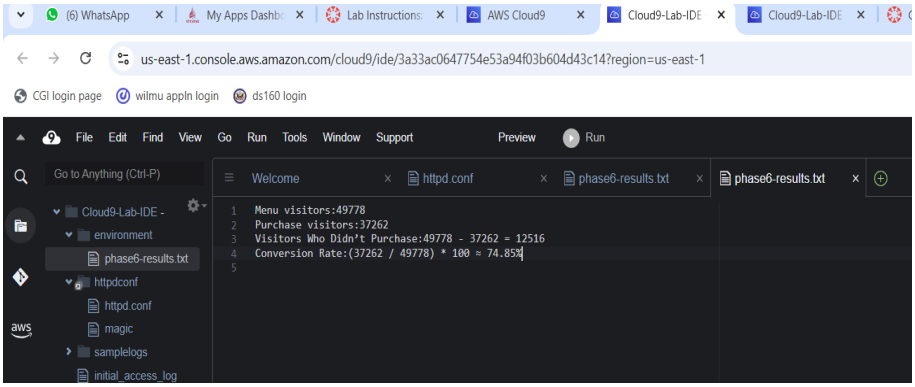
1. Visitors Who Didn't Purchase:

49778 - 37262 = 12516

2. Conversion Rate:

(37262 / 49778) * 100 ≈ 74.85%

Create a result file in environment folder and note down all the values.



PHASE7: **Adjust the Pipeline to Deliver New Insights**

Step 1: Stop the CloudWatch Agent

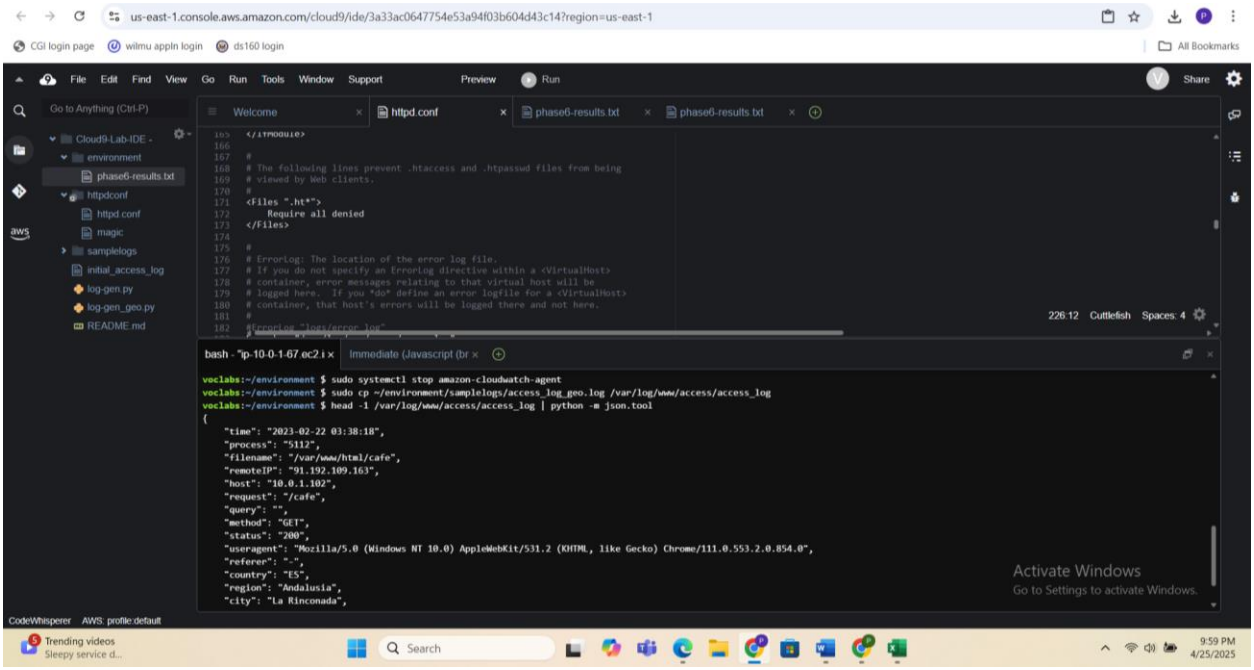
sudo systemctl stop amazon-cloudwatch-agent

Step 2: Replace the Access Log with Geolocation Data

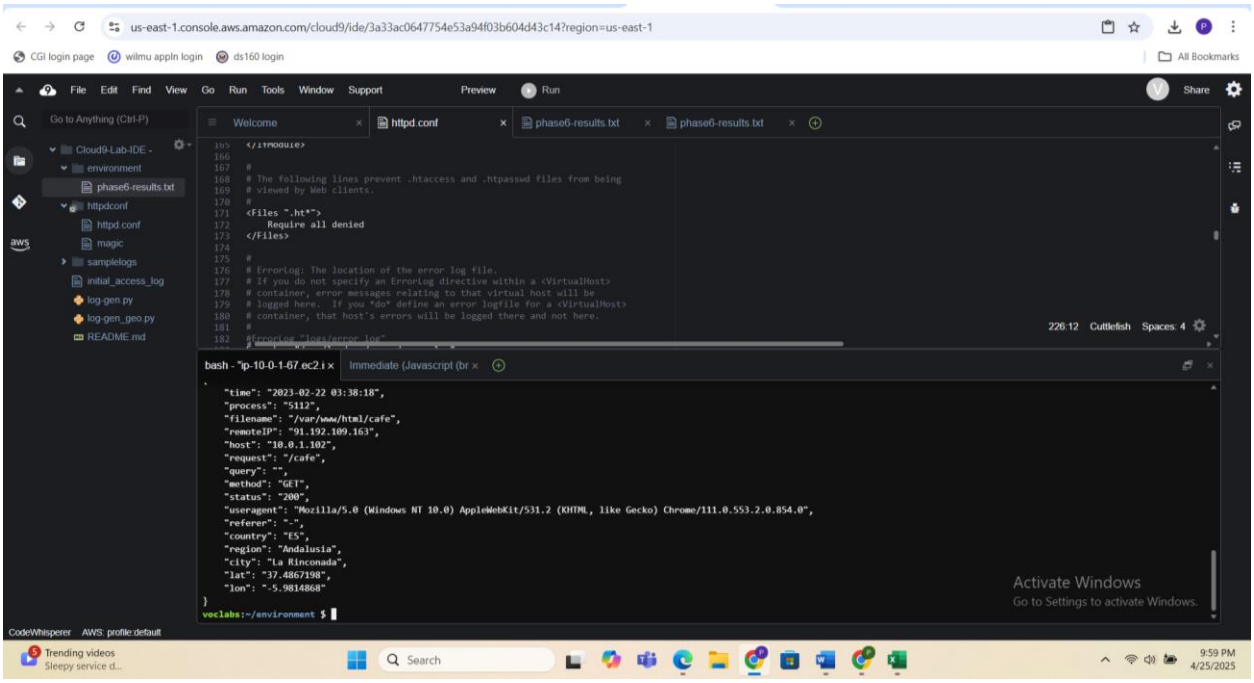
sudo cp ~/environment/samplelogs/access_log_geo.log /var/log/www/access/access_log

Step3: Confirm Log Has Geolocation Fields

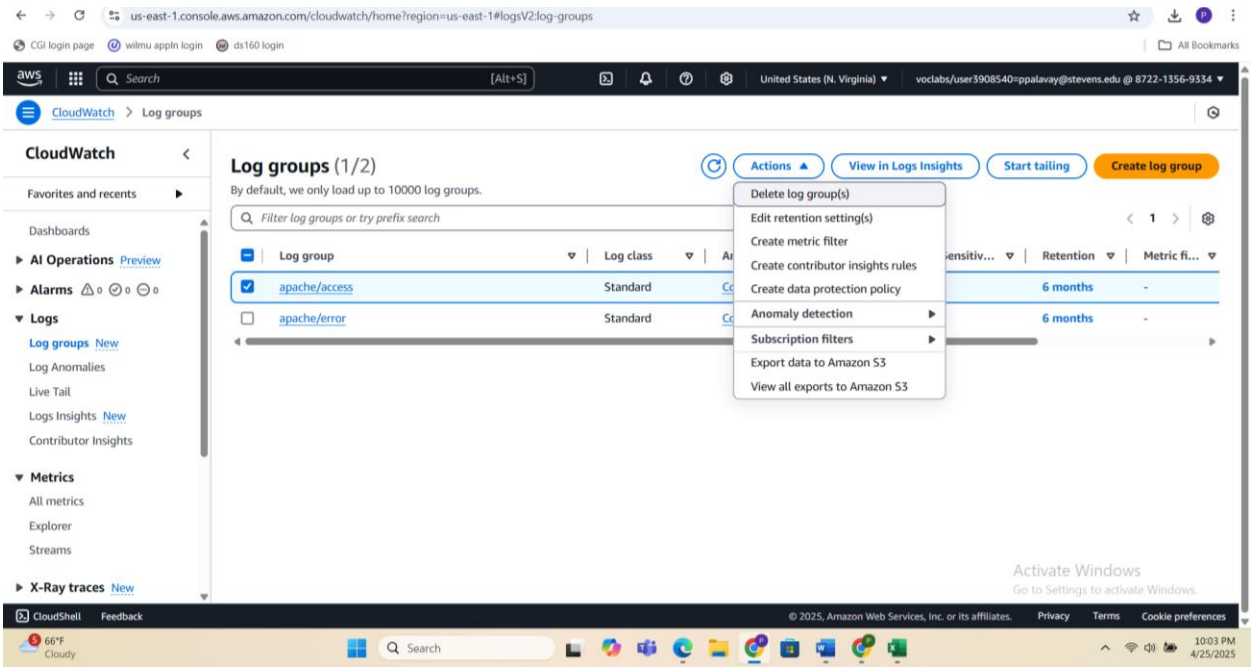
head -1 /var/log/www/access/access_log | python -m json.tool



