

# VPC Monitoring with Flow Logs

N

Nchindo Boris

The screenshot shows the AWS CloudWatch Logs Insights interface. The left sidebar navigation includes CloudWatch, Favorites and recent, Dashboards, AI Operations, Alarms, Logs (selected), Log groups, Log Anomalies, Live Tail, Logs Insights (selected), Contributor Insights, Metrics, Application Signals (APM), Network Monitoring, and Insights. The main content area has tabs for Logs (10), Patterns (-), and Visualization. The Logs (10) tab is active, displaying a histogram of bytes transferred over time (14:20 to 15:20) and a list of 10 log entries. The visualization tab shows a histogram with a peak around 14:30. The patterns tab has options for Summarize results, Investigate, Export results, and Add to dashboard.

#	srcAddr	dstAddr	bytesTransferred
1	10.1.0.144	10.1.1.214	158888
2	10.2.1.214	10.1.0.144	158888
3	18.143.37.195	10.1.0.144	11592
4	18.136.234.195	10.1.0.144	10136
5	13.218.114.184	10.1.0.144	9352
6	16.171.83.98	10.1.0.144	7988
7	16.16.109.128	10.1.0.144	7088
8	57.188.218.113	10.1.0.144	6168
9	44.218.233.115	10.1.0.144	6104
10	3.114.113.149	10.1.0.144	5768



# Introducing Today's Project!

## What is Amazon VPC?

Amazon VPC is a networking service that lets you create a logically isolated section of the AWS cloud.

## How I used Amazon VPC in this project

I used Amazon VPC in this project to monitor flow logs with CloudWatch

## One thing I didn't expect in this project was...

I didn't expect in this project to be easy and interesting as it was

## This project took me...

It took me 1 hour to complete this project

# In the first part of my project...

## Step 1 - Set up VPCs

In this step I will create two VPCs from scratch!

## Step 2 - Launch EC2 instances

In this step I will Launch an EC2 instance in each VPC, so we can use them to test your VPC peering connection later.

## Step 3 - Set up Logs

In this step I will;

- Set up a way to track all inbound and outbound network traffic.
- Set up a space that stores all of these records.

## Step 4 - Set IAM permissions for Logs

In this step I will;

- Give VPC Flow Logs the permission to write logs and send them to CloudWatch.
- Finish setting up your subnet's flow log.

