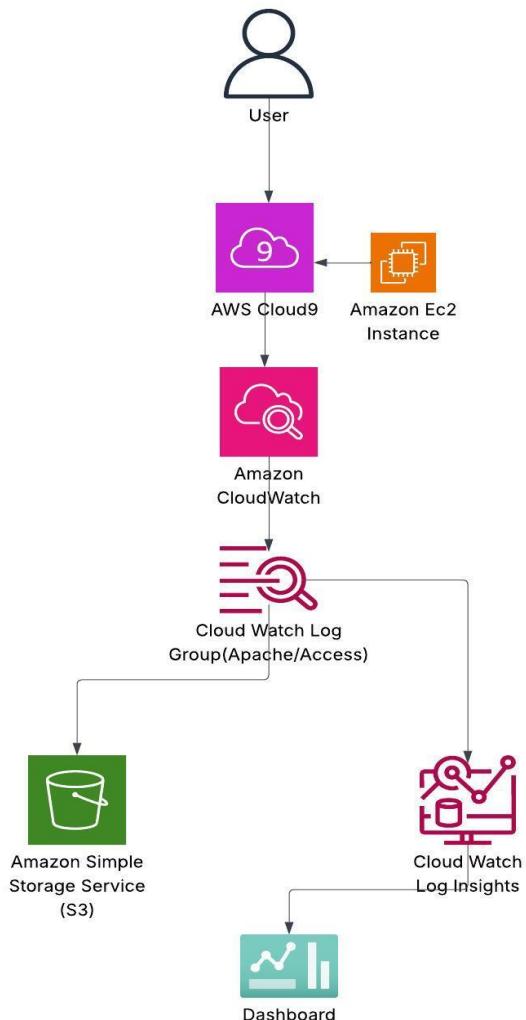


PHASE1:

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.micro

*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

The screenshot shows the AWS Pricing Calculator interface. At the top, it displays the total cost for a 12-month period: **50.40 USD**, which includes the upfront cost. Below this, there is a table titled "My Estimate" showing the breakdown of costs for different AWS services:

Service Name	Status	Upfront cost	Monthly cost	Description	Region	Config Summary
Amazon CloudWatch	-	0.00 USD	2.55 USD	-	US East (N. Virginia)	GetMetricData: Nu...
Amazon Simple St...	-	0.00 USD	1.39 USD	-	US East (N. Virginia)	S3 Standard stora...
Amazon EC2	-	0.00 USD	0.21 USD	-	US East (N. Virginia)	Tenancy (Shared In...
Amazon Athena	-	0.00 USD	0.05 USD	-	US East (N. Virginia)	Total number of q...

At the bottom of the page, there are links for Privacy, Site terms, and Cookie preferences, along with a copyright notice for 2025. The system status bar at the bottom right shows the date (4/26/2025), time (7:30 PM), battery level, and signal strength.

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.

The screenshot shows the AWS EC2 Instances dashboard. On the left, a sidebar lists navigation options: Instances, Images, Elastic Block Store, Network & Security, and CloudShell. The main area displays a table titled 'Instances (1/1) Info' with one row. The instance details are as follows:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
aws-cloud9-Cl...	i-0f5c6d7607928bf40	Running	t3.medium	3/3 checks passed	View alarms +	us-east-1a	ec2-13-217-

Below the instance table, there is a detailed view for the selected instance, showing its security group rules. The 'Inbound rules' tab is active, displaying two entries:

Name	Security group rule ID	Port range	Protocol	Source	Security group
-	sgr-060114d13365bc1ba	22	TCP	35.172.155.96/27	aws-cloud9-Clc
-	sgr-0262cb712ebc29c72	22	TCP	35.172.155.192/27	aws-cloud9-Clc

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

The screenshot shows the AWS Security Groups dashboard. On the left, a sidebar lists navigation options: Instances, Images, Elastic Block Store, Network & Security, and CloudShell. The main area displays a table titled 'Security Groups (1/3) Info' with three rows. The security group details are as follows:

Name	Security group ID	Security group name	VPC ID	Description
aws-cloud9-Cloud9...	sg-0d4a3374e5205f3ae	aws-cloud9-Cloud9-Lab-IDE-3a33ac0647754e53a94f03b604d43c14	ypc-0354622352e08329e	Security group for A
-	sg-07b5e710cab5b3e23	default	ypc-0495d32086625ce2d	default VPC security
-	sg-0d82227822b08c662	default	ypc-0354622352e08329e	default VPC security

Below the security group table, there is a detailed view for the selected security group, showing its inbound rules. The 'Inbound rules' tab is active, displaying two entries:

Name	Security group rule ID	IP version	Type	Protocol	Port range
-	sgr-060114d13365bc1ba	IPv4	SSH	TCP	22
-	sgr-0262cb712ebc29c72	IPv4	SSH	TCP	22

Choose the **Inbound rules** tab.

The screenshot shows the AWS Cloud9 - Lab IDE interface. At the top, there's a navigation bar with links like 'EC2', 'sg-0d4a3374e5205f3ae - aws-cloud9-Cloud9-Lab-IDE-3a33ac0647754e53a94f03b604d43c14-InstanceSecurityGroup-b4a14Q1p8qK8', and 'Edit inbound rules'. Below the navigation is a search bar and a toolbar with icons for copy, paste, and refresh.

The main content area is titled 'Edit inbound rules' with a 'Info' link. It displays two existing rules:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-060114d13365bc1ba	SSH	TCP	22	Custom	35.172.155.96/27
sgr-0262cb712ebc29c72	SSH	TCP	22	Custom	35.172.155.192/27

At the bottom of the table, there's a blue 'Add rule' button. To the right of the table are three buttons: 'Cancel', 'Preview changes', and 'Save rules'.

Below the main content area, there's a watermark for 'Activate Windows' with the text 'Go to Settings to activate Windows.' and a link to 'Cookie preferences'.

Click **Edit Inbound Rules**.

Add this rule:

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

Inbound rules

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-060114d13365bc1ba	SSH	TCP	22	Custom	35.172.155.96/27
sgr-0262cb712ebc29c72	SSH	TCP	22	Custom	35.172.155.192/27
-	HTTP	TCP	80	Anyw...	0.0.0.0/0

Add rule

⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

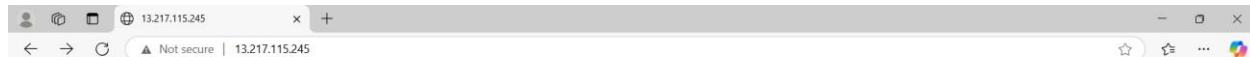
Cancel Preview changes Save rules

Save the rule.

Inbound security group rules successfully modified on security group (sg-0d4a3374e5205f3ae | aws-cloud9-Cloud9-Lab-IDE-3a33ac0647754e53a94f03b604d43c14-InstanceSecurityGroup-b4a14Q1p8qk8)

Security Groups (3)

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



Web server is working. Access the [café website](#).



Click on the website to see menu, order history.

The screenshot shows a web browser window with the title 'Café'. The address bar displays '13.217.115.245/cafe/'. The page content includes:

- A header with navigation links: Home, About Us, Contact Us, Menu, Order History.
- A large image showing various baked goods like croissants, pastries, and cakes.
- A descriptive paragraph: "Our café offers an assortment of delicious and delectable pastries and coffees that will put a smile on your face. From cookies to croissants, tarts and cakes, each treat is especially prepared to excite your tastebuds and brighten your day!"
- Two promotional banners:
 - "Frank bakes a rich variety of cookies. Try them all!" with an image of cookies.
 - "Our tarts are always a customer favorite!" with an image of tarts.
- A footer with a 'Sports headline' link, pinned icons, and a 'Activate Windows' notification.

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.

The screenshot shows the AWS Management Console with the search bar set to 'CLOUD9'. The left sidebar is expanded to show the 'EC2' section, which includes 'Instances' (with 'Instances' selected), 'Images', and 'Elastic Block Store'. The main content area displays the 'Services' section under 'Cloud9', featuring 'Cloud9' (A Cloud IDE for Writing, Running, and Debugging Code), 'Amazon CodeCatalyst' (Integrated DevOps Service), and 'AWS Cloud Map' (Build a dynamic map of your cloud). Below this, the 'Features' section lists 'Cloud WAN' (VPC feature) and 'Namespaces' (AWS Cloud Map feature). A message 'Were these results helpful?' with 'Yes' and 'No' buttons is visible. On the right, a detailed view of a specific instance is shown, including its ID (47754e53a94f03b604d43c14), private and public IPv4 addresses, elastic IP addresses, and an auto-scaling group name. The status bar at the bottom indicates it's 4:58 PM on 4/25/2025.

Open the terminal from cloud 9 environment

The screenshot shows the AWS Cloud9 control interface with the search bar set to 'Cloud9'. The left sidebar shows 'My environments', 'Shared with me', and 'All account environments'. The main content area displays the 'Environments (1)' section, listing a single environment named 'Cloud9-Lab-IDE'. It shows details like 'Name: Cloud9-Lab-IDE', 'Type: EC2 instance', 'Connection: Secure Shell (SSH)', and 'Owner: arn:aws:sts::872213569334:assumed-role/vocabs/user3908540-ppalavay@stevens.edu'. A message at the top encourages exploring AWS Toolkits. The status bar at the bottom indicates it's 4:59 PM on 4/25/2025.

Run this command

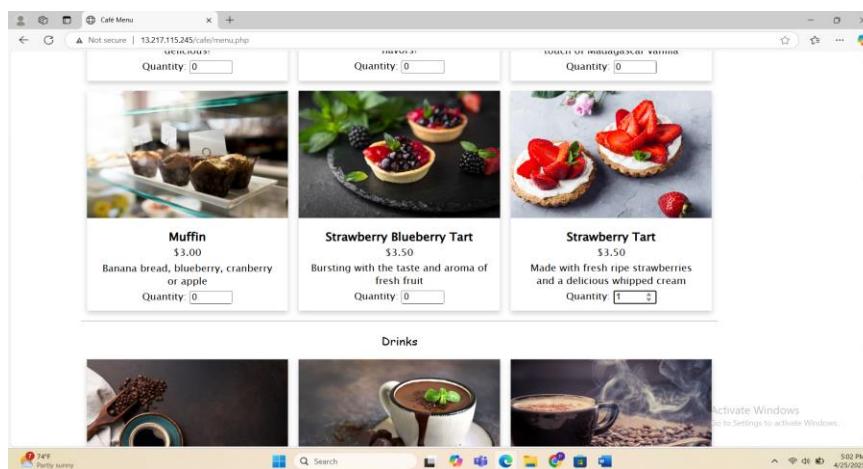
```
sudo tail -f /var/log/httpd/access_log
```

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

The screenshot shows a browser window with several tabs open. The active tab is a Cloud9 terminal window titled "sudo - ip-10-0-1-67.ec2.i" showing log output from a file named "access_log". The log contains numerous entries from various IP addresses, mostly from 108.5.108.167, detailing HTTP requests for images like "Strawberry-Tarts.jpg" and "Cake-Vitrine.jpg". The terminal also shows command-line interactions with "voclaee" and "tail". Below the terminal is a "Welcome" message from AWS Cloud9: "Welcome to your development environment". On the left, there's a sidebar with project files including "c9", "samplelogs", "log-gen.py", "log-gen_geo.py", and "README.md". The status bar at the bottom indicates "CodeWhisperer AWS profile default" and the date "4/25/2025".

Going back to the browser and interacting with the website:

- Visit pages, Submit an order, Refresh the homepage



Café

Home Menu Order History

Order History

Order Number: 1 Date: 2025-04-25 Time: 17:02:58 Total Amount: \$3.50

Item	Price	Quantity	Amount
Strawberry Tart	\$3.50	1	\$3.50

© 2020, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Activate Windows
Go to Settings to activate Windows.