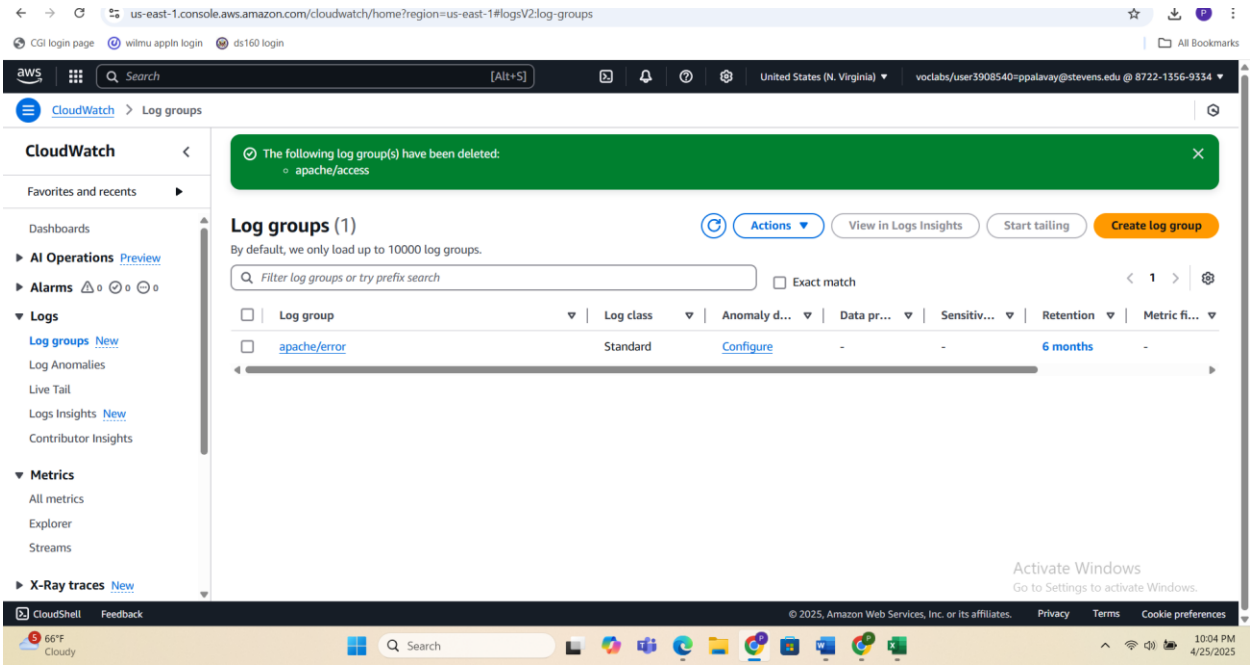
If we see city, region, lat, lon, we're good to go



Delete the Old Log Group



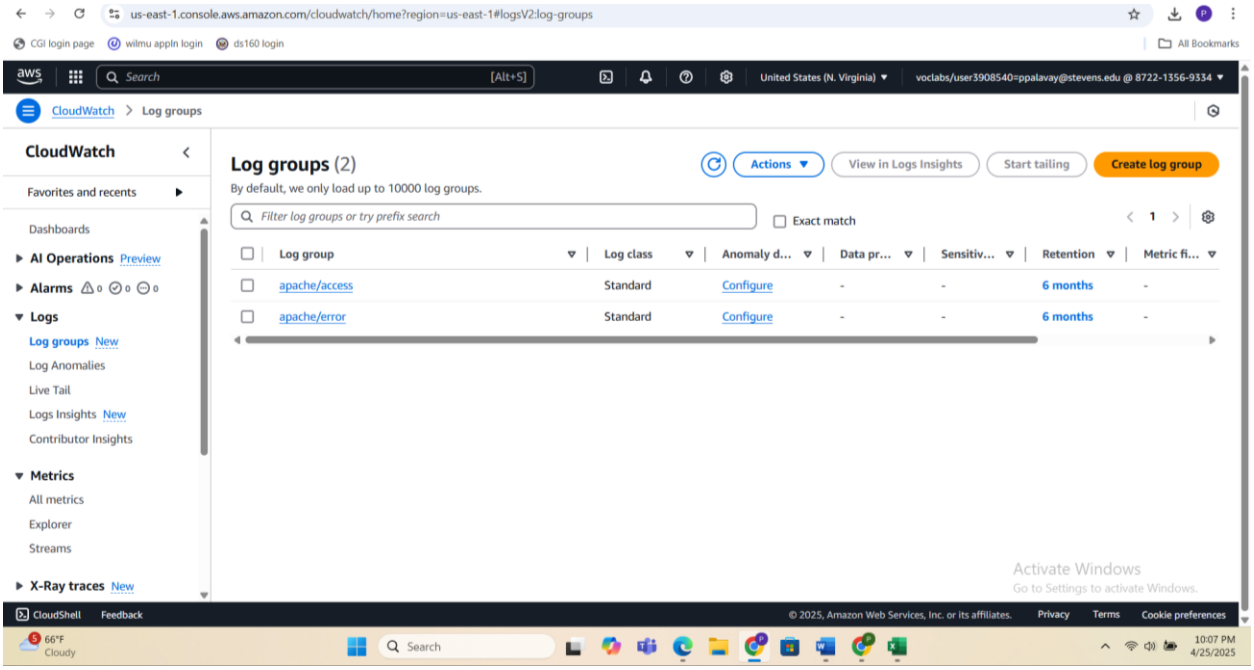
Log Group(apache/access) deleted successfully



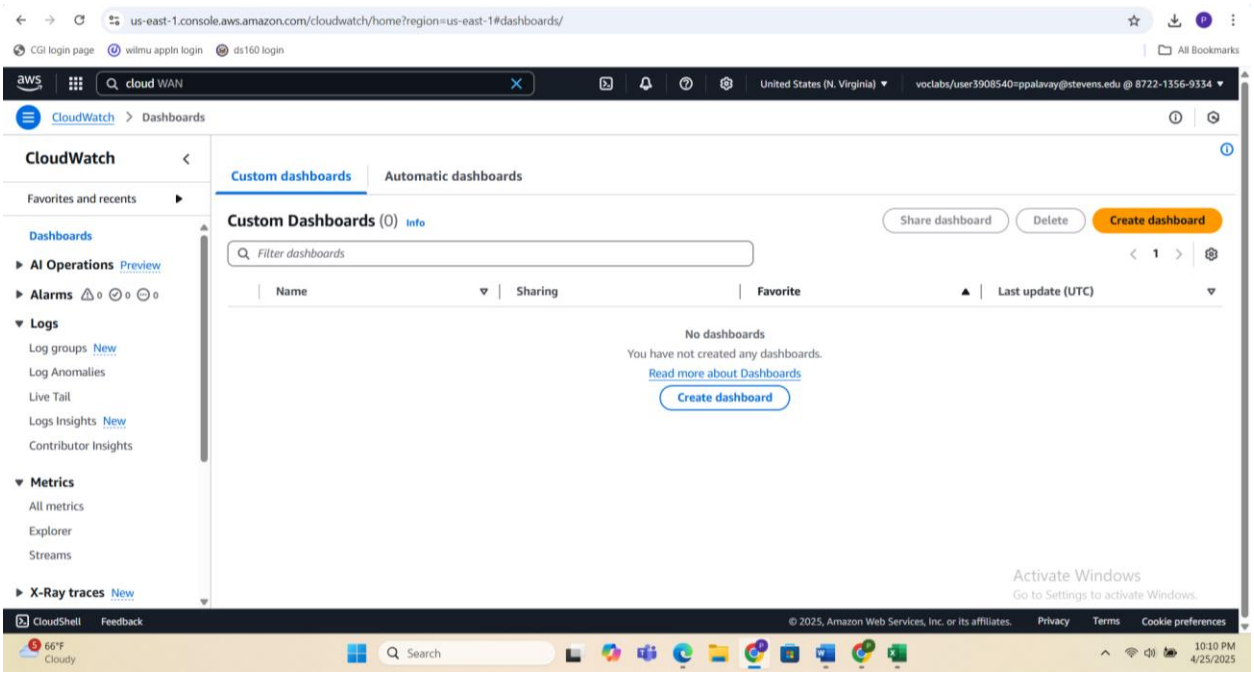
Restart the CloudWatch Agent

sudo systemctl start amazon-cloudwatch-agent

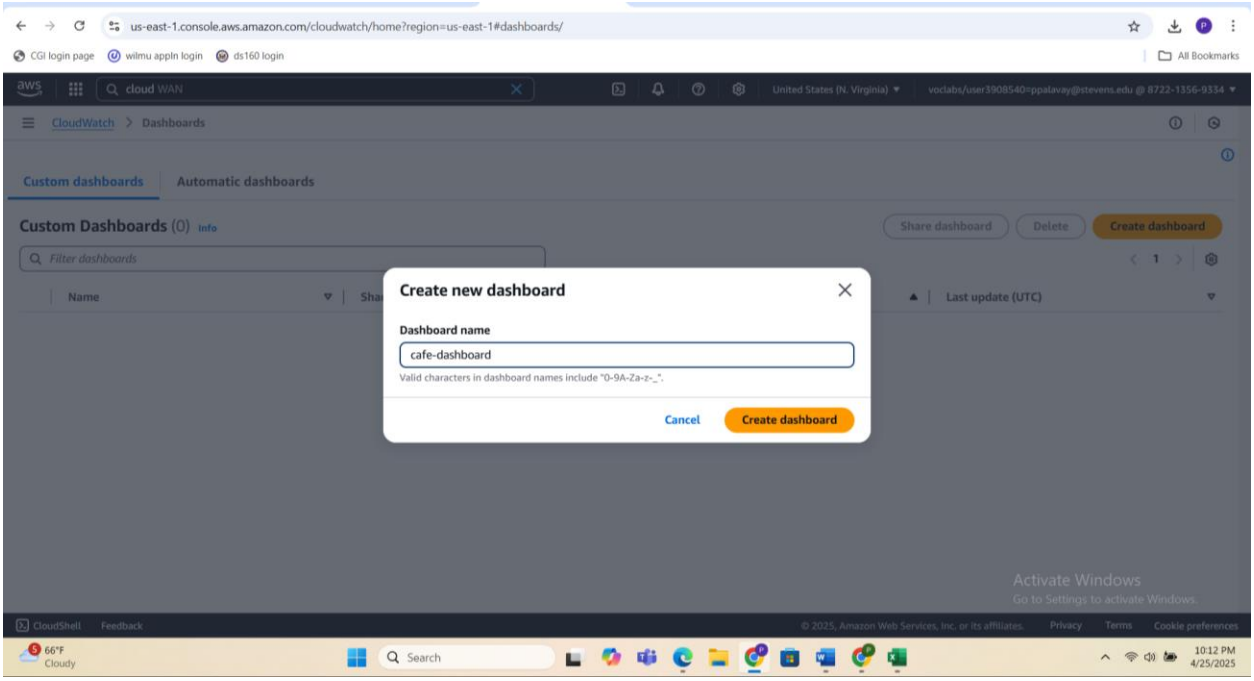
we can see apache/access log group again.