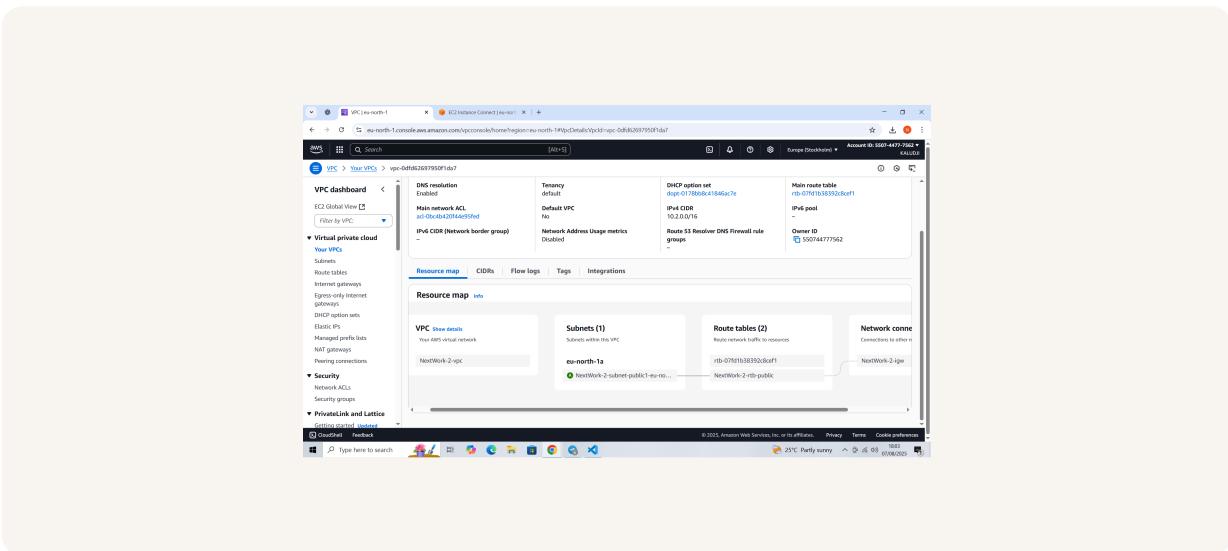
# Multi-VPC Architecture

I started my project by launching 2 VPCs

The CIDR blocks for VPCs 1 and 2 are 10.2.0.0/16 and 10.1.0.0/16. They have to be unique so the IP addresses of their resources don't overlap.

## I also launched EC2 instances in each subnet

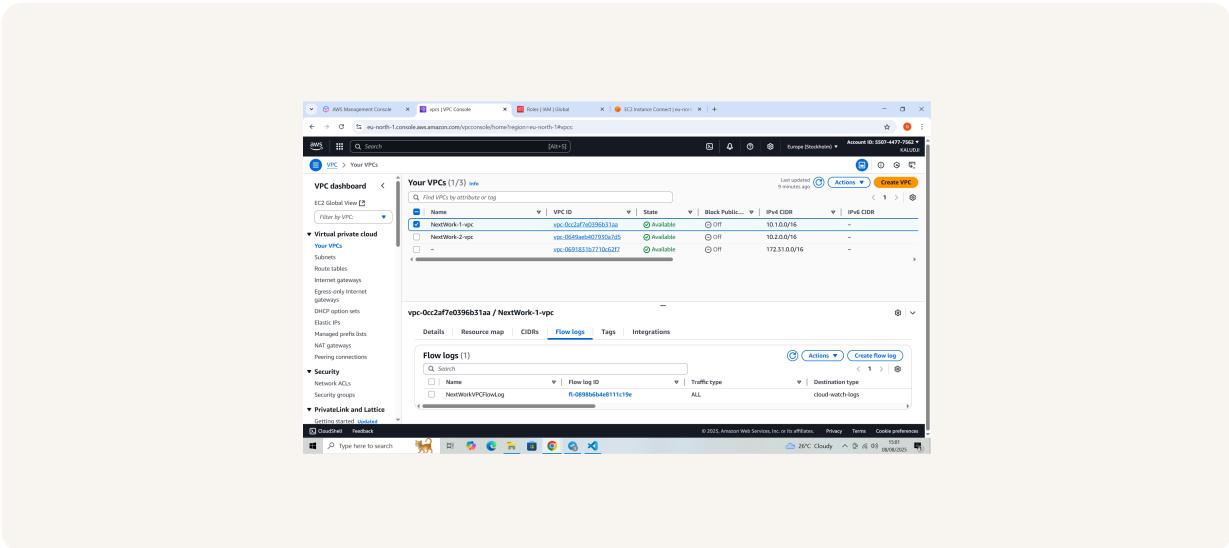
My EC2 instances' security groups allow ICMP traffic from ALL IP addresses.



# Logs

Logs are like a diary for your computer systems. They record everything that happens, from users logging in to errors popping up.

Log group is like a big folder in AWS where you keep related logs together.