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

```

36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36 Edg/135.0.0.0"
108.5.108.167 - [25/Apr/2025:21:01:00 +0000] "GET /cafe/images/Croissants.jpg HTTP/1.1" 200 351781 "http://13.217.115.245/cafe/menu.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36 Edg/135.0.0.0"
108.5.108.167 - [25/Apr/2025:21:01:00 +0000] "GET /cafe/images/Latte-Chip-Cookies.jpg HTTP/1.1" 200 488820 "http://13.217.115.245/cafe/menu.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36 Edg/135.0.0.0"
108.5.108.167 - [25/Apr/2025:21:01:00 +0000] "GET /cafe/images/Latte.jpg HTTP/1.1" 200 19081 "http://13.217.115.245/cafe/menu.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36 Edg/135.0.0.0"
108.5.108.167 - [25/Apr/2025:21:01:00 +0000] "GET /cafe/images/Muffins.jpg HTTP/1.1" 200 243718 "http://13.217.115.245/cafe/menu.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36 Edg/135.0.0.0"
108.5.108.167 - [25/Apr/2025:21:01:00 +0000] "GET /cafe/images/Donuts.jpg HTTP/1.1" 200 380753 "http://13.217.115.245/cafe/menu.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36 Edg/135.0.0.0"
108.5.108.167 - [25/Apr/2025:21:01:00 +0000] "GET /cafe/images/Coffee.jpg HTTP/1.1" 200 631884 "http://13.217.115.245/cafe/menu.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36 Edg/135.0.0.0"
:1 - - [25/Apr/2025:21:01:00 +0000] "OPTIONS * HTTP/1.1" 200 "-" "Apache/2.4.62 () (internal dummy connection)"
108.5.108.167 - [25/Apr/2025:21:02:58 +0000] "POST /cafe/processOrder.php HTTP/1.1" 200 1250 "http://13.217.115.245/cafe/menu.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36 Edg/135.0.0.0"
108.5.108.167 - [25/Apr/2025:21:03:04 +0000] "GET /cafe/orderHistory.php HTTP/1.1" 200 1114 "http://13.217.115.245/cafe/processOrder.php" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36 Edg/135.0.0.0"
157.55.39.25 - [25/Apr/2025:21:03:45 +0000] "GET /robots.txt HTTP/1.1" 404 196 "-" "Mozilla/5.0 AppleWebKit/537.36 (KHTML, like Gecko; compatible; bingbot/2.0; +http://www.bing.com/bingbot.htm) Chrome/116.0.1938.76 Safari/537.36"
52.167.144.157 - [25/Apr/2025:21:03:47 +0000] "GET / HTTP/1.1" 200 89 "-" "Mozilla/5.0 AppleWebKit/537.36 (KHTML, like Gecko; compatible; bingbot/2.0; +http://www.bing.com/bingbot.htm) Chrome/116.0.1938.76 Safari/537.36"
52.167.144.157 - [25/Apr/2025:21:04:18 +0000] "GET /cafe HTTP/1.1" 301 235 "-" "Mozilla/5.0 AppleWebKit/537.36 (KHTML, like Gecko; compatible; bingbot/2.0; +http://www.bing.com/bingbot.htm) Chrome/116.0.1938.76 Safari/537.36"

```

Activate Windows
Go to Settings to activate Windows.

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.

```

sudo -ip-10-0-1-67.ec2.i x Immediate
veclabs:/environment $ sudo cp /var/log/httpd/access_log /home/ec2-user/environment/initial_access_log
veclabs:/environment $ sudo yum install -y amazon-cloudwatch-agent
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
237 packages excluded due to repository priority protections
Resolving Dependencies
--> Running transaction check
-->> Package amazon-cloudwatch-agent.x86_64 0:1.300052.1-1.amzn2 will be installed
-->> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch      Version          Repository      Size
=====
Installing:
amazon-cloudwatch-agent   x86_64    1.300052.1-1.amzn2   amzn2-core      114 M

Transaction Summary
Install 1 Package

Total download size: 114 M
Installed size: 359 M
Downloading packages:

```

```

bash -ip-10-0-1-67.ec2.i x Immediate
Installing:
amazon-cloudwatch-agent   x86_64    1.300052.1-1.amzn2   amzn2-core      114 M

Transaction Summary
Install 1 Package

Total download size: 114 M
Installed size: 359 M
Downloading packages:
amazon-cloudwatch-agent-1.300052.1-1.amzn2.x86_64.rpm
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
create group cwagent, result: 0
create user cwagent, result: 0
  Installing : amazon-cloudwatch-agent-1.300052.1-1.amzn2.x86_64
  Verifying  : amazon-cloudwatch-agent-1.300052.1-1.amzn2.x86_64

Installed:
  amazon-cloudwatch-agent.x86_64 0:1.300052.1-1.amzn2

Complete!
veclabs:/environment $ 

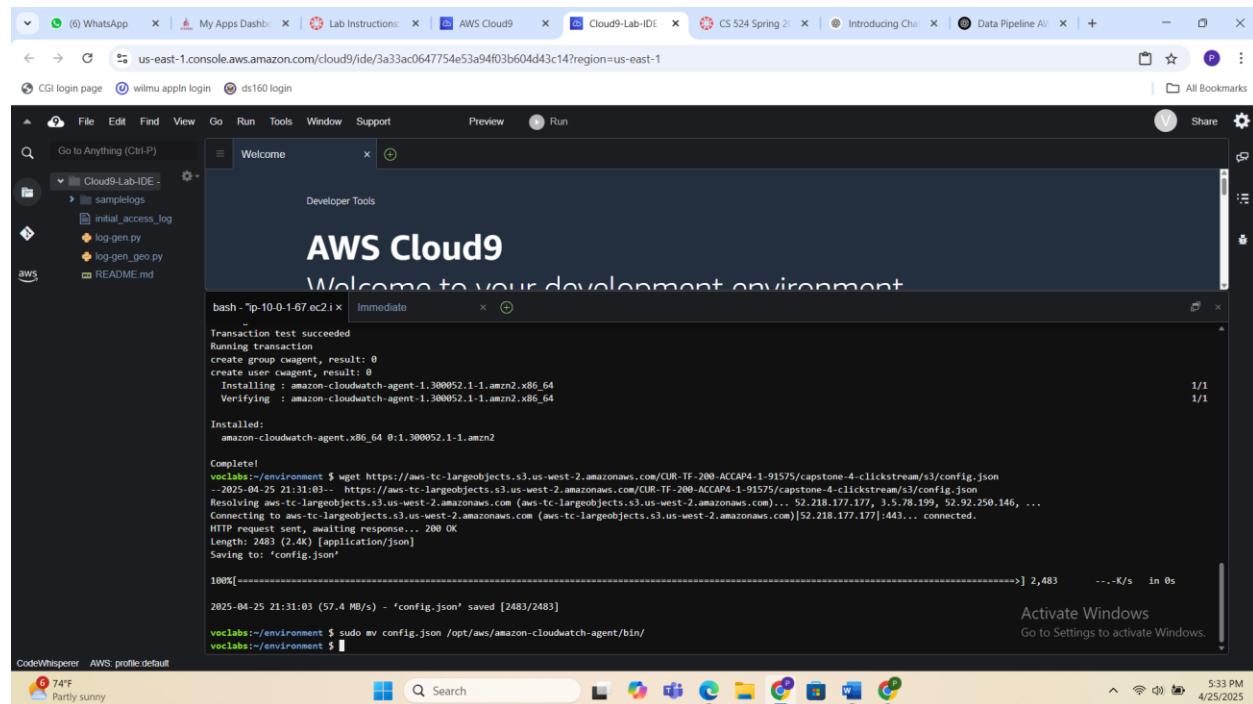
```

Download the Prebuilt Config File

wget <https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCP4-1-91575/capstone-4-clickstream/s3/config.json>

Run the command

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



The screenshot shows the AWS Cloud9 IDE interface. In the terminal window, the command `sudo mv config.json /opt/aws/amazon-cloudwatch-agent/bin/` is run, and the output shows the file being moved successfully.

```
bash ~ip-10-0-1-67.ec2.i x Immediate x + 1/1 1/1
Transaction test succeeded
Running transaction
create group cwaagent, result: 0
create user cwaagent, result: 0
    Installing : amazon-cloudwatch-agent-1.300052.1-1.amzn2.x86_64
    Verifying : amazon-cloudwatch-agent-1.300052.1-1.amzn2.x86_64

Installed:
    amazon-cloudwatch-agent.x86_64 0:1.300052.1-1.amzn2

Complete!
voclab:~/environment $ wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCPA4-1-91575/capstone-4-clickstream/s3/config.json
--2025-04-25 21:31:03-- https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCPA4-1-91575/capstone-4-clickstream/s3/config.json
Resolving aws-tc-largeobjects.s3.us-west-2.amazonaws.com (aws-tc-largeobjects.s3.us-west-2.amazonaws.com) ... 52.218.177.177, 3.5.78.199, 52.92.250.146, ...
Connecting to aws-tc-largeobjects.s3.us-west-2.amazonaws.com (aws-tc-largeobjects.s3.us-west-2.amazonaws.com)|52.218.177.177|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2483 [application/json]
Saving to: "config.json"

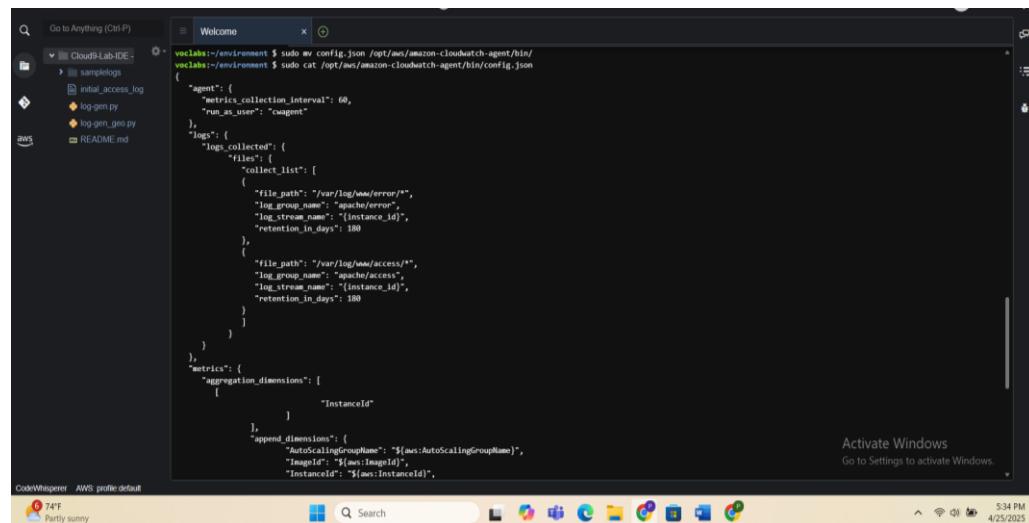
100%[=====] 2,483 --.-K/s in 0s
2025-04-25 21:31:03 (57.4 MB/s) - "config.json" saved [2483/2483]

voclab:~/environment $ sudo mv config.json /opt/aws/amazon-cloudwatch-agent/bin/
voclab:~/environment $
```

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.



The screenshot shows the AWS Cloud9 IDE interface with the config.json file open in the code editor. The file contains configuration for the CloudWatch Agent, specifying log collection rules and metrics dimensions.

```
Go to Anything (Ctrl+P) Welcome x
Cloud9-Lab-IDE : samplelogs initial_access_log log-gen.py log-gen_geo.py README.md
voclab:~/environment $ sudo mv config.json /opt/aws/amazon-cloudwatch-agent/bin/
voclab:~/environment $ sudo cat /opt/aws/amazon-cloudwatch-agent/bin/config.json
{
    "agent": {
        "metrics_collection_interval": 60,
        "run_as_user": "cwaagent"
    },
    "logs": {
        "log_collected": [
            {
                "files": [
                    {
                        "collect_list": [
                            {
                                "file_path": "/var/log/sys/error/*",
                                "log_group_name": "sys-error",
                                "log_stream_name": "${instance_id}",
                                "retention_in_days": 360
                            },
                            {
                                "file_path": "/var/log/www/access/*",
                                "log_group_name": "apache-access",
                                "log_stream_name": "${instance_id}",
                                "retention_in_days": 360
                            }
                        ]
                    }
                ],
                "metrics": {
                    "aggregation_dimensions": [
                        {
                            "dimensions": [
                                "InstanceId"
                            ],
                            "append_dimensions": {
                                "AutoScalingGroupName": "${aws:AutoScalingGroupName}",
                                "ImageId": "${aws:ImageId}",
                                "InstanceId": "${aws:InstanceId}"
                            }
                        }
                    ]
                }
            }
        ]
    }
}
```

```

    "append_dimensions": {
        "AutoScalingGroupName": "${aws:AutoScalingGroupName}",
        "ImageId": "${aws:ImageId}",
        "InstanceId": "${aws:InstanceId}",
        "InstanceType": "${aws:InstanceType}"
    },
    "metrics_collected": {
        "collected": [
            {
                "measurement": [
                    "used_percent"
                ],
                "resources": [
                    "*"
                ],
                "ignore_file_system_types": [
                    "sysfs", "devtmpfs"
                ]
            },
            {
                "measurement": [
                    "mem_used_percent"
                ],
                "metrics_collection_interval": 60
            },
            {
                "statsd": [
                    {
                        "metrics_aggregation_interval": 60,
                        "metrics_collection_interval": 10,
                        "service_address": ":8125"
                    }
                ]
            }
        ]
    }
}

```

Activate Windows
Go to Settings to activate Windows.