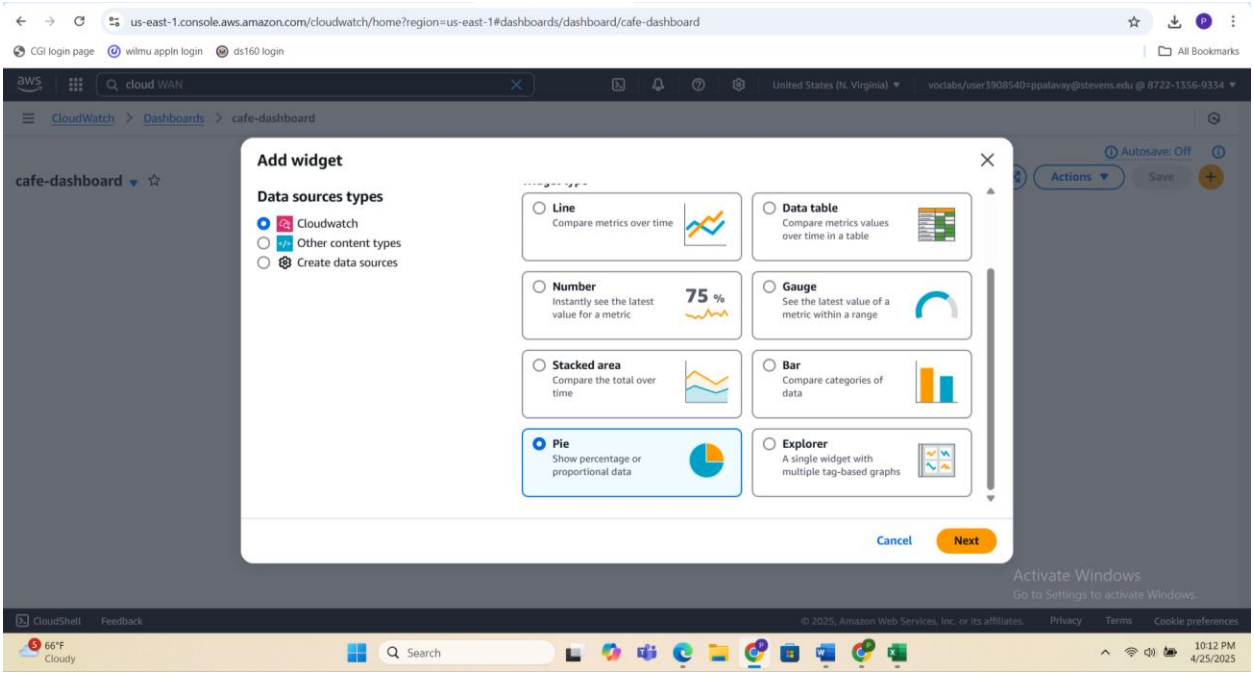
TASK2: Create a Dashboard



Name the dashboard as café-dashboard



We should add widgets



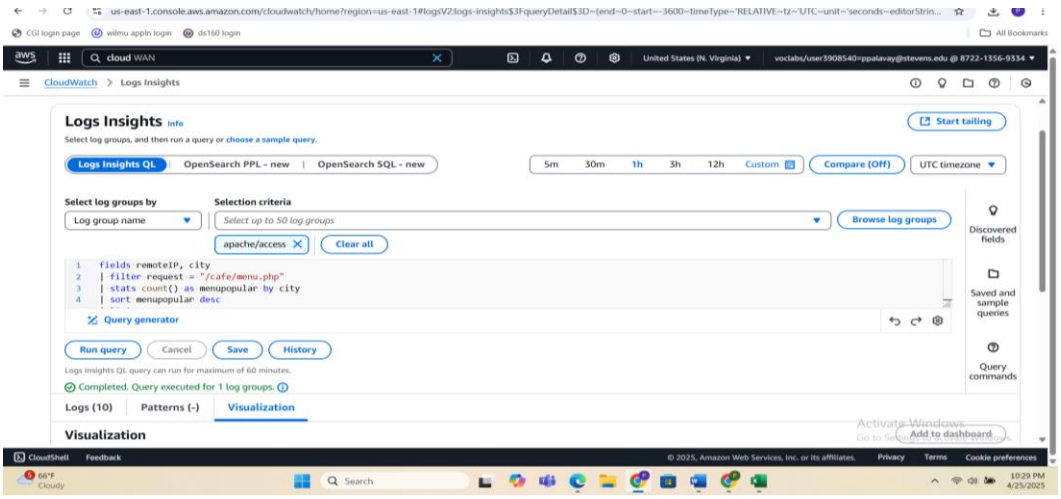
Click **Add widget**

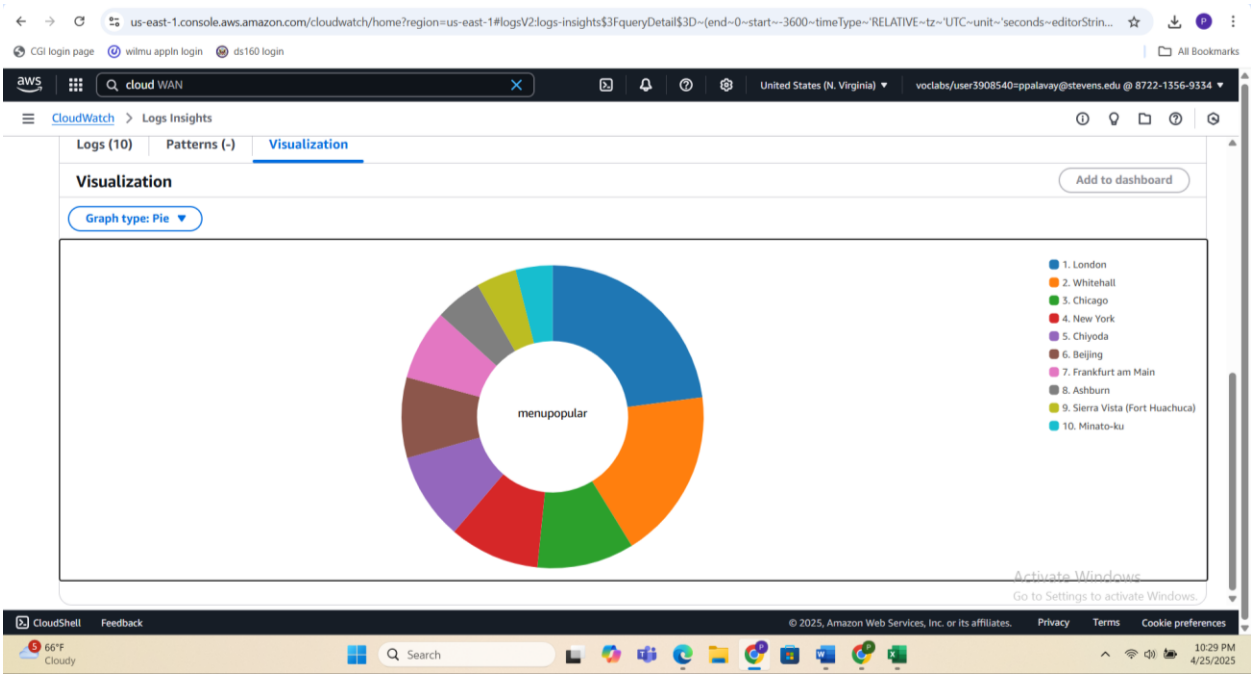
Data type:logs

Select pie chart

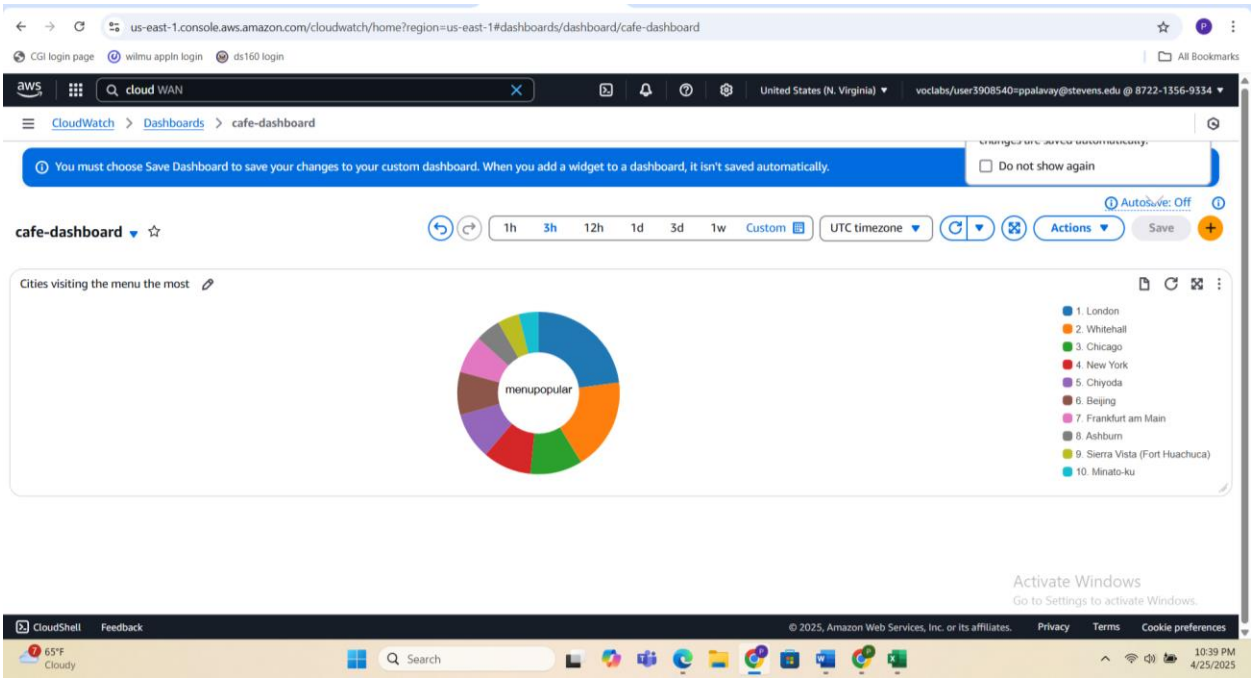