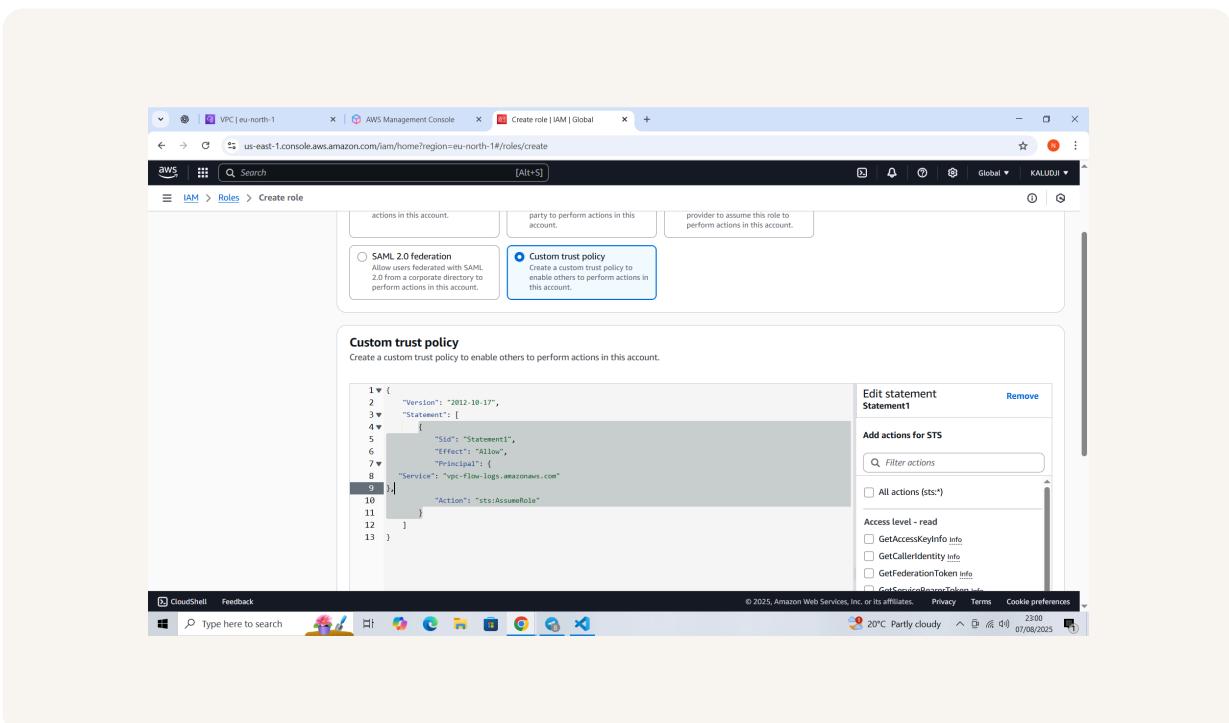


# IAM Policy and Roles

I created an IAM policy to make sure that your VPC can now send log data to your log group!

I also created an IAM role because I want to Assign the IAM role to services

A custom trust policy is used to very narrowly define who can use a role.



# In the second part of my project...

## Step 5 - Ping testing and troubleshooting

In this step, I am going to Get Instance 1 to send test messages to Instance 2.

## Step 6 - Set up a peering connection

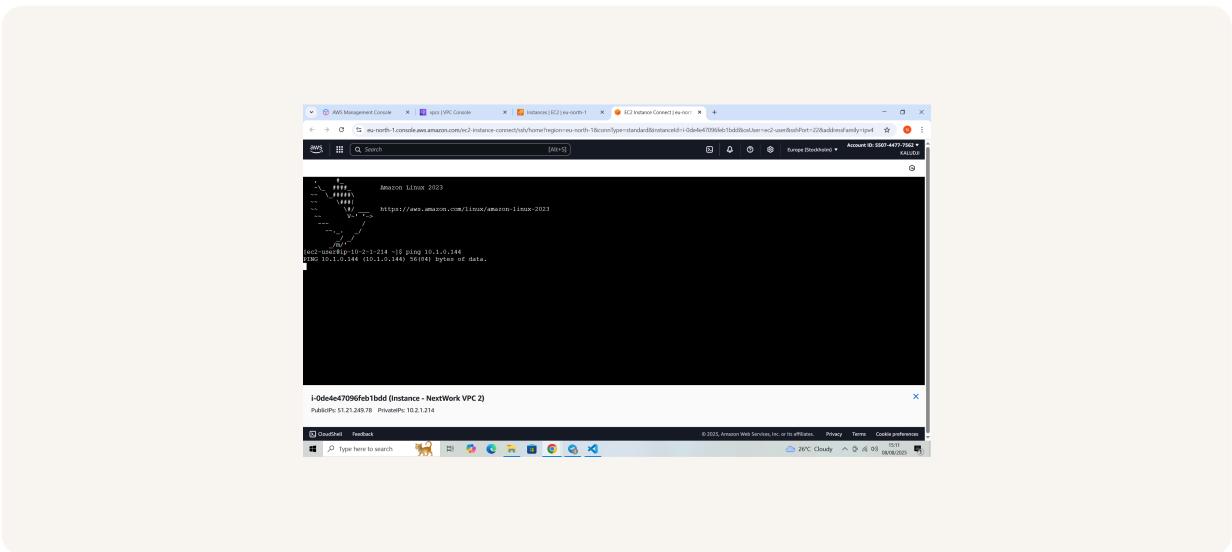
In this step, I am going to Set up a connection link between your VPCs.