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

In -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.

```

voclabs:~/.aws/credentials $ In -s /etc/httpd/conf /home/ec2-user/environment/httpdconf
voclabs:~/.aws/credentials $ sudo chown -R ec2-user /etc/httpd/conf
voclabs:~/.aws/credentials $ 

```

Activate Windows
Go to Settings to activate Windows.

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\"}"
```

```
181 #
182 #ErrorLog "logs/error_log"
183 ErrorLog "/var/log/www/error/error_log"
184 ErrorLogFormat "{\"time\": \"%{usec_frac}t\", \"function\" : \"%m\", \"process\" : \"[pid%P]\", \"message\" : \"%M\"}"
185 |
186 #
187 #
188 # LogLevel: Control the number of messages logged to the error_log.
189 # Possible values include: debug, info, notice, warn, error, crit,
190 #
```

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%T%{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
```

```
195 #
196 # The following directives define some format nicknames for use with
197 # a CustomLog directive (see below).
198 #
199 #LogFormat "%h %l %u \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" combined
200 LogFormat '{ \"time\": \"%Y-%m-%d%T%{T}t.%{msec_frac}tZ\", \"process\":\"%D\", \"filename\":\"%f\", \"remoteIP\":\"%a\", \"host\":\"%V\", \"request\":\"%U\", \"query\":\"%q\"}
201 LogFormat '{ \"time\": \"%Y-%m-%d%T%{T}t.%{msec_frac}tZ\", \"process\":\"%D\", \"filename\":\"%f\", \"remoteIP\":\"%a\", \"host\":\"%V\", \"request\":\"%U\", \"query\":\"%q\"}
202 LogFormat "%h %l %u \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %0" combined
203 LogFormat '{ \"time\": \"%Y-%m-%d%T%{T}t.%{msec_frac}tZ\", \"process\":\"%D\", \"filename\":\"%f\", \"remoteIP\":\"%a\", \"host\":\"%V\", \"request\":\"%U\", \"query\":\"%q\"}
204 <IfModule logio_module>
205 # You need to enable mod_logio.c to use %I and %O
206 LogFormat "%h %l %u \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %I %O" combined
207 </IfModule>
208 #
209 #
210 # The location and format of the access logfile (Common Logfile Format).
```

Find:

```
CustomLog "logs/access_log" combined
```

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

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

```
219 #
220 # If you prefer a logfile with access, agent, and referer information
221 # (Combined Logfile Format) you can use the following directive.
222 #
223 CustomLog "logs/access_log" combined
224 CustomLog "/var/log/www/access/access_log" cloudwatch
225
226 </IfModule>
227
228 <IfModule alias_module>
229 #
230 # Redirect: Allows you to tell clients about documents that used to
231 # exist in your server's namespace, but do not anymore. The client
```

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
```

The screenshot shows a Windows desktop environment with several open windows. In the top taskbar, there are tabs for WhatsApp, My Apps Dashboard, Lab Instructions: Buil..., AWS Cloud9, Cloud9-Lab-IDE - A..., CS 524 Spring 2025, Data Pipeline AWS G..., and a few others. Below the taskbar, a browser window is open to the URL <https://us-east-1.console.aws.amazon.com/cloud9/ide/3a33ac0647754e53a94f03b604d43c14?region=us-east-1>. The page displays a code editor for an `httpd.conf` file. The code in the editor is as follows:

```
    # If you prefer a logfile with access, agent, and referer information
    # (Combined Logfile Format) you can use the following directive.
    CustomLog "logs/access_log" combined
    CustomLog "/var/log/www/access_log" cloudwatch

    </IfModule>
    <IfModule alias_module>
        # Redirect: Allows you to tell clients about documents that used to
        # exist in your server's namespace, but do not anymore. The client
        # will make a new request for the document at its new location.
        # Example:
        # Redirect permanent /foo http://www.example.com/bar
    
```

Below the code editor is a terminal window titled `bash - ip-10-0-1-67.ec2.i...` with the title bar `Immediate`. The terminal output is:

```
***** processing amazon-cloudwatch-agent *****
amazon-cloudwatch-agent is not configured. Applying amazon-cloudwatch-agent default configuration.
I! Trying to detect region from ec2 D! [EC2] Found active network interface I! imds retry client will retry 1 timesSuccessfully fetched the config and saved in /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.d/default.tmp
Start configuration validation...
2025/04/25 22:37:23 Reading json config file path: /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.d/default.tmp ...
2025/04/25 22:37:23 I! Valid Json Input schema.
2025/04/25 22:37:23 I! Extagger processor required because append_dimensions is set
2025/04/25 22:37:23 Configuration validation first phase succeeded
I! Detecting run_as_user...
I! Trying to detect region from ec2
D! [EC2] Found active network interface
I! imds retry client will retry 1 times
/opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent -schematest -config /opt/aws/amazon-cloudwatch-agent/etc/amazon-cloudwatch-agent.toml
Configuration validation second phase succeeded
Configuration validation succeeded
Created symlink from /etc/systemd/system/multi-user.target.wants/amazon-cloudwatch-agent.service to /etc/systemd/system/amazon-cloudwatch-agent.service.
```

The status bar at the bottom of the terminal window shows "Activate Windows Go to Settings to activate Windows".

To check if the agent is running properly

Run this:

```
sudo service amazon-cloudwatch-agent status
```

The terminal window shows the output of the command `sudo service amazon-cloudwatch-agent status`:

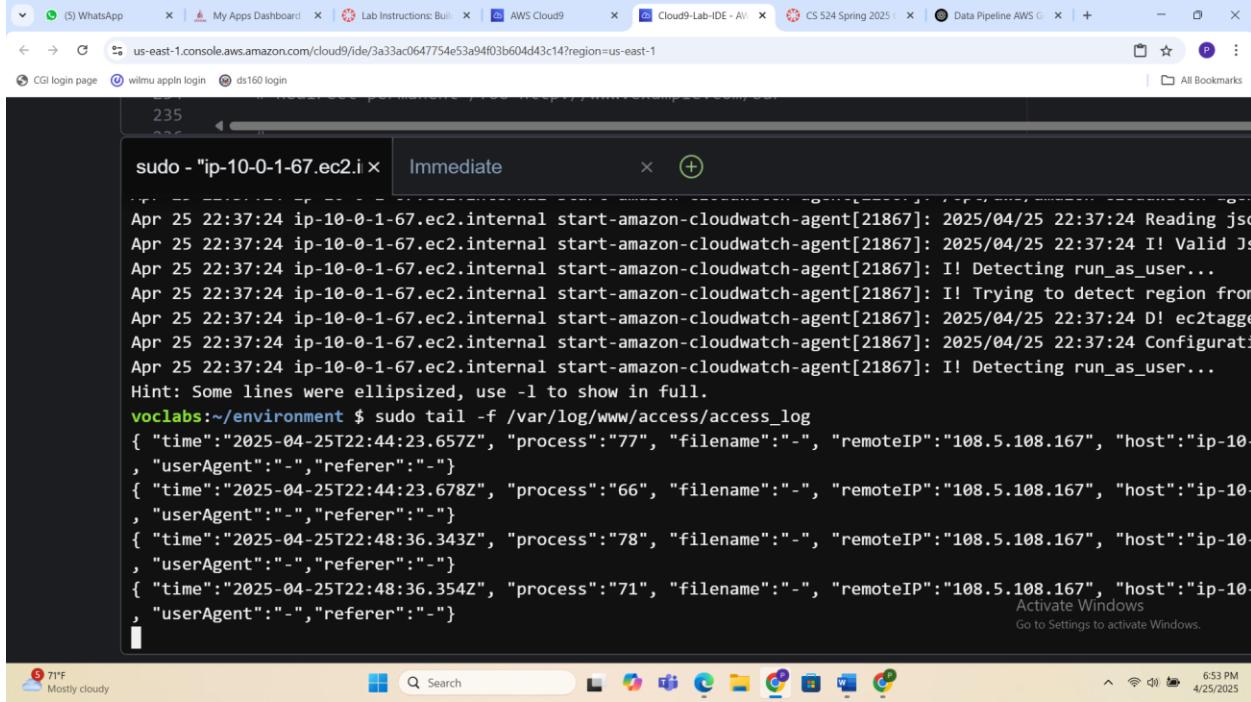
```
vocabs:~/environment $ sudo service amazon-cloudwatch-agent status
Redirecting to /bin/systemctl status amazon-cloudwatch-agent.service
● amazon-cloudwatch-agent.service - Amazon CloudWatch Agent
   Loaded: loaded (/etc/systemd/system/amazon-cloudwatch-agent.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2025-04-25 22:37:24 UTC; 17s ago
     Main PID: 21867 (amazon-cloudwat)
```

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
```



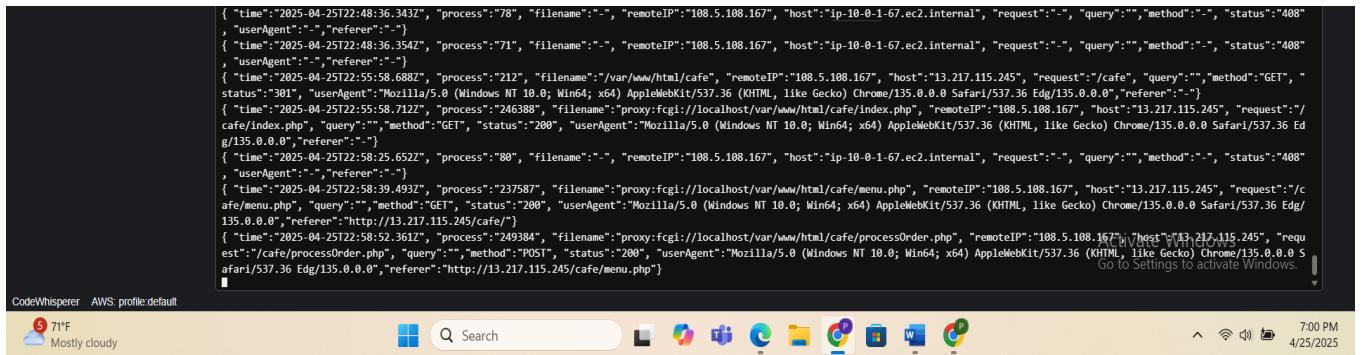
```
Apr 25 22:37:24 ip-10-0-1-67.ec2.internal start-amazon-cloudwatch-agent[21867]: 2025/04/25 22:37:24 Reading json
Apr 25 22:37:24 ip-10-0-1-67.ec2.internal start-amazon-cloudwatch-agent[21867]: 2025/04/25 22:37:24 I! Valid Js
Apr 25 22:37:24 ip-10-0-1-67.ec2.internal start-amazon-cloudwatch-agent[21867]: I! Detecting run_as_user...
Apr 25 22:37:24 ip-10-0-1-67.ec2.internal start-amazon-cloudwatch-agent[21867]: I! Trying to detect region from
Apr 25 22:37:24 ip-10-0-1-67.ec2.internal start-amazon-cloudwatch-agent[21867]: 2025/04/25 22:37:24 D! ec2tagge
Apr 25 22:37:24 ip-10-0-1-67.ec2.internal start-amazon-cloudwatch-agent[21867]: 2025/04/25 22:37:24 Configurati
Apr 25 22:37:24 ip-10-0-1-67.ec2.internal start-amazon-cloudwatch-agent[21867]: I! Detecting run_as_user...
Hint: Some lines were ellipsized, use -l to show in full.
vclabs:~/environment $ sudo tail -f /var/log/www/access/access_log
{ "time":"2025-04-25T22:44:23.657Z", "process":77, "filename": "-", "remoteIP": "108.5.108.167", "host": "ip-10-
, "userAgent": "-", "referer": "-"}
{ "time":"2025-04-25T22:44:23.678Z", "process":66, "filename": "-", "remoteIP": "108.5.108.167", "host": "ip-10-
, "userAgent": "-", "referer": "-"}
{ "time":"2025-04-25T22:48:36.343Z", "process":78, "filename": "-", "remoteIP": "108.5.108.167", "host": "ip-10-
, "userAgent": "-", "referer": "-"}
{ "time":"2025-04-25T22:48:36.354Z", "process":71, "filename": "-", "remoteIP": "108.5.108.167", "host": "ip-10-
, "userAgent": "-", "referer": "-"}  
Activate Windows
Go to Settings to activate Windows.
```