Choose apache/access as your log group

Run the query

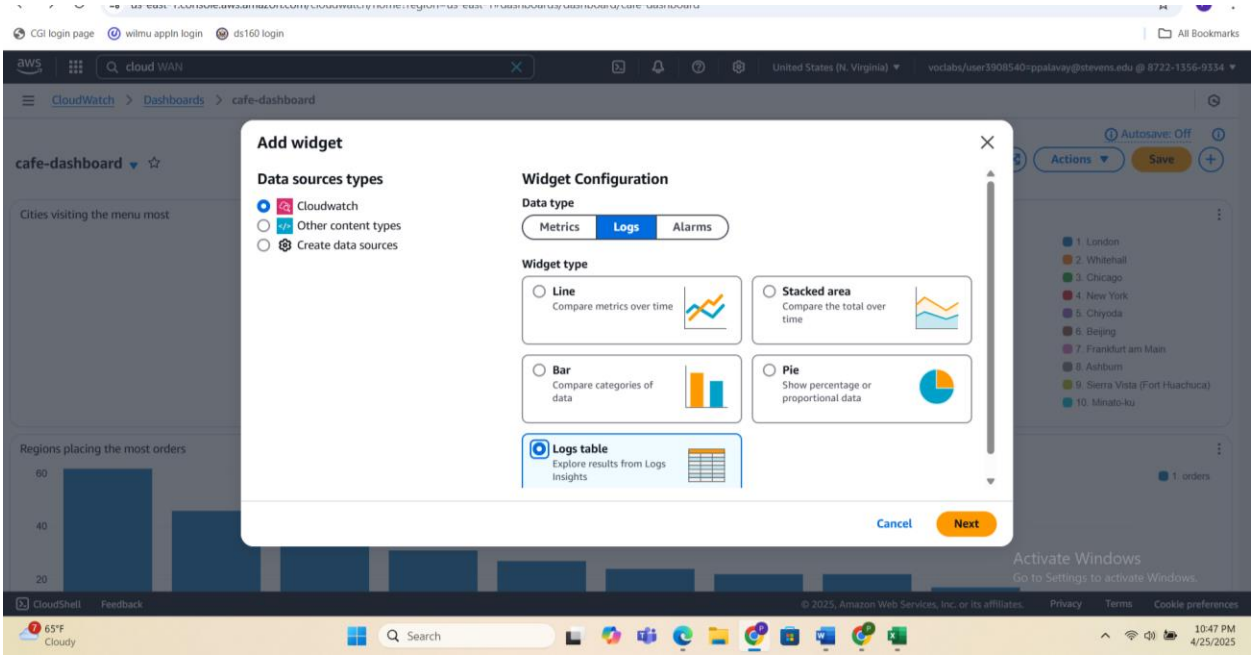




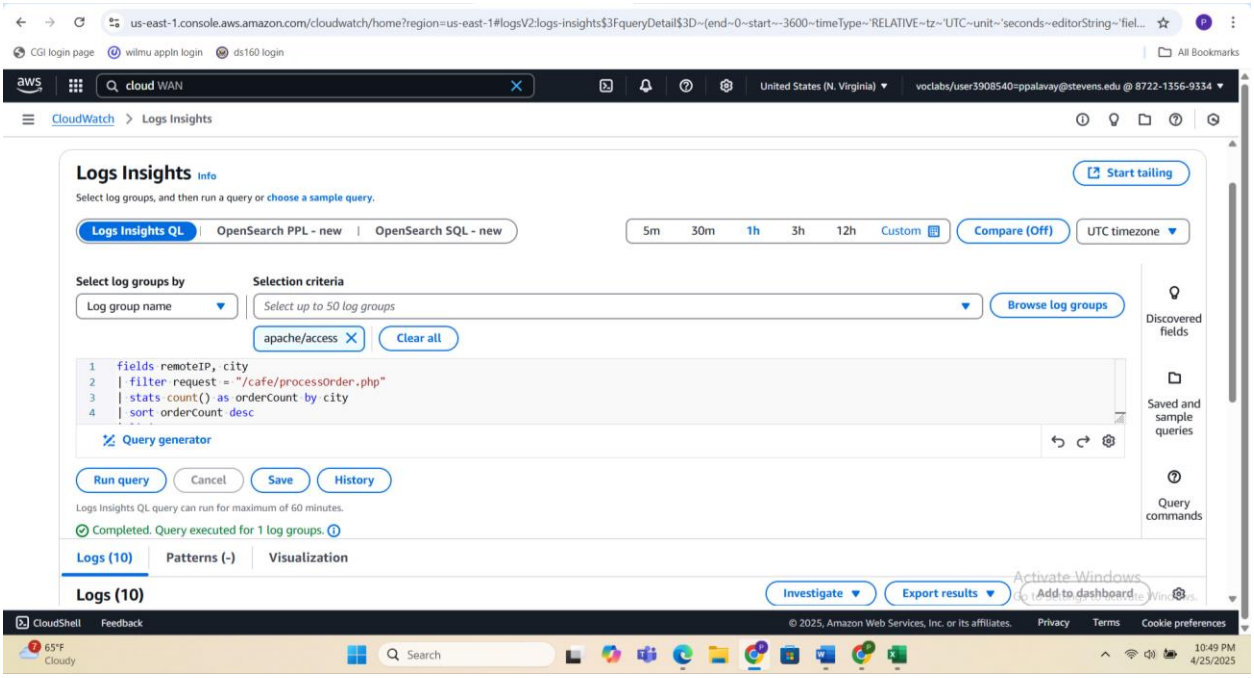
Rename the widget as cities visiting the menu most



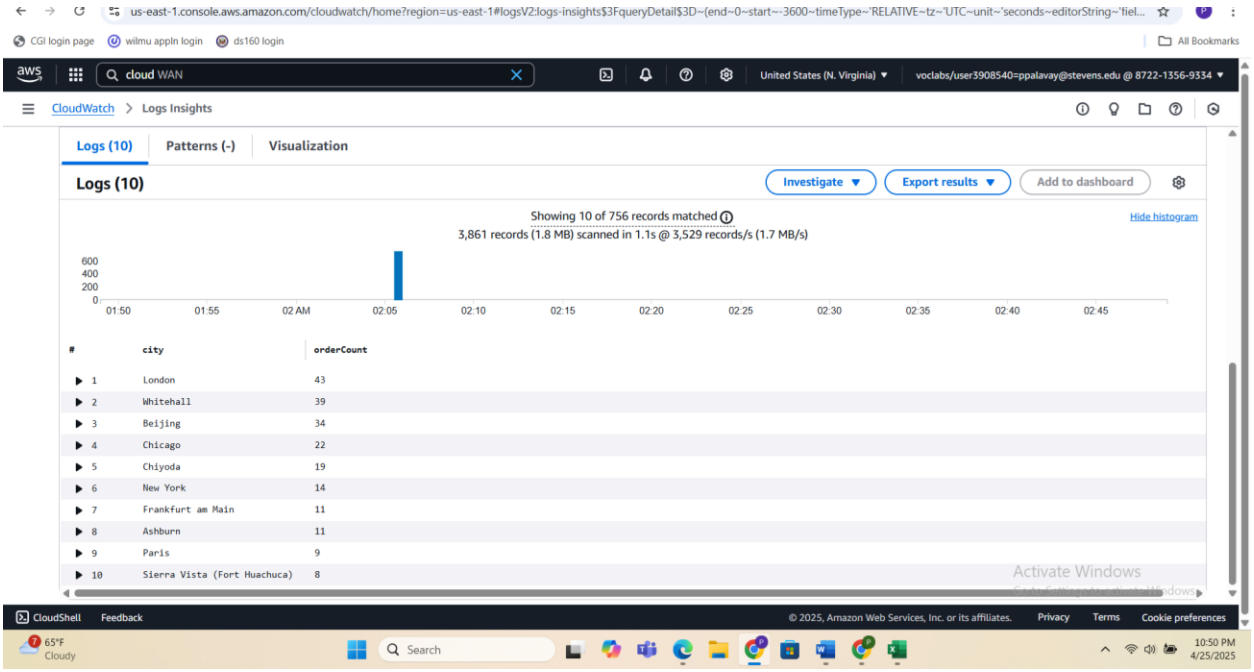
Create second widget same as first but widget type is **logs table**