## Step 7 - Analyze flow logs

In this step, I am going to:  
- Review the flow logs recorded about VPC 1's public subnet.  
- Analyse the flow logs to get some tasty insights

# Connectivity troubleshooting

My first ping test between my EC2 instances had no replies, which means there's a problem with the connection.



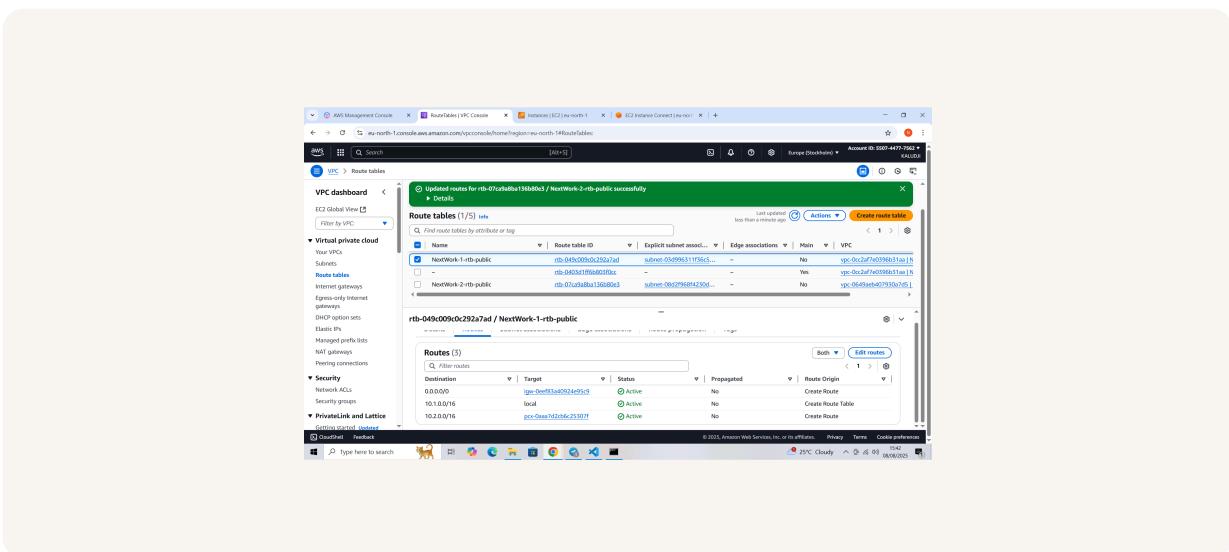
I receive ping replies after the ping test using the other instance's public IP address, which means Instance 2 is correctly configured to respond to ping requests, and Instance 1 can actually communicate with Instance 2 on the public internet

# Connectivity troubleshooting

Looking at VPC 1's route table, I identified that the ping test with Instance 2's private address failed because a VPC peering connection that directly connects VPCs 1 and 2 is lacking. I should set up a new route that directs traffic to our peering