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



```
{ "time":"2025-04-25T22:48:36.343Z", "process":78, "filename": "-", "remoteIP": "108.5.108.167", "host": "ip-10-0-1-67.ec2.internal", "request": "", "query": "", "method": "-", "status": "408"
, "userAgent": "-", "referer": "-"}
{ "time": "2025-04-25T22:48:36.354Z", "process": "71", "filename": "-", "remoteIP": "108.5.108.167", "host": "ip-10-0-1-67.ec2.internal", "request": "", "query": "", "method": "-", "status": "408"
, "userAgent": "-", "referer": "-"}
{ "time": "2025-04-25T22:48:36.343Z", "process": "212", "filename": "/var/www/html/cafe", "remoteIP": "108.5.108.167", "host": "13.217.115.245", "request": "/cafe", "query": "", "method": "GET", "status": "301", "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36 Edg/135.0.0.0", "referer": "-"}
{ "time": "2025-04-25T22:55:58.712Z", "process": "246388", "filename": "/proxy.fcgi://localhost/var/www/html/cafe/index.php", "remoteIP": "108.5.108.167", "host": "13.217.115.245", "request": "/cafe/index.php", "query": "", "method": "GET", "status": "200", "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36 Edg/135.0.0.0", "referer": "-"}
{ "time": "2025-04-25T22:58:25.652Z", "process": "80", "filename": "-", "remoteIP": "108.5.108.167", "host": "ip-10-0-1-67.ec2.internal", "request": "", "query": "", "method": "-", "status": "408"
, "userAgent": "-", "referer": "-"}
{ "time": "2025-04-25T22:58:39.493Z", "process": "237587", "filename": "/proxy.fcgi://localhost/var/www/html/cafe/menu.php", "remoteIP": "108.5.108.167", "host": "13.217.115.245", "request": "/afe/menu.php", "query": "", "method": "GET", "status": "200", "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36 Edg/135.0.0.0", "referer": "http://13.217.115.245/cafe/"}
{ "time": "2025-04-25T22:58:52.361Z", "process": "249384", "filename": "/proxy.fcgi://localhost/var/www/html/cafe/processOrder.php", "remoteIP": "108.5.108.167", "host": "13.217.115.245", "request": "/cafe/processOrder.php", "query": "", "method": "POST", "status": "200", "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36 Edg/135.0.0.0", "referer": "http://13.217.115.245/cafe/menu.php"}  
Go to Settings to activate Windows.
```

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

The screenshot shows the AWS CloudWatch Log Groups interface. On the left, there's a navigation sidebar with sections for CloudWatch, AI Operations, Alarms, Logs (with Log groups selected), Metrics, and X-Ray traces. The main area is titled "Log groups (2)" and displays two entries: "apache/access" and "apache/error". Both entries are listed under the "Standard" log class. The "apache/access" entry has a retention period of "6 months". At the top right, there are buttons for "Actions", "View in Logs Insights", "Start tailing", and "Create log group". Below the table is a search bar and some filtering options. The bottom of the screen shows the standard Windows taskbar with various pinned icons.

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

We should see entries in JSON format showing page visits!

The screenshot shows the AWS CloudWatch Log Events interface for the "apache/access" log group. The left sidebar is identical to the previous screenshot. The main area is titled "Log events" and shows a table of log entries. The columns are "Timestamp" and "Message". There are approximately 15 entries listed, each showing a timestamp and a JSON-formatted log message. The messages describe various HTTP requests, such as "GET /index.html HTTP/1.1" and "GET / HTTP/1.1". The bottom right corner of the interface has a "Back to top" button.

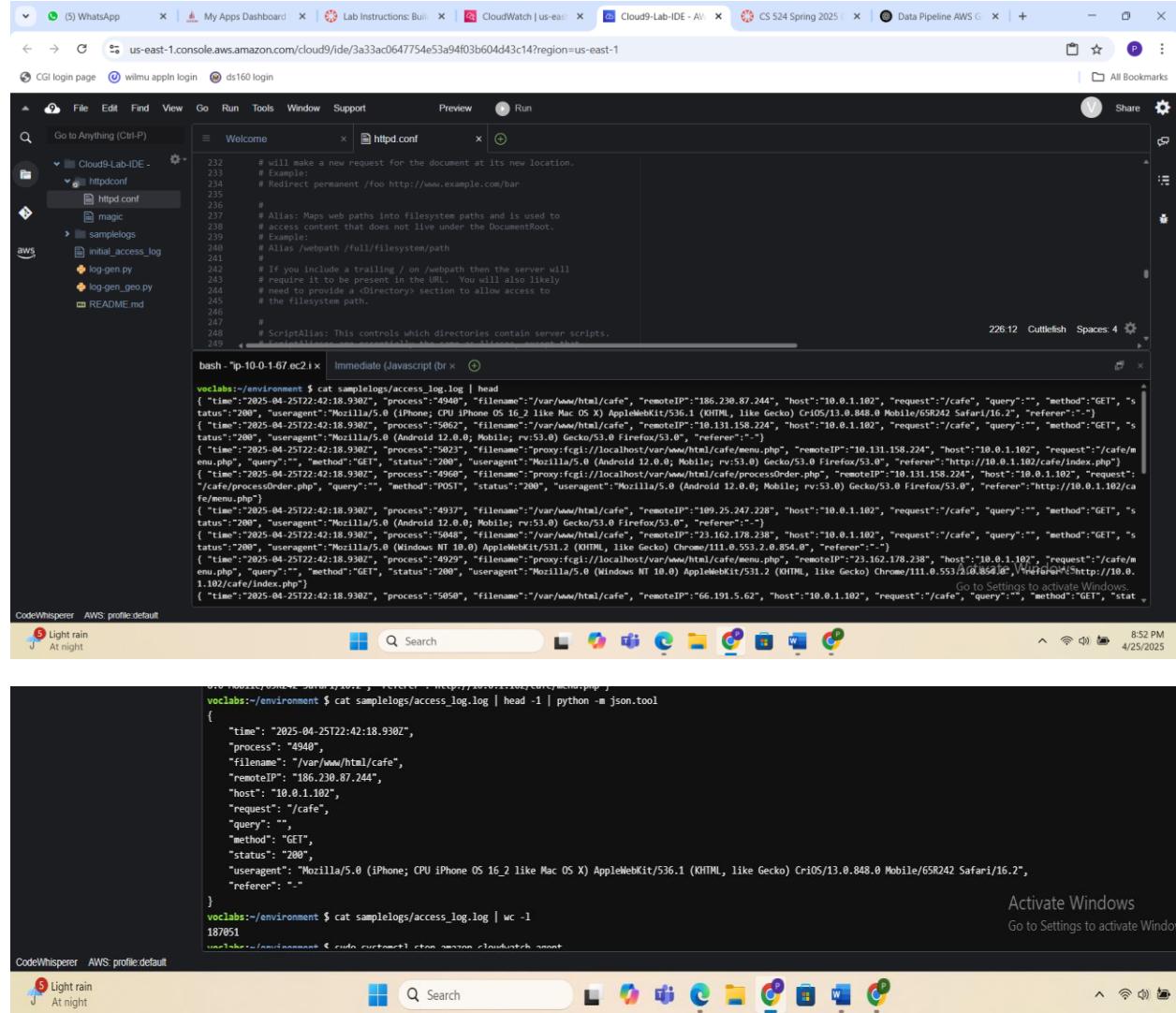
Phase 5: Use Simulated Logs

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
```



```
232 # will make a new request for the document at its new location.
233 # Example:
234 # Redirect permanent /foo http://www.example.com/bar
235 #
236 # Alias: Maps web paths into filesystem paths and is used to
237 # access content that does not live under the DocumentRoot.
238 # Examples:
239 # Alias /webpath /full/ filesystem path
240 # If you include a trailing / on webpath then the server will
241 # require it to be present in the URL. You will also likely
242 # need to provide a <Directory> section to allow access to
243 # the filesystem path.
244 #
245 # ScriptAlias: This controls which directories contain server scripts.
246 #
247 #
248 # ScriptAlias: This controls which directories contain server scripts.
249 
```

```
voclabs:~/environment $ cat samplelogs/access_log.log | head
{
  "time": "2025-04-25T22:42:18.930Z", "process": "4948", "filename": "/var/www/html/cafe", "remoteIP": "186.230.87.244", "host": "10.0.1.102", "request": "/cafe", "query": "", "method": "GET", "status": "200", "useragent": "Mozilla/5.0 (iPhone; CPU iPhone OS 16_2 like Mac OS X) AppleWebKit/536.1 (KHTML, like Gecko) CriOS/13.0.848.0 Mobile/65R242 Safari/16.2", "referrer": ""}
{
  "time": "2025-04-25T22:42:18.930Z", "process": "5062", "filename": "/var/www/html/cafe", "remoteIP": "10.131.158.224", "host": "10.0.1.102", "request": "/cafe", "query": "", "method": "GET", "status": "200", "useragent": "Mozilla/5.0 (Android 12.0.0; Mobile; rv:53.0) Gecko/53.0 Firefox/53.0", "referrer": ""}
{
  "time": "2025-04-25T22:42:18.930Z", "process": "5023", "filename": "proxy.cgi:/localhost/var/www/html/cafe/menu.php", "remoteIP": "10.131.158.224", "host": "10.0.1.102", "request": "/cafe/menu.php", "query": "", "method": "GET", "status": "200", "useragent": "Mozilla/5.0 (Android 12.0.0; Mobile; rv:53.0) Gecko/53.0 Firefox/53.0", "referrer": ""}
{
  "time": "2025-04-25T22:42:18.930Z", "process": "4968", "filename": "proxy.cgi:/localhost/var/www/html/cafe/processOrder.php", "remoteIP": "10.131.158.224", "host": "10.0.1.102", "request": "/cafe/processOrder.php", "query": "", "method": "POST", "status": "200", "useragent": "Mozilla/5.0 (Android 12.0.0; Mobile; rv:53.0) Gecko/53.0 Firefox/53.0", "referrer": "http://10.0.1.102/cafe/menu.php"}
{
  "time": "2025-04-25T22:42:18.930Z", "process": "4937", "filename": "/var/www/html/cafe", "remoteIP": "109.25.247.228", "host": "10.0.1.102", "request": "/cafe", "query": "", "method": "GET", "status": "200", "useragent": "Mozilla/5.0 (Android 12.0.0; Mobile; rv:53.0) Gecko/53.0 Firefox/53.0", "referrer": ""}
{
  "time": "2025-04-25T22:42:18.930Z", "process": "5048", "filename": "/var/www/html/cafe", "remoteIP": "23.162.178.238", "host": "10.0.1.102", "request": "/cafe", "query": "", "method": "GET", "status": "200", "useragent": "Mozilla/5.0 (Windows NT 10.0) AppleWebKit/531.2 (KHTML, like Gecko) Chrome/111.0.553.2.0.854.0", "referrer": ""}
{
  "time": "2025-04-25T22:42:18.930Z", "process": "4929", "filename": "proxy.cgi:/localhost/var/www/html/cafe/menu.php", "remoteIP": "23.162.178.238", "host": "10.0.1.102", "request": "/cafe/menu.php", "query": "", "method": "GET", "status": "200", "useragent": "Mozilla/5.0 (Windows NT 10.0) AppleWebKit/531.2 (KHTML, like Gecko) Chrome/111.0.553.2.0.854.0", "referrer": "http://10.0.1.102/cafe/index.php"}
{
  "time": "2025-04-25T22:42:18.930Z", "process": "5050", "filename": "/var/www/html/cafe", "remoteIP": "66.191.5.62", "host": "10.0.1.102", "request": "/cafe", "query": "", "method": "GET", "status": "200", "useragent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.553.2.0.854.0", "referrer": "http://10.0.1.102/cafe/index.php"} 
```

```
voclabs:~/environment $ cat samplelogs/access_log.log | head -1 | python -m json.tool
{
  "time": "2025-04-25T22:42:18.930Z", "process": "4948", "filename": "/var/www/html/cafe", "remoteIP": "186.230.87.244", "host": "10.0.1.102", "request": "/cafe", "query": "", "method": "GET", "status": "200", "useragent": "Mozilla/5.0 (iPhone; CPU iPhone OS 16_2 like Mac OS X) AppleWebKit/536.1 (KHTML, like Gecko) CriOS/13.0.848.0 Mobile/65R242 Safari/16.2", "referrer": ""} 
```

```
voclabs:~/environment $ cat samplelogs/access_log.log | wc -l
187051 
```

```
voclabs:~/environment $ sudo systemctl stop amazon-cloudwatch-agent
Activate Windows
Go to Settings to activate Windows. 
```