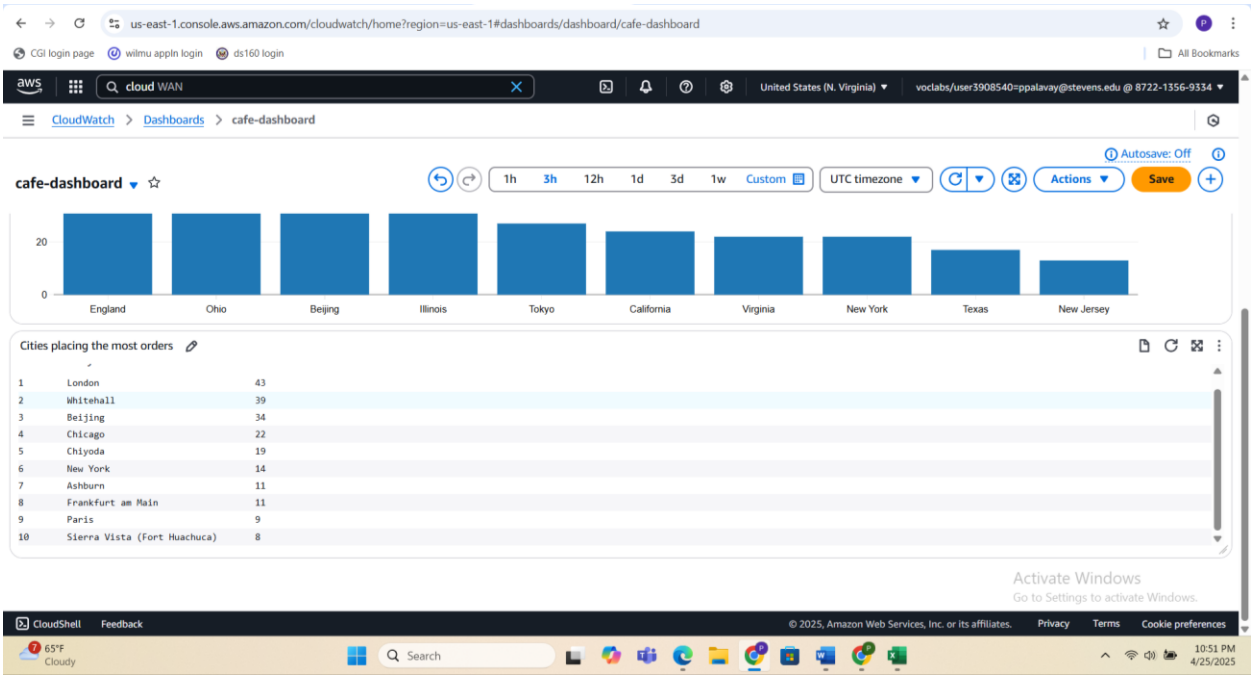
Run the query



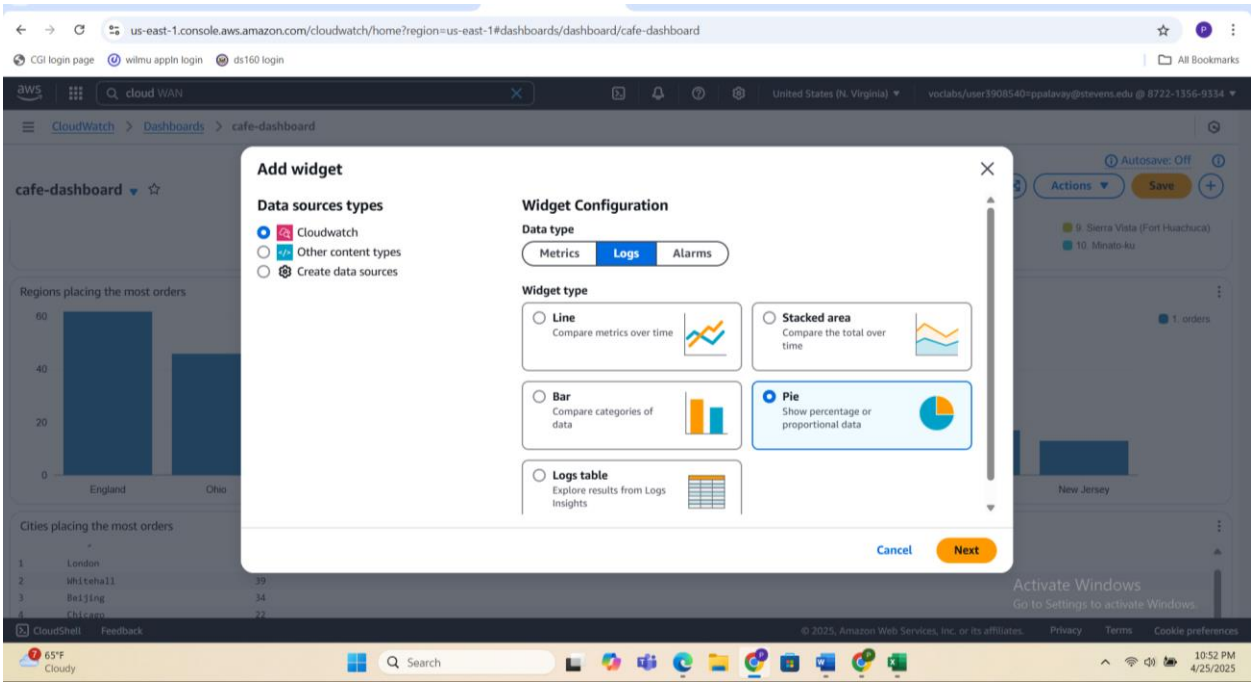
We can see the logs table output

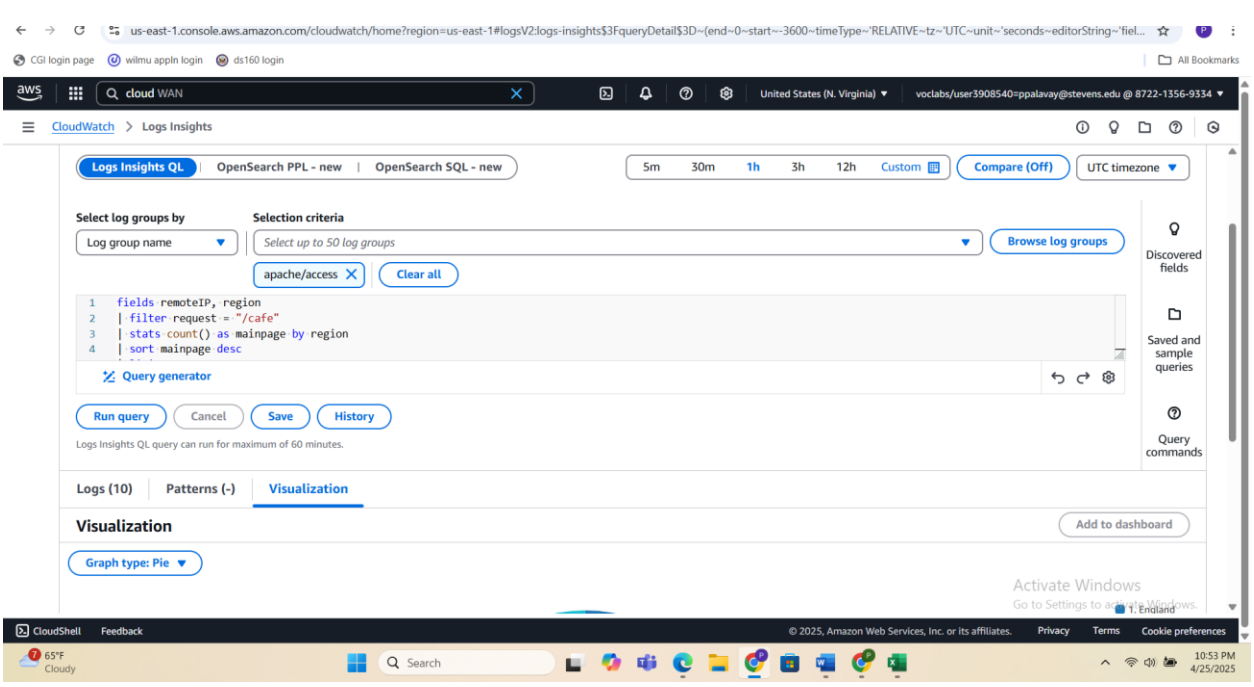


Rename this widget as cities placing the most orders

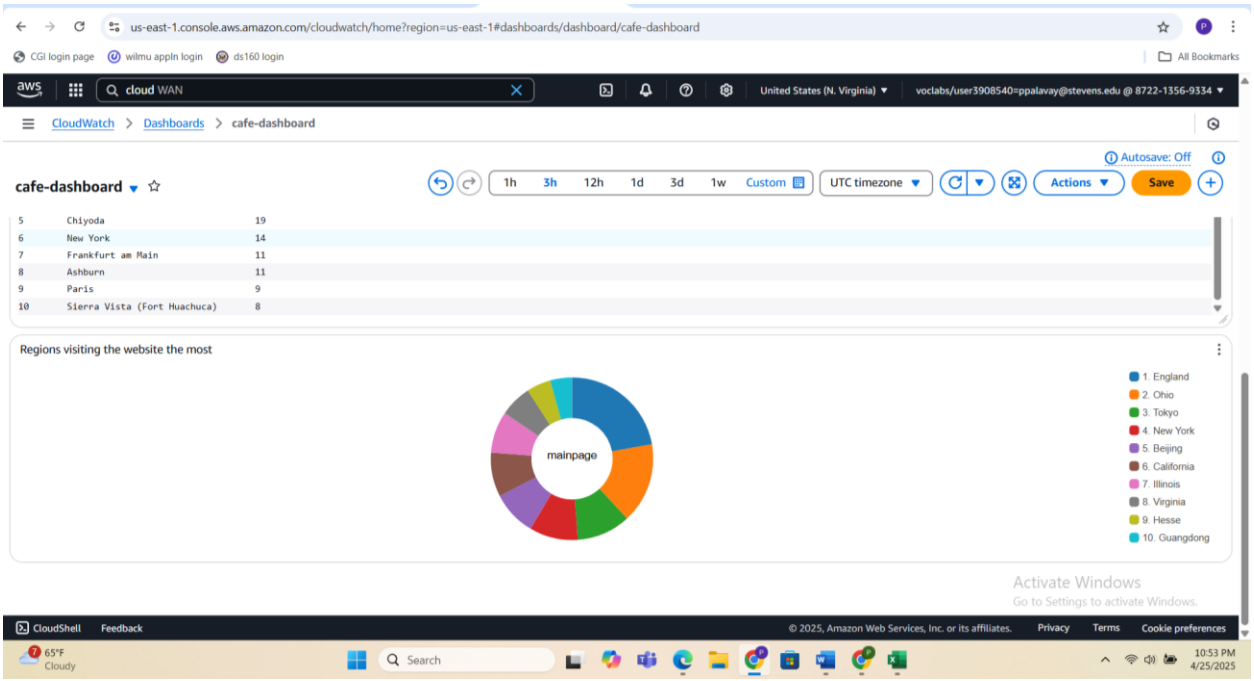


Create a third widget the widget type is **pie**

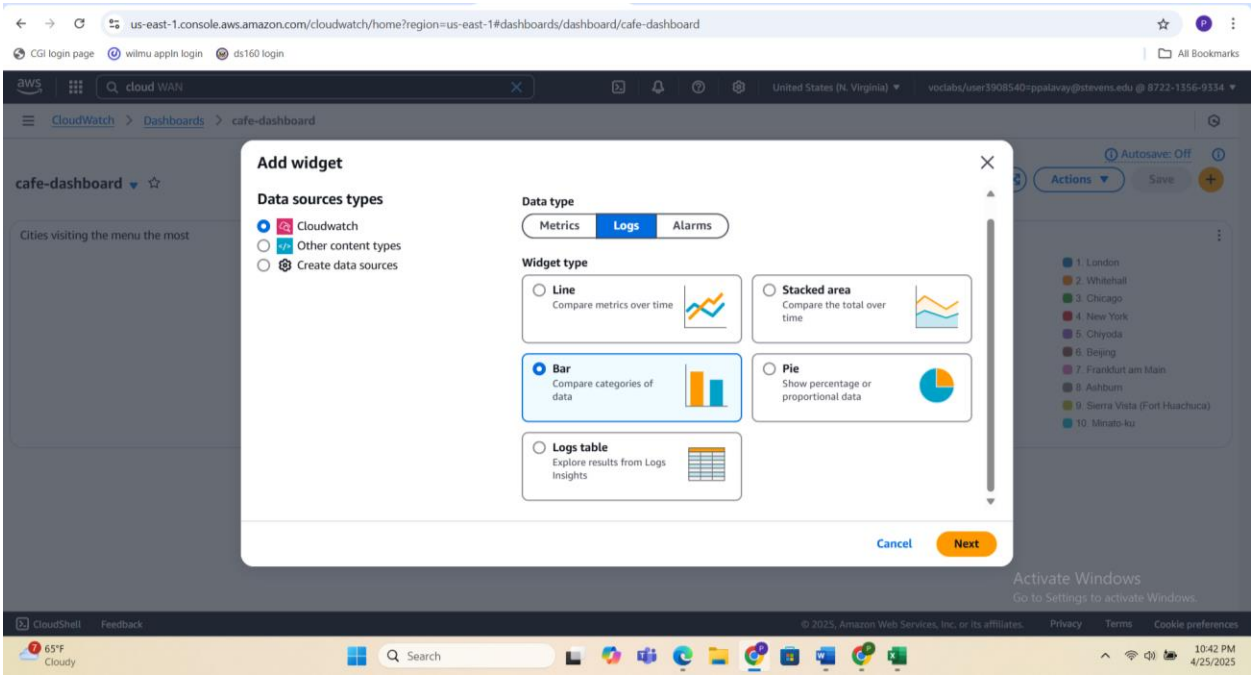


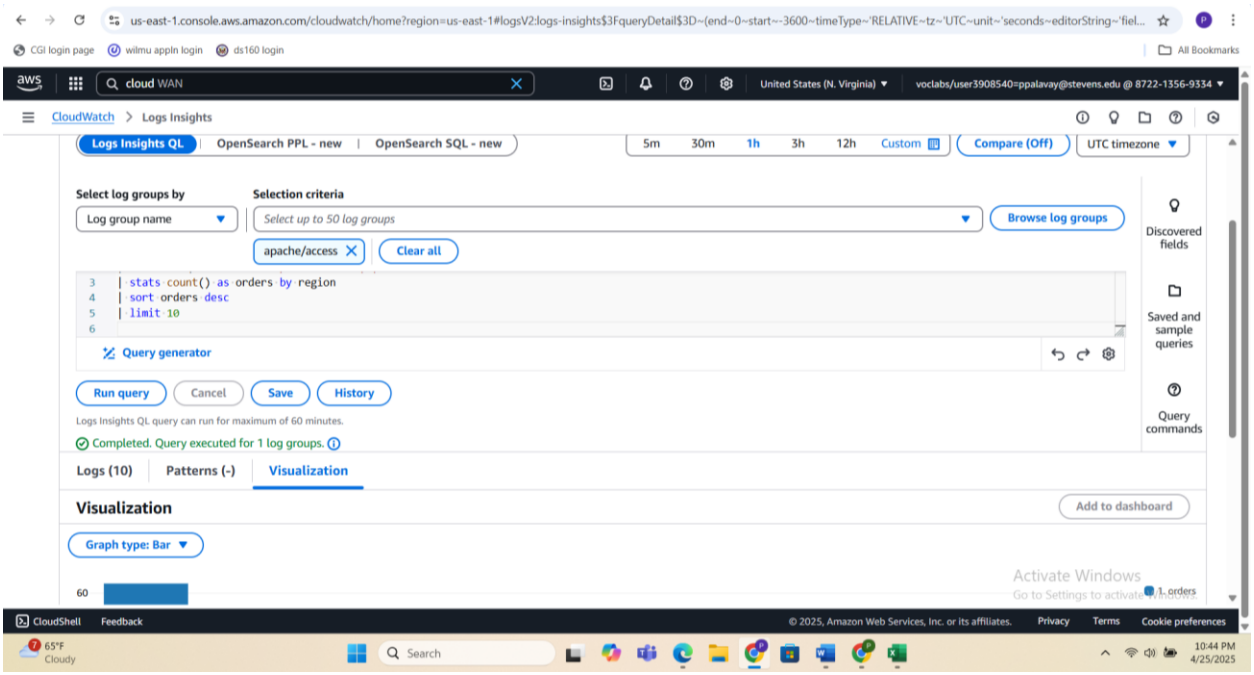


Reanme the widget as Regions visiting the website most

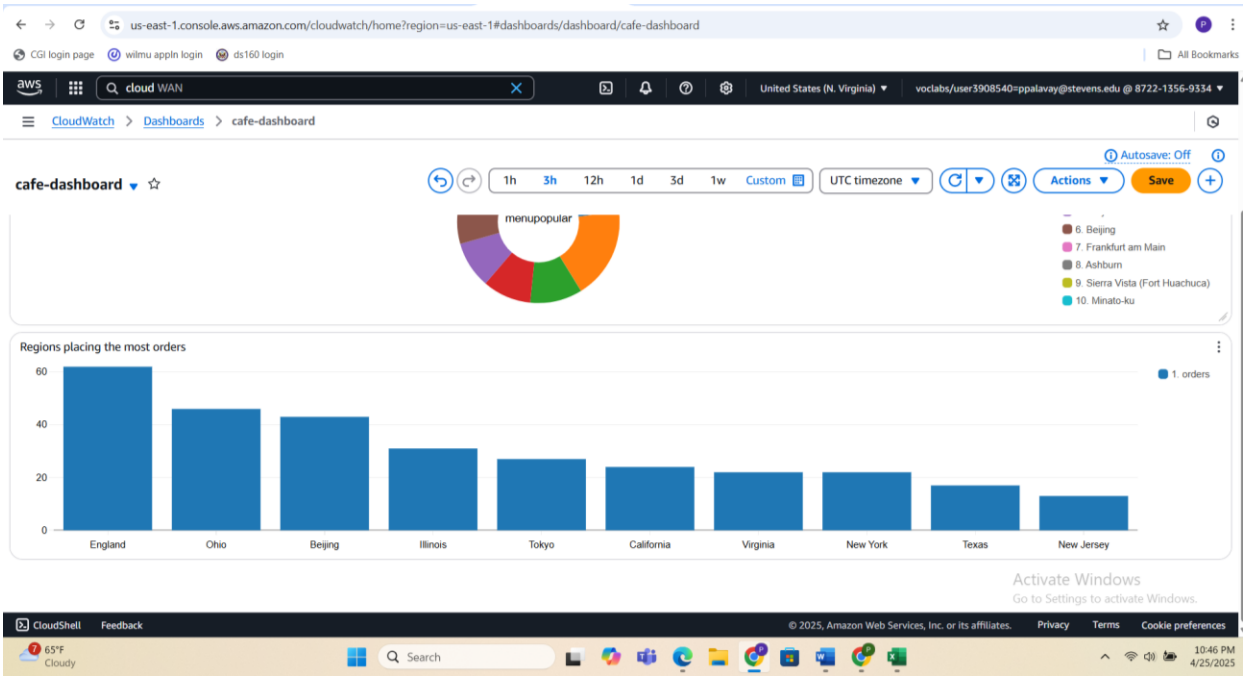


Fourth Widget :widget type-Bar graph

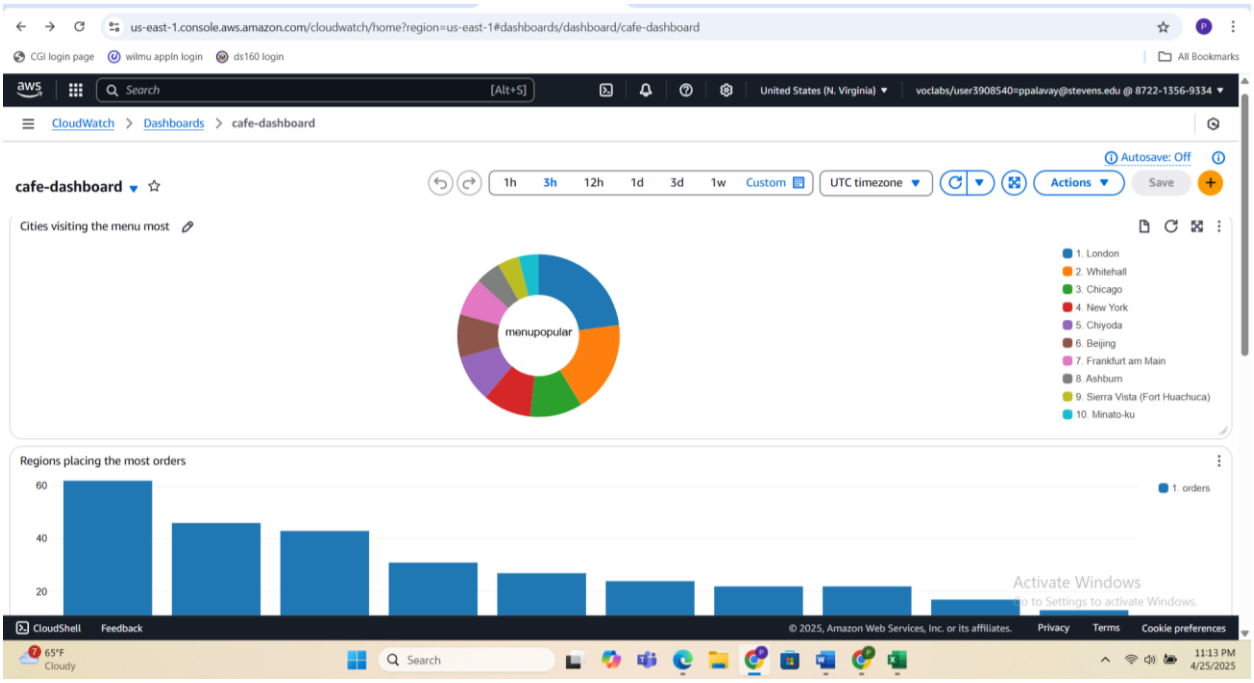


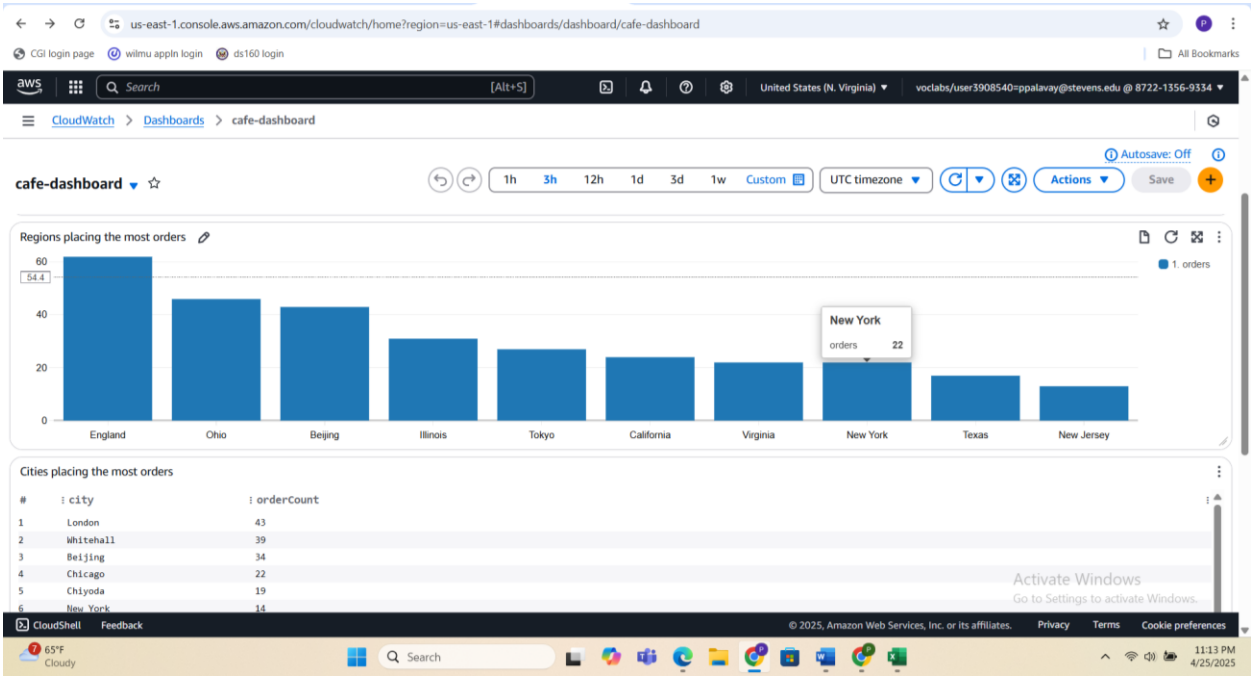


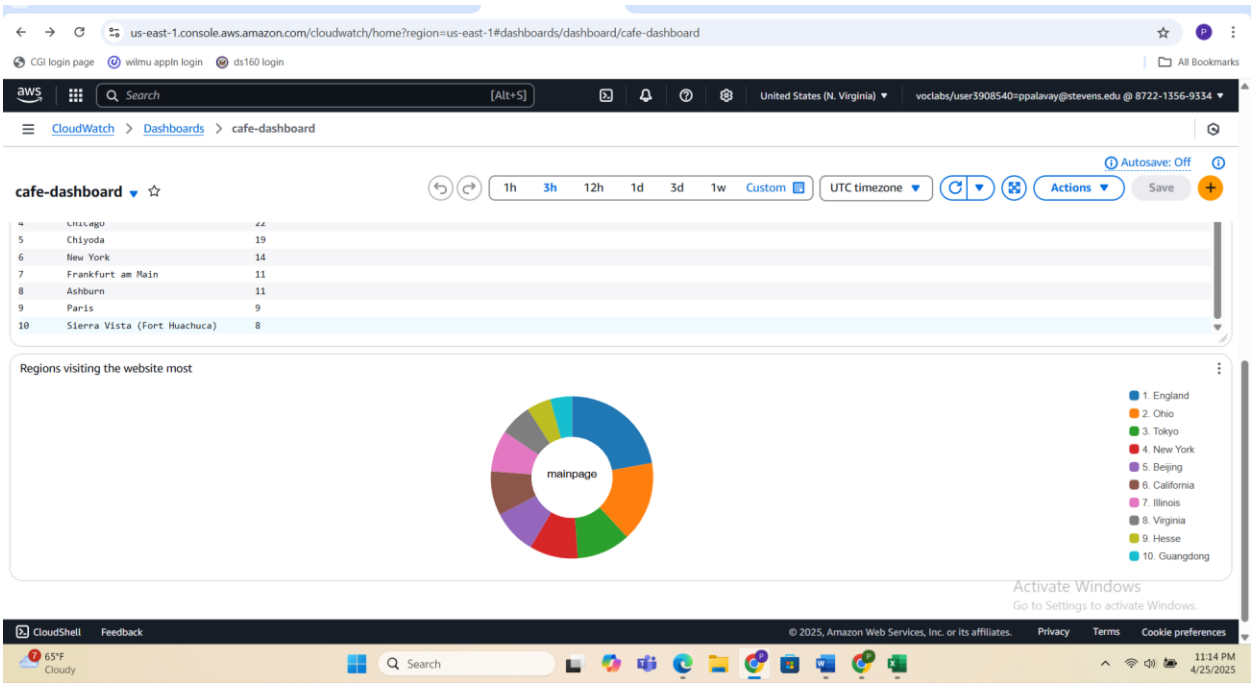
Rename this as Regions placing the most orders



We can see all the four widgets clearly in the café-dashboard


us-east-1.console.aws.amazon.com/cloudwatch/home?region=us-east-1#dashboards/dashboard/cafe-dashboard
CloudWatch > Dashboards > cafe-dashboard
cafe-dashboard
Autosave: Off
1h 3h 12h 1d 3d 1w Custom
UTC timezone
Actions
Save
Cities visiting the menu most
menupopular
1. London
2. Whitehall
3. Chicago
4. New York
5. Chiyoda
6. Beijing
7. Frankfurt am Main
8. Ashburn
9. Sierra Vista (Fort Huachuca)
10. Minato-ku
Regions placing the most orders
1. orders
60
40
20
Activate Windows
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
65°F
Cloudy
11:13 PM
4/25/2025


us-east-1.console.aws.amazon.com/cloudwatch/home?region=us-east-1#dashboards/dashboard/cafe-dashboard
CloudWatch > Dashboards > cafe-dashboard
cafe-dashboard
Autosave: Off
1h 3h 12h 1d 3d 1w Custom
UTC timezone
Actions
Save
Regions placing the most orders
1. orders
60
54.4
40
20
0
New York
orders 22
England
Ohio
Beijing
Illinois
Tokyo
California
Virginia
New York
Texas
New Jersey
Cities placing the most orders
city orderCount
1 London 43
2 Whitehall 39
3 Beijing 34
4 Chicago 22
5 Chiyoda 19
6 New York 14
Activate Windows
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
65°F
Cloudy
11:13 PM
4/25/2025



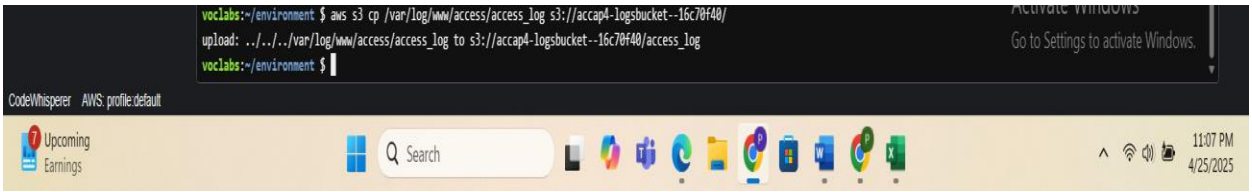
Upload Logs to S3 (Final Backup)

We also need to **copy your access logs** to an Amazon **S3 bucket** as a backup.

Here's how:

1. In your **Cloud9 terminal**, run:

   aws s3 cp/var/log/www/access/access_log s3://accap4-logsbucket—16c70f40/



We can see upload clearly in the terminal.

Summary:

1. Overview of the Project

The café website generates clickstream data through user interactions such as clicks, page views, and purchases. This data can provide valuable insights into how users navigate the website, which menu items are popular, and where visitors are coming from geographically. By leveraging AWS services, we set up a scalable, reliable, and efficient data pipeline for capturing and analyzing this data.

Our data pipeline incorporated several components:

- **Apache Web Server Logs**: The Apache web server generates logs that contain information about each user's visit, such as the pages they accessed, the time spent on each page, their geographic location (through IP address), and their interaction with menu items.
- **CloudWatch Agent**: The CloudWatch agent was installed on the Apache web server to stream the logs to CloudWatch Logs for centralized monitoring and analysis.
- **Data Processing**: The logs were processed to convert them into a more structured JSON format, with additional geolocation data added to enrich the analysis.
- **Storage**: Processed logs were backed up to Amazon S3, providing a durable and cost-effective storage solution for future analytics and historical data review.
- **Dashboarding and Visualization**: We created a CloudWatch Dashboard to visually present the most critical metrics, including the number of visitors, the number of purchases, and geographic breakdowns of users.

The final analysis of the data revealed a 74.85% conversion rate, which is a strong indicator of the effectiveness of the website in converting visitors into customers. Moreover, the geographic breakdown highlighted key cities where the café's online presence is particularly strong, including Madrid, London, and New York.

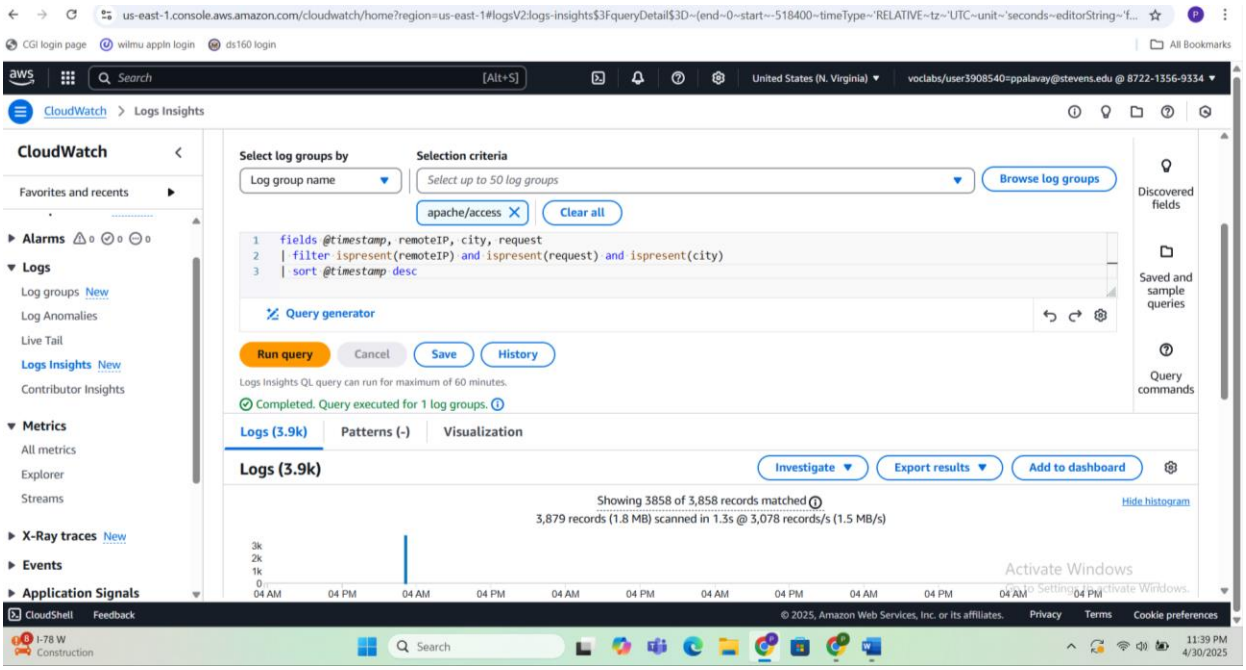
EXTRA CREDITs:phase1)