```
CodeWhisperer AWS profile:default
Light rain At night
8:52 PM
4/25/2025 
```

```
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
```

```

    "referrer": "."
}
vocabs:~/environment $ cat samplelogs/access_log.log | wc -l
187051
vocabs:~/environment $ sudo systemctl stop amazon-cloudwatch-agent
vocabs:~/environment $ sudo cp ~/environment/samplelogs/access_log.log /var/log/www/access/access_log
vocabs:~/environment $ sudo systemctl start amazon-cloudwatch-agent
vocabs:~/environment $ 

```

Activate Windows
Go to Settings to activate Windows.

CodeWhisperer AWS profile default

Light rain J Tomorrow

Search

8:53 PM 4/25/2025

We can see more log events

The screenshot shows the AWS CloudWatch Log Events interface. The left sidebar navigation includes 'CloudWatch' (selected), 'Logs' (selected), 'Metrics' (unselected), and 'X-Ray traces' (unselected). Under 'Logs', there are sections for 'Log groups' (New), 'Log Anomalies', 'Live Tail', 'Logs Insights' (New), and 'Contributor Insights'. The main content area is titled 'Log events' and displays a table of log entries. The columns are 'Timestamp' and 'Message'. The table contains approximately 20 log entries from February 24, 2023, at 08:29:12 UTC. Each entry includes a timestamp, process ID, file name, remote IP, host, and other metadata. A search bar at the top allows filtering by search terms. Below the search bar are buttons for 'Clear', time ranges ('1m', '30m', '1h', '12h', 'Custom'), and 'UTC timezone'. Top navigation tabs include 'CloudWatch' (selected), 'Log groups' (selected), 'apache/access' (selected), and 'i-0f5c6d7607928bf40'. The top right corner shows the user's email (vocabs: user3908540:ppalavay@stevens.edu) and the date (4/25/2025).

This screenshot is identical to the one above, showing the AWS CloudWatch Log Events interface. It displays the same log entries for the apache/access log group, with the same sidebar navigation, search bar, and filter options. The top navigation tabs and user information are also identical.

The screenshot shows the AWS CloudWatch Log events interface. The left sidebar includes sections for Favorites and recents, AI Operations (Alarms, Logs, Metrics), and X-Ray traces. The main content area displays a table of log events with columns for Timestamp and Message. The timestamp column shows dates from February 2025. The message column contains JSON log entries. A search bar at the top allows filtering by search terms, phrases, or values. Buttons for Actions, Start tailing, and Create metric filter are available at the top right. The bottom of the screen shows the AWS navigation bar and system status.

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

The screenshot shows the AWS CloudWatch Logs Insights interface. The left sidebar includes sections for Favorites and recents, Logs (Log groups, Log Anomalies, Live Tail, Logs Insights), Metrics, and X-Ray traces. The main content area features a "Logs Insights Info" section with a "Logs Insights QL" input field containing a sample query. Below it is a "Select log groups" section with a dropdown for "Log group name". To the right is a "Discovered fields" panel titled "Learn more" which lists "About field indexes" and provides instructions for indexing log groups. It also includes a "Create field indexes" button and a "Filter by field name or field counts" search bar. The bottom of the screen shows the AWS navigation bar and system status.

Click Save query

Name: menu-visitors

Folder: non-geo-results

The screenshot shows the 'Save a new query' page in the AWS CloudWatch Logs Insights interface. The 'Query name' field is set to 'menu-visitors'. The 'Folder - optional' dropdown shows 'non-geo-results' with a 'Create new' button. The 'Query definition details' section includes 'Logs Insights QL' as the language. Under 'Log groups - optional', 'Select log groups by' is set to 'Log group name' with 'apache/access' selected. The 'Selection criteria' dropdown shows 'Select up to 50 log groups' and 'Browse log groups'. The 'Query text' section contains the following Log Insights QL query:

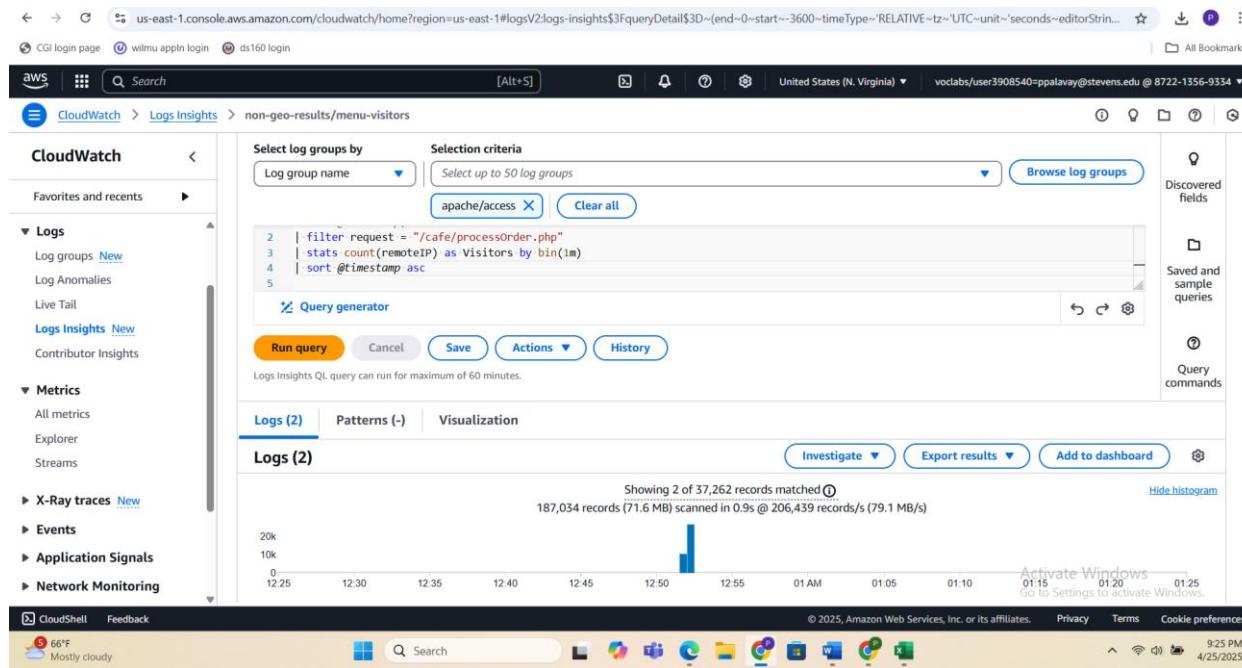
```
fields @timestamp, remotelP  
| filter request = "/cafe/menu.php"  
| stats count(remotelP) as menuVisitors
```

The screenshot shows the results page for the 'non-geo-results/menu-visitors' query. A green success message at the top states 'non-geo-results/menu-visitors has been created successfully.' The interface includes a 'Query generator' section with 'Run query', 'Save', 'Actions', and 'History' buttons. Below this is a 'Logs (1)' section showing a histogram of records over time. The histogram shows a single peak at approximately 12:50. The log entry details show a single record with the timestamp '2025-04-25T12:50:00Z' and the value 'menuVisitors' with a count of 1 and ID 49778.

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:

$$(37262 / 49778) * 100 \approx 74.85\%$$

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

```

1 Menu visitors:49778
2 Purchase visitors:37262
3 Visitors Who Didn't Purchase:49778 - 37262 = 12516
4 Conversion Rate:(37262 / 49778) * 100 ≈ 74.85%
5

```

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

```

bash -> head -1 /var/log/www/access/access_log | python -m json.tool
{
    "time": "2023-02-22 03:38:18",
    "process": "5112",
    "filename": "/var/www/html/cafe",
    "remoteIP": "91.192.109.163",
    "host": "10.0.1.102",
    "request": "/cafe",
    "query": "",
    "method": "GET",
    "status": "200",
    "useragent": "Mozilla/5.0 (Windows NT 10.0) AppleWebKit/531.2 (KHTML, like Gecko) Chrome/111.0.553.2.0.854.0",
    "refer": "",
    "country": "ES",
    "region": "Andalusia",
    "city": "La Rinconada"
}

```

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

```

time: "2023-02-22 03:38:18",
process: "5112",
filename: "/var/www/html/cafe",
remoteIP: "91.192.109.163",
host: "10.0.1.102",
request: "/cafe",
query: "",
method: "GET",
status: "200",
useragent: "Mozilla/5.0 (Windows NT 10.0) AppleWebKit/531.2 (KHTML, like Gecko) Chrome/111.0.553.2.0.854.0",
refer: "-",
country: "ES",
region: "Andalusia",
city: "La Rinconada",
lat: "37.4867198",
lon: "-5.981868"
}

```

Activate Windows
Go to Settings to activate Windows.

Delete the Old Log Group

Action	Description
Delete log group(s)	Deletes the selected log groups.
Edit retention setting(s)	Changes the retention period for the log group.
Create metric filter	Creates a metric filter for the log group.
Create contributor insights rules	Creates contributor insights rules for the log group.
Create data protection policy	Creates a data protection policy for the log group.
Anomaly detection	Enables anomaly detection for the log group.
Subscription filters	Creates subscription filters for the log group.
Export data to Amazon S3	Exports data from the log group to Amazon S3.
View all exports to Amazon S3	Views all exports to Amazon S3 for the log group.

Activate Windows
Go to Settings to activate Windows.

Log Group(apache/access) deleted successfully

The screenshot shows the AWS CloudWatch Log groups page. On the left, there's a navigation sidebar with sections like AI Operations, Alarms, Logs (Log groups, Log Anomalies, Live Tail, Logs Insights, Contributor Insights), Metrics, and X-Ray traces. The main content area has a green banner at the top stating: "The following log group(s) have been deleted: apache/access". Below this, a table lists one log group: "apache/error" (Standard, Configure, 6 months). A search bar and filter options are also present.

Restart the CloudWatch Agent