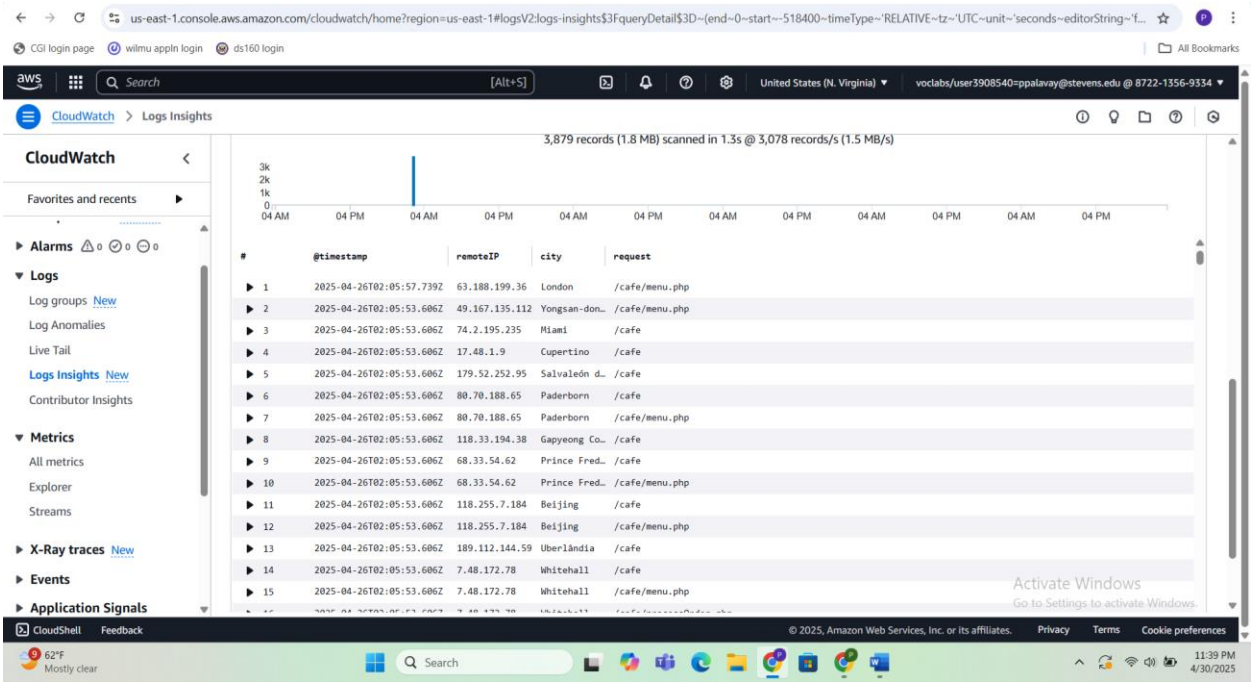
Data validation and error handling

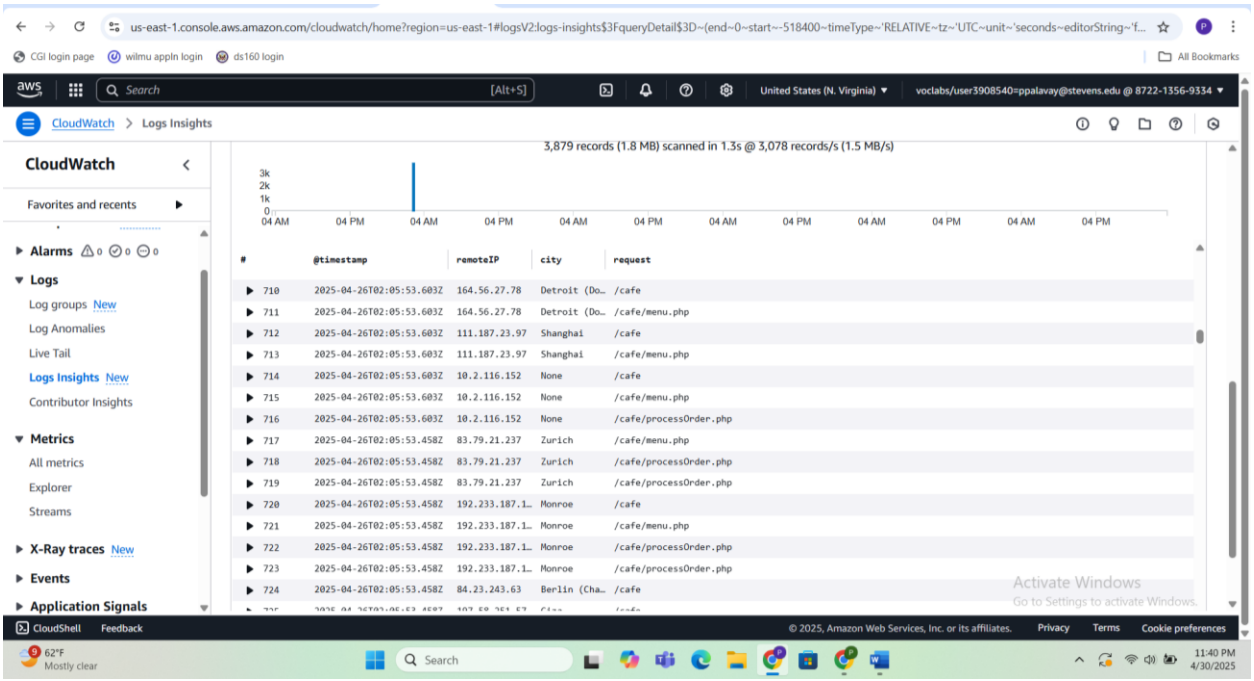
1. This filters out bad records that are missing fields.

```
fields @timestamp, remoteIP, city, request
| filter ispresent(remoteIP) and ispresent(request) and ispresent(city)
| sort @timestamp desc
```

These are valid logs

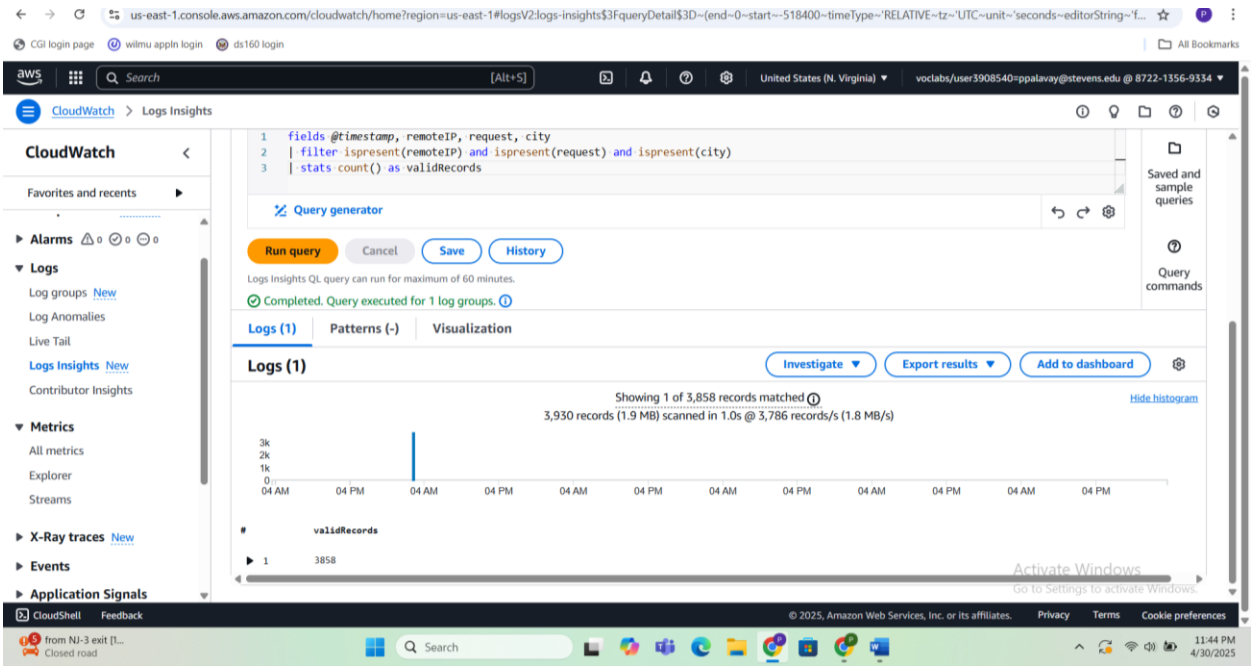






This query give s the count of the valid logs

```
fields @timestamp, remoteIP, request, city
| filter ispresent(remoteIP) and ispresent(request) and ispresent(city)
| stats count() as validRecords .
```

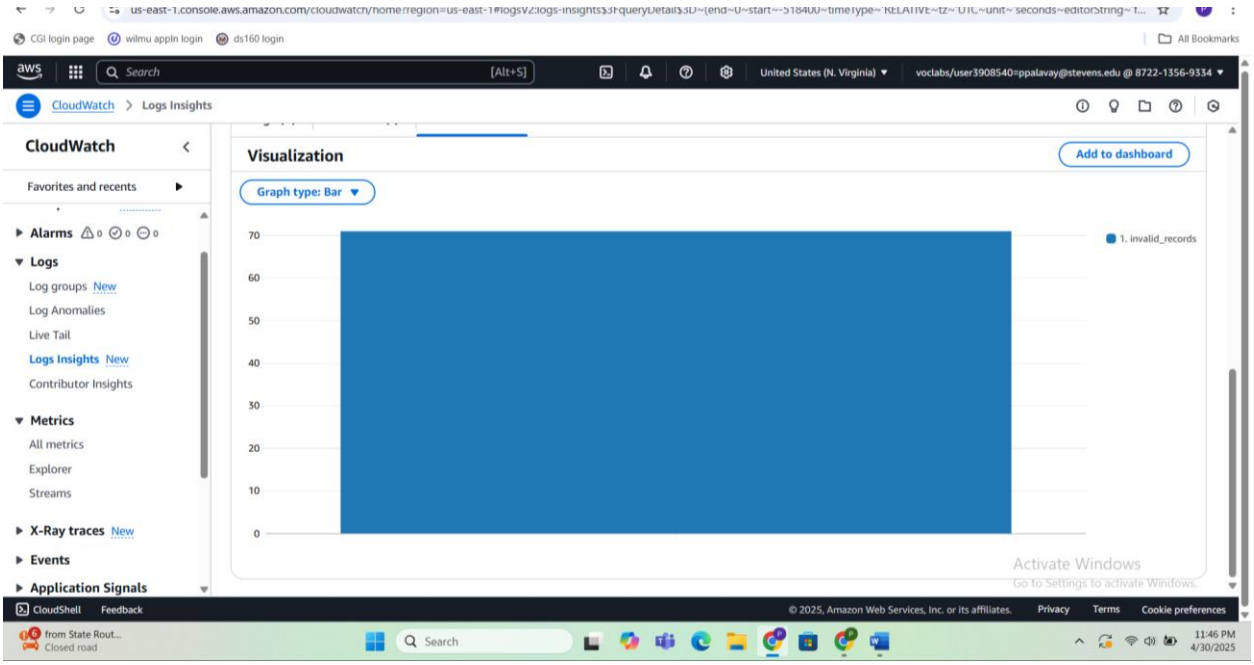


This query gives the count of invalid records

fields @timestamp, remoteIP, request, city

| filter not ispresent(remoteIP) or not ispresent(request) or not ispresent(city)

| stats count() as invalid_records



Phase 7 queries:

1.fields remoteIP, city

| filter request = "/cafe/menu.php"

| stats count() as menupopular by city

| sort menupopular desc

| limit 10

2.fields remoteIP, city

| filter request = "/cafe/processOrder.php"

| stats count() as orderCount by city

| sort orderCount desc

| limit 10

3.

```
fields remoteIP, region

| filter request = "/cafe"

| stats count() as mainpage by region

| sort mainpage desc

| limit 10
```

4.

```
fields remoteIP, region

| filter request = "/cafe/processOrder.php"

| stats count() as orders by region

| sort orders desc

| limit 10
```