```
sudo systemctl start amazon-cloudwatch-agent
```

we can see apache/access log group again.

This screenshot shows the same AWS CloudWatch Log groups page after the restart. The green banner at the top is no longer present. The table now shows two log groups: "apache/access" and "apache/error", both listed under Standard log class and 6 months retention period. The rest of the interface remains the same, including the sidebar and the bottom navigation bar.

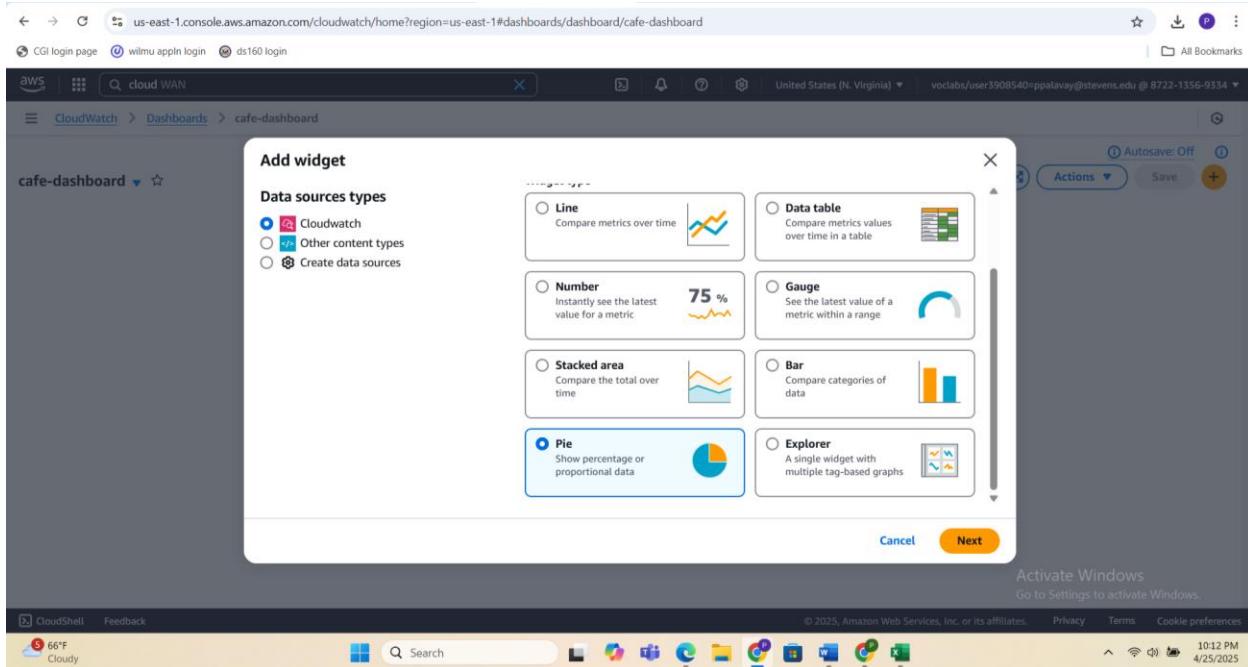
TASK2: Create a Dashboard

The screenshot shows the AWS CloudWatch Dashboards interface. On the left, there's a sidebar with various monitoring services: AI Operations, Alarms, Logs, Metrics, and X-Ray traces. The 'Logs' section is currently selected. The main content area is titled 'Custom dashboards' and shows a message 'No dashboards'. It includes a search bar, a 'Create dashboard' button, and a link to 'Read more about Dashboards'. At the bottom right, there's a weather widget showing '66°F Cloudy'.

Name the dashboard as café-dashboard

The screenshot shows the 'Create new dashboard' dialog box overlaid on the CloudWatch Dashboards page. The dialog has a title 'Create new dashboard' and a 'Dashboard name' input field containing 'café-dashboard'. Below the input field is a note about valid characters. At the bottom are 'Cancel' and 'Create dashboard' buttons.

We should add widgets



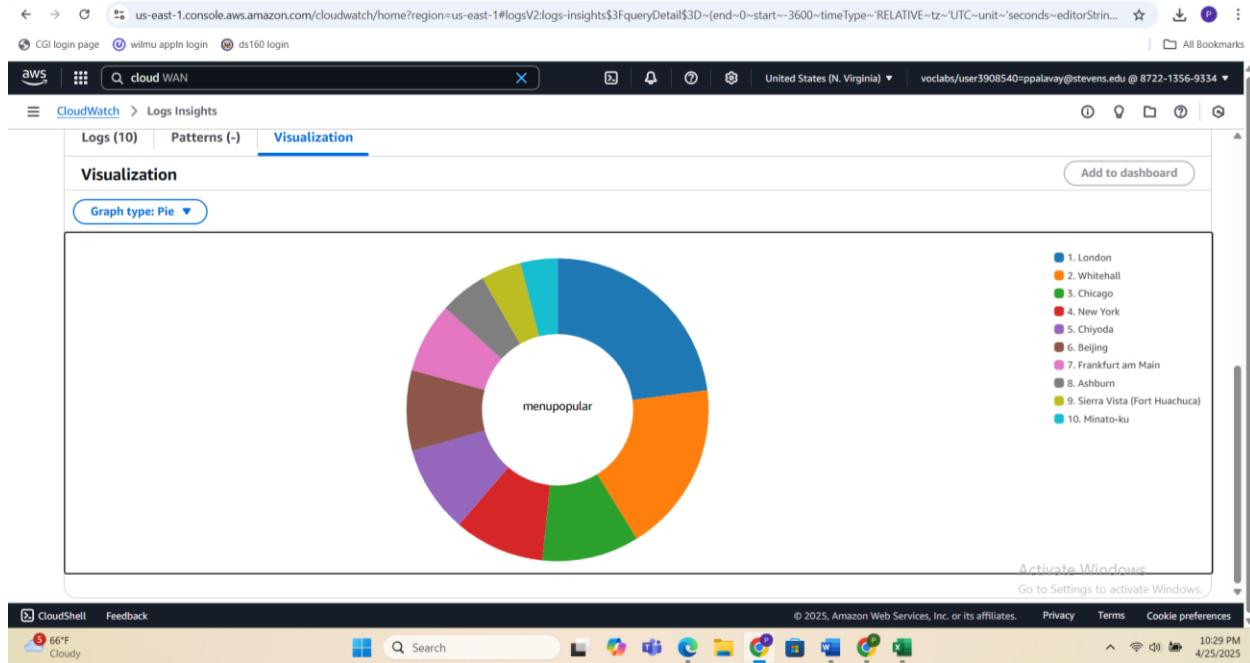
Click Add widget

Data type:logs

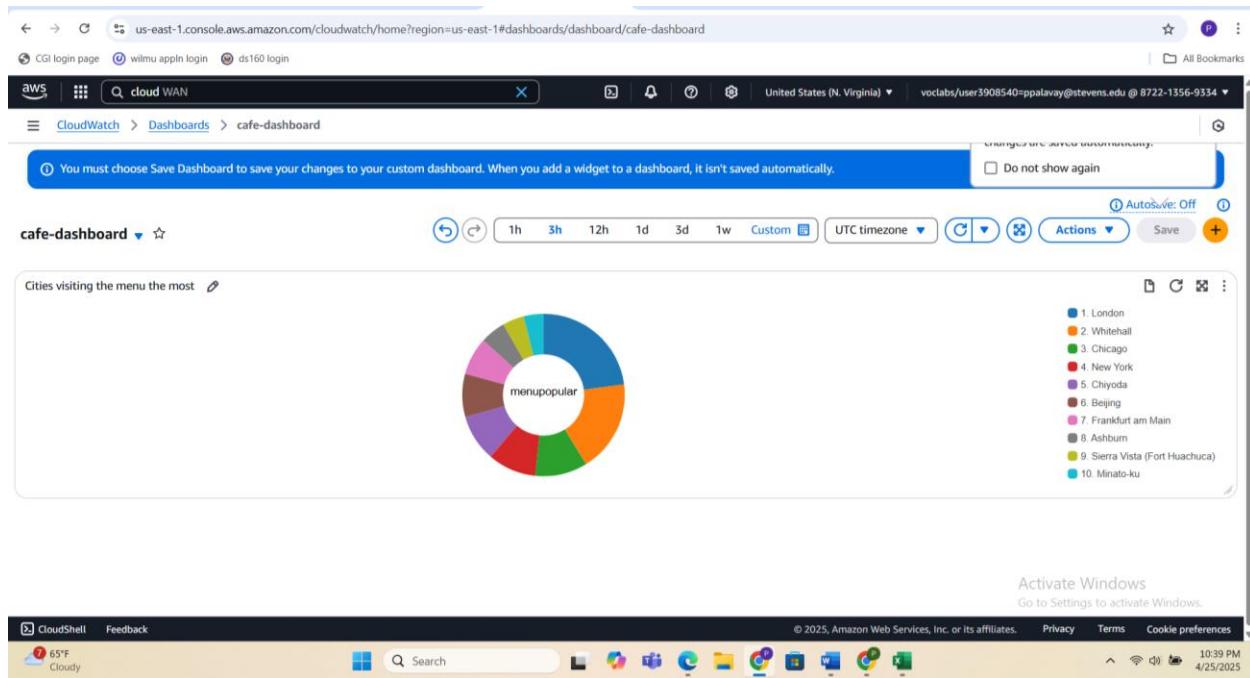
Select pie chart

Choose apache/access as your log group

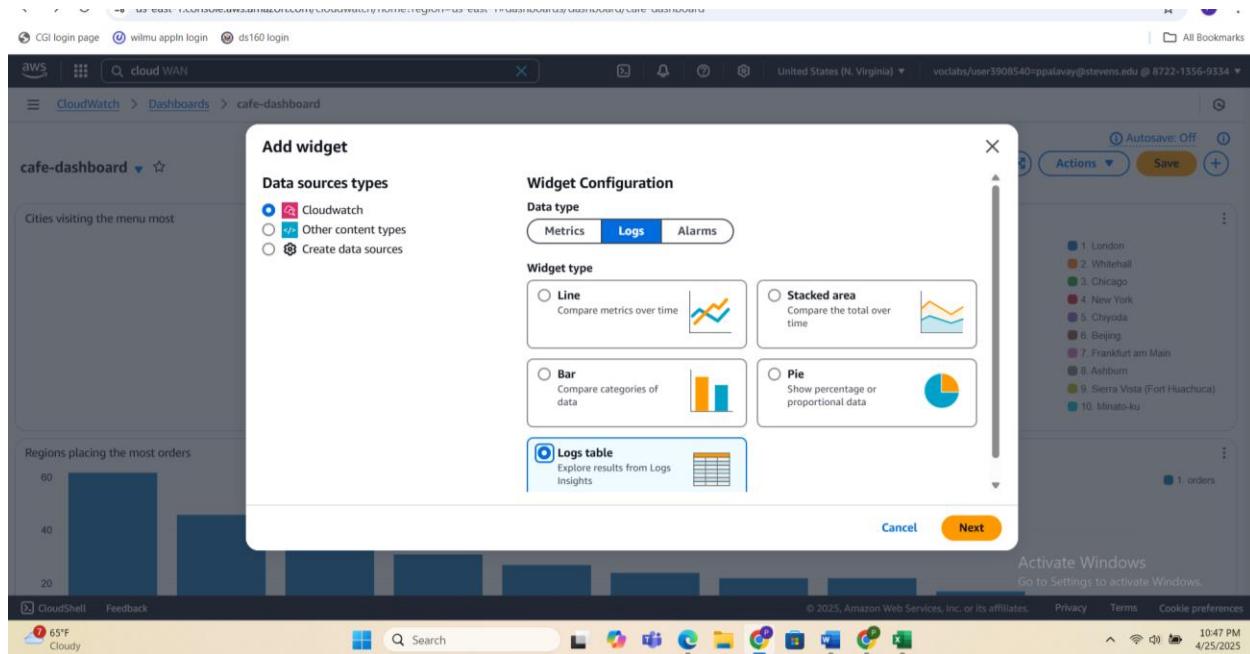
Run the query



Rename the widget as cities visiting the menu most



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

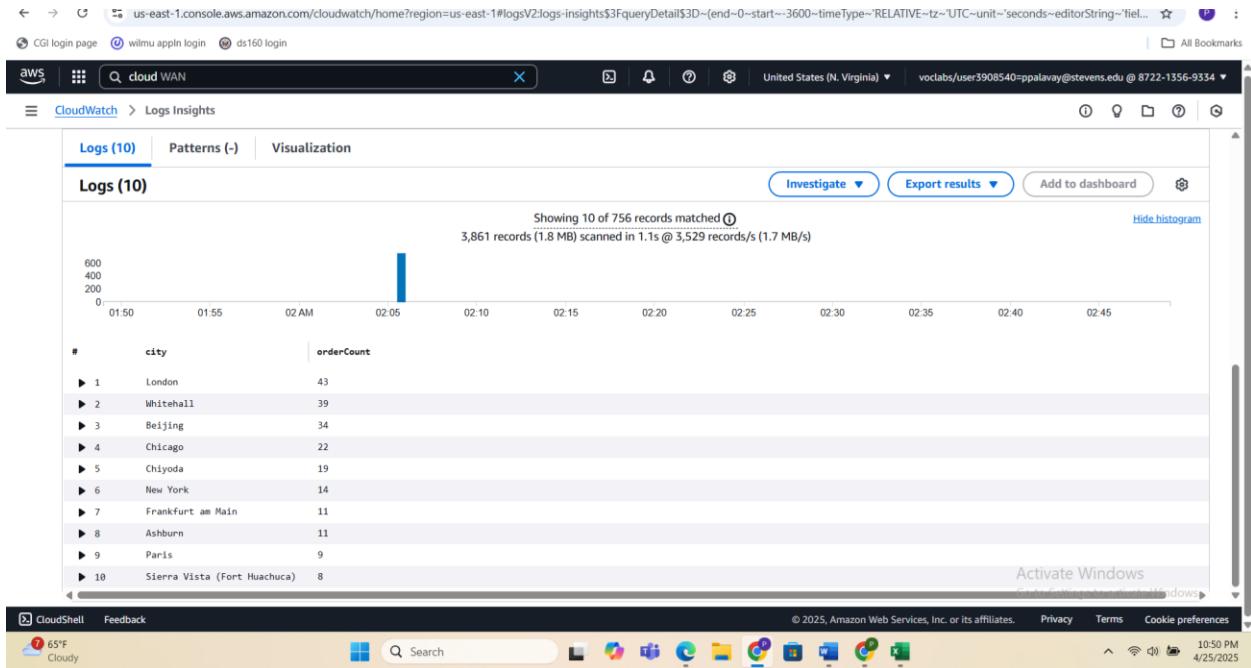


Run the query

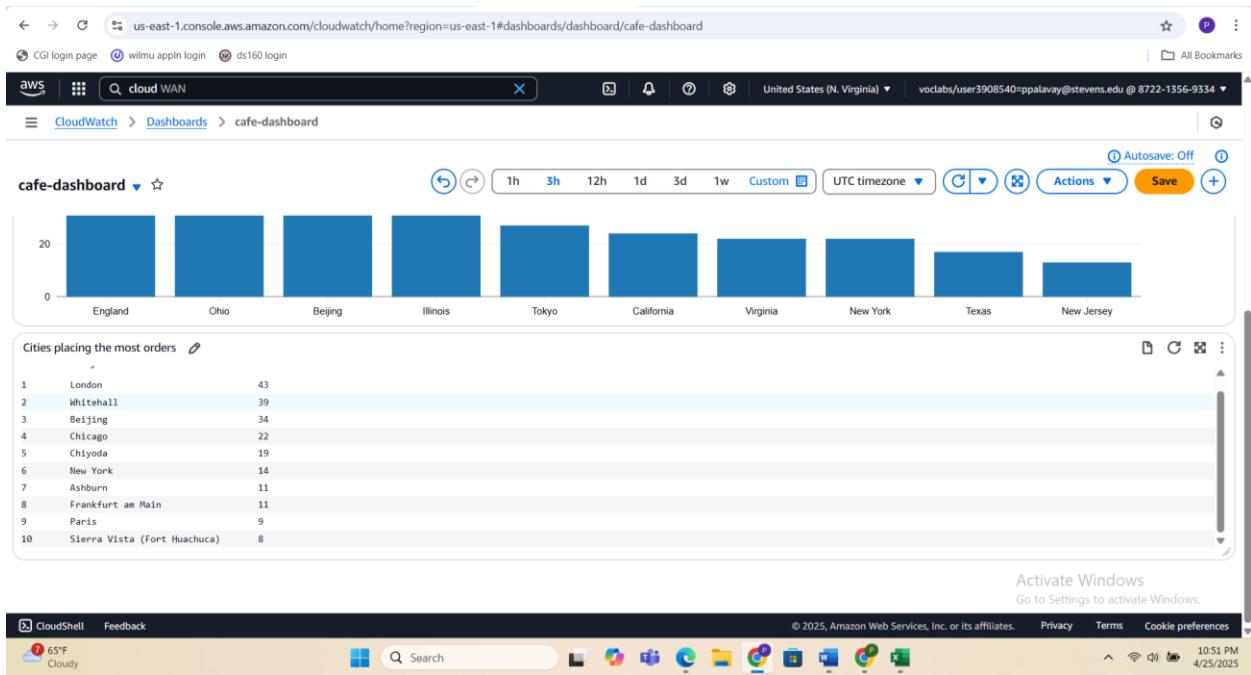
```

1 fields remoteIP, city
2 | filter request = "/cafe/processOrder.php"
3 | stats count() as orderCount by city
4 | sort orderCount desc
    
```

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

us-east-1.console.aws.amazon.com/cloudwatch/home?region=us-east-1#dashboards/dashboard/cafe-dashboard

CloudWatch > Dashboards > cafe-dashboard

Add widget

Data sources types

- Cloudwatch
- Other content types
- Create data sources

Widget Configuration

Data type

- Metrics
- Logs
- Alarms

Widget type

- Line
- Stacked area
- Bar
- Pie
- Logs table

Regions placing the most orders

Region	Orders
England	60
Ohio	45

Cities placing the most orders

City	Orders
London	39
Whitehall	34
Beijing	27
Chicago	27

Logs Insights

Logs Insights QL

```
1 fields remoteIP, region
2 | filter request = "/cafe"
3 | stats count() as mainpage_by_region
4 | sort mainpage desc
```

Selection criteria

Log group name: apache/access

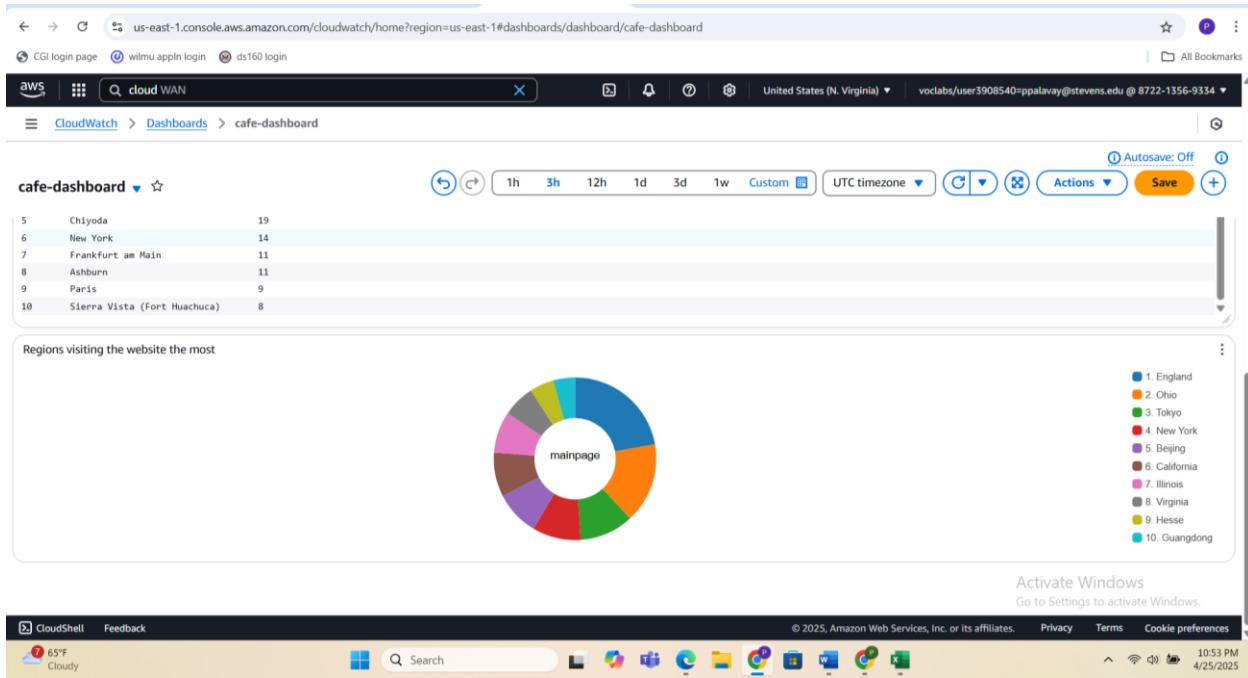
Logs (10) Patterns (-) Visualization

Graph type: Pie

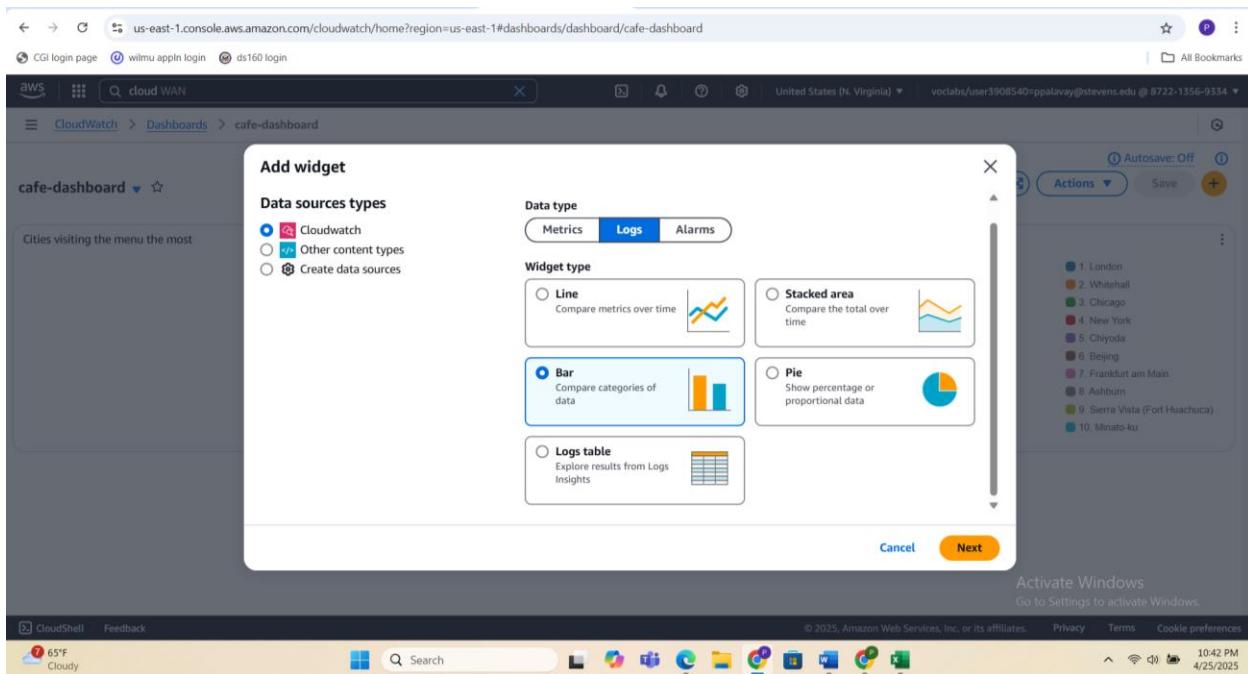
Activate Windows

Go to Settings to activate Windows.

Rename the widget as Regions visiting the website most



Fourth Widget :widget type-Bar graph



The screenshot shows the AWS CloudWatch Logs Insights interface. At the top, there are tabs for 'Logs Insights QL', 'OpenSearch PPL - new', and 'OpenSearch SQL - new'. Below this, there are time range controls (5m, 30m, 1h, 3h, 12h, Custom) and a 'Compare (Off)' button. The main area displays a query editor with the following code:

```

3 | stats count() as orders by region
4 | sort orders desc
5 | limit 10
6

```

Below the code, there are buttons for 'Run query', 'Cancel', 'Save', and 'History'. A message indicates 'Completed. Query executed for 1 log groups.' The visualization tab is selected, showing a bar chart with a single data point at index 60. To the right, there are sections for 'Discovered fields', 'Saved and sample queries', and 'Query commands'. The bottom of the screen shows the Windows taskbar with various pinned icons.

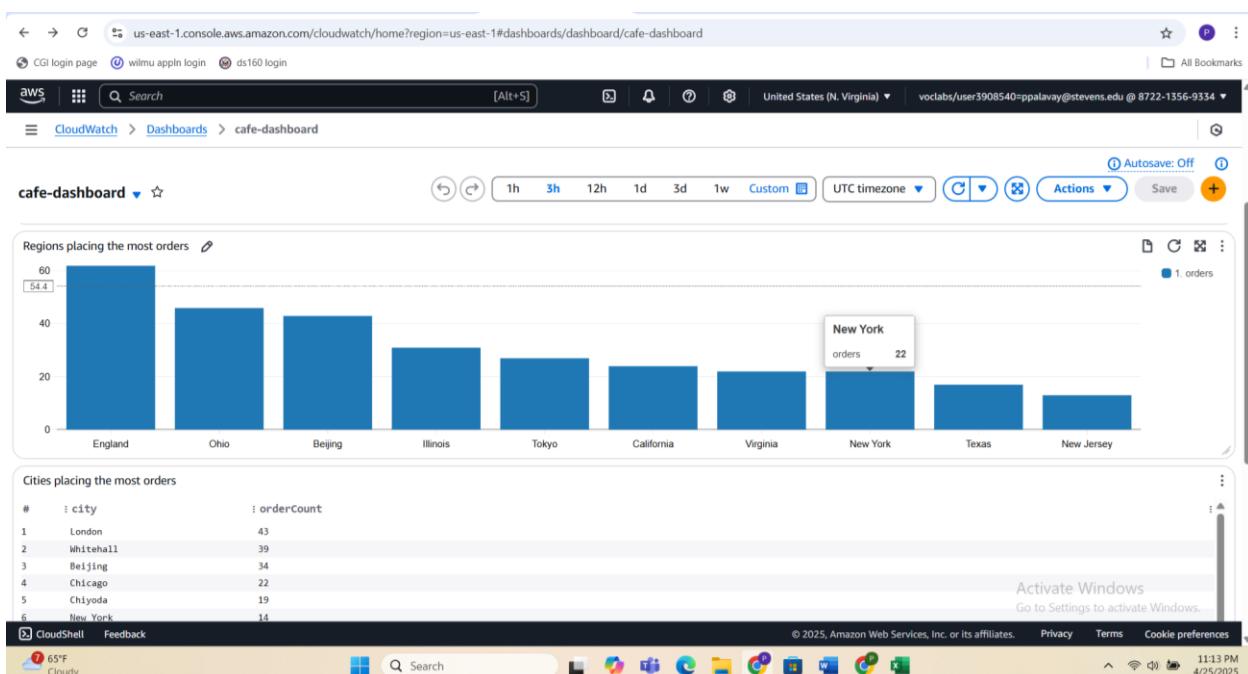
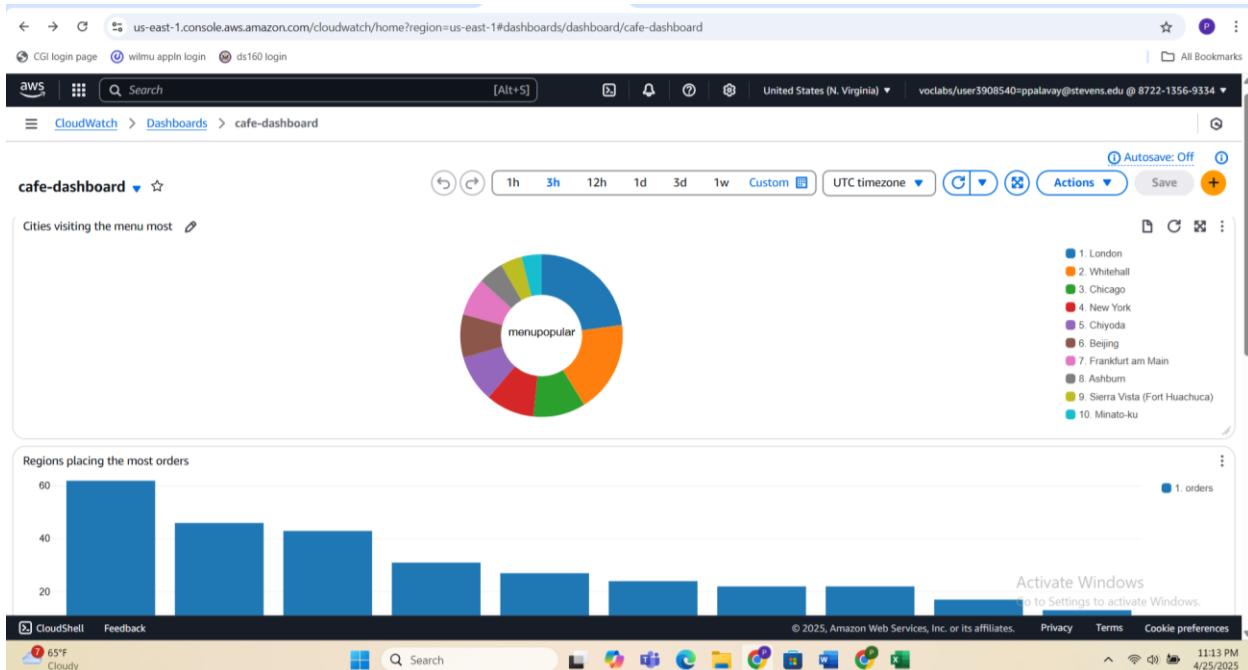
Rename this as Regions placing the most orders

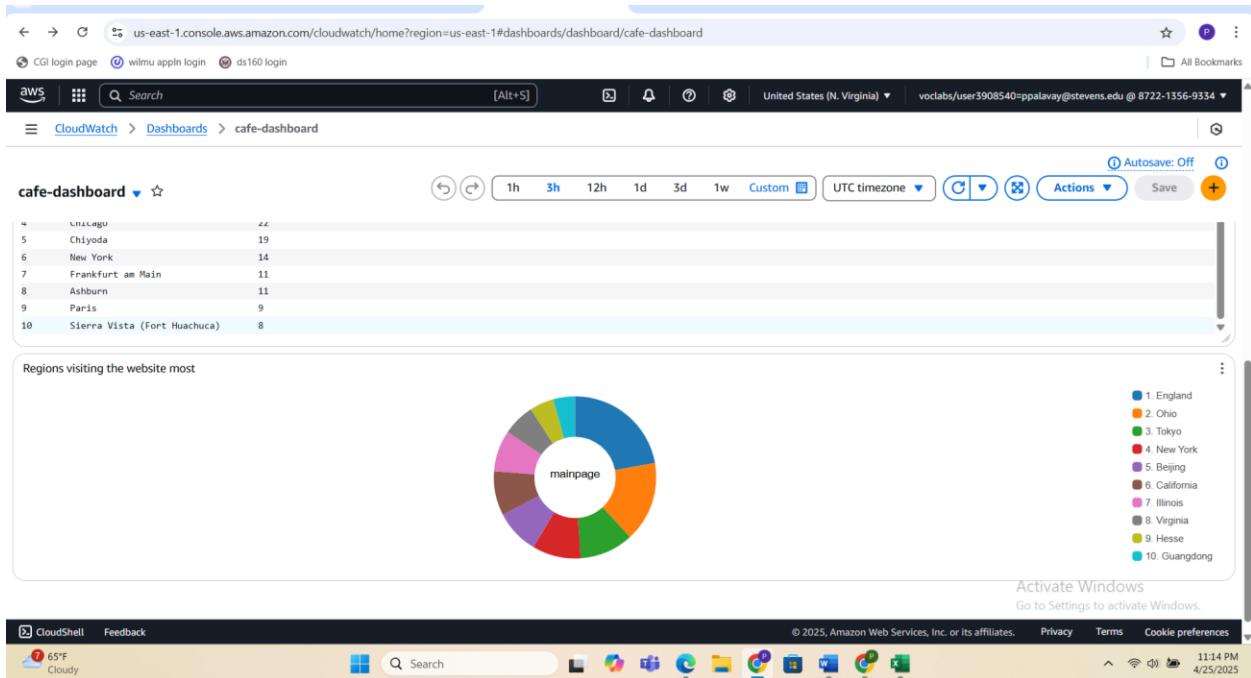
The screenshot shows the AWS CloudWatch Dashboards interface. The dashboard is titled 'cafe-dashboard'. It features a donut chart titled 'menupopular' and a bar chart titled 'Regions placing the most orders'. The bar chart data is as follows:

Region	orders
England	60
Ohio	45
Beijing	42
Illinois	32
Tokyo	28
California	25
Virginia	23
New York	22
Texas	18
New Jersey	15

The bottom of the screen shows the Windows taskbar with various pinned icons.

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





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/
```

```
voclabs:~/environment $ aws s3 cp /var/log/www/access/access_log s3://accap4-logsbucket-16c70f40/
upload: ..../var/log/www/access/access_log to s3://accap4-logsbucket-16c70f40/access_log
voclabs:~/environment $ |
```

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

The screenshot shows the AWS CloudWatch Logs Insights interface. On the left, there's a sidebar with navigation links for CloudWatch, Metrics, X-Ray traces, Events, and Application Signals. The main area has a search bar at the top. Below it, there's a section for "Select log groups by" and "Selection criteria". A code editor window displays a valid Log Insights query:

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

Below the query editor is a "Query generator" section with "Run query", "Save", and "History" buttons. The results section is titled "Logs (3.9k)" and shows a histogram with 3,858 records matched. The table below lists the first 15 records:

#	@timestamp	remoteIP	city	request
1	2025-04-26T02:05:57.739Z	63.188.199.36	London	/cafe/menu.php
2	2025-04-26T02:05:53.606Z	49.167.135.112	Yongsan-dong	/cafe/menu.php
3	2025-04-26T02:05:53.606Z	74.2.195.235	Miami	/cafe
4	2025-04-26T02:05:53.606Z	17.48.1.9	Cupertino	/cafe
5	2025-04-26T02:05:53.606Z	179.52.252.95	Salvadón d...	/cafe
6	2025-04-26T02:05:53.606Z	80.70.188.65	Paderborn	/cafe
7	2025-04-26T02:05:53.606Z	80.70.188.65	Paderborn	/cafe/menu.php
8	2025-04-26T02:05:53.606Z	118.33.194.38	Gapeyong Co...	/cafe
9	2025-04-26T02:05:53.606Z	68.33.54.62	Prince Fred...	/cafe
10	2025-04-26T02:05:53.606Z	68.33.54.62	Prince Fred...	/cafe/menu.php
11	2025-04-26T02:05:53.606Z	118.255.7.184	Beijing	/cafe
12	2025-04-26T02:05:53.606Z	118.255.7.184	Beijing	/cafe/menu.php
13	2025-04-26T02:05:53.606Z	189.112.144.59	Überländia	/cafe
14	2025-04-26T02:05:53.606Z	7.48.172.78	Whitehall	/cafe
15	2025-04-26T02:05:53.606Z	7.48.172.78	Whitehall	/cafe/menu.php

This screenshot shows the same AWS CloudWatch Logs Insights interface as the previous one, but with a different set of log records displayed. The results section shows a histogram with 3,879 records matched. The table below lists the first 15 records:

#	@timestamp	remoteIP	city	request
1	2025-04-26T02:05:57.739Z	63.188.199.36	London	/cafe/menu.php
2	2025-04-26T02:05:53.606Z	49.167.135.112	Yongsan-dong	/cafe/menu.php
3	2025-04-26T02:05:53.606Z	74.2.195.235	Miami	/cafe
4	2025-04-26T02:05:53.606Z	17.48.1.9	Cupertino	/cafe
5	2025-04-26T02:05:53.606Z	179.52.252.95	Salvadón d...	/cafe
6	2025-04-26T02:05:53.606Z	80.70.188.65	Paderborn	/cafe
7	2025-04-26T02:05:53.606Z	80.70.188.65	Paderborn	/cafe/menu.php
8	2025-04-26T02:05:53.606Z	118.33.194.38	Gapeyong Co...	/cafe
9	2025-04-26T02:05:53.606Z	68.33.54.62	Prince Fred...	/cafe
10	2025-04-26T02:05:53.606Z	68.33.54.62	Prince Fred...	/cafe/menu.php
11	2025-04-26T02:05:53.606Z	118.255.7.184	Beijing	/cafe
12	2025-04-26T02:05:53.606Z	118.255.7.184	Beijing	/cafe/menu.php
13	2025-04-26T02:05:53.606Z	189.112.144.59	Überländia	/cafe
14	2025-04-26T02:05:53.606Z	7.48.172.78	Whitehall	/cafe
15	2025-04-26T02:05:53.606Z	7.48.172.78	Whitehall	/cafe/menu.php

This query gives the count of the valid logs

fields @timestamp, remoteIP, request, city

| filter ispresent(remoteIP) and ispresent(request) and ispresent(city)

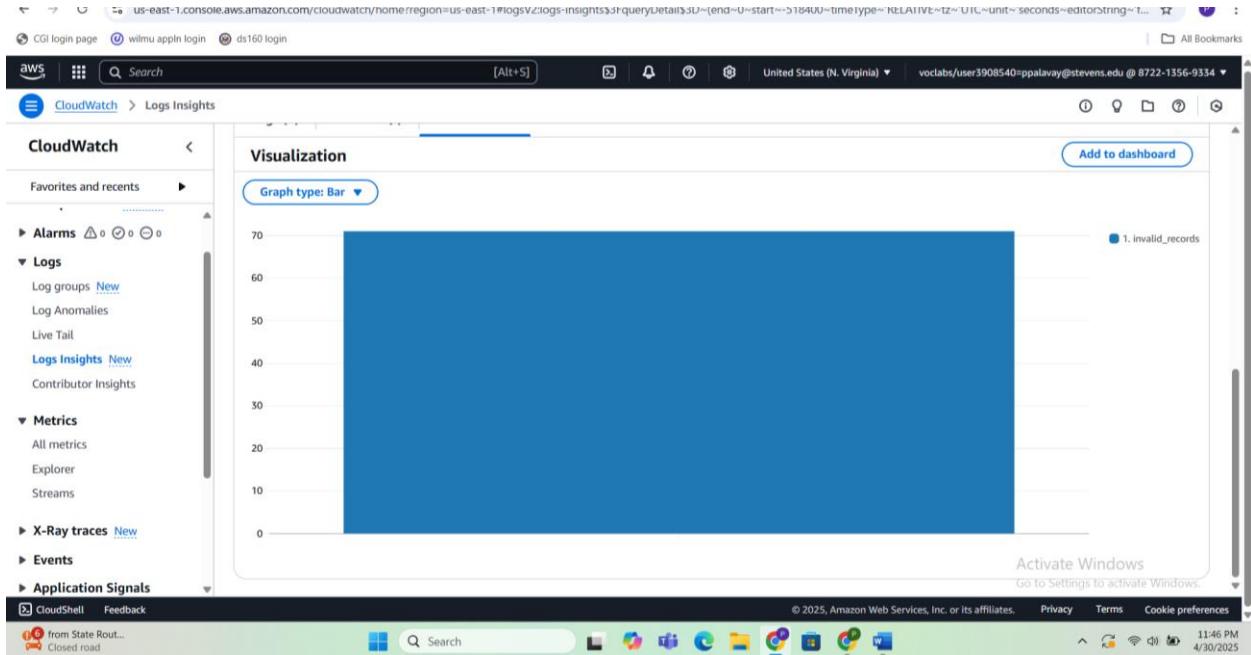
| stats count() as validRecords .

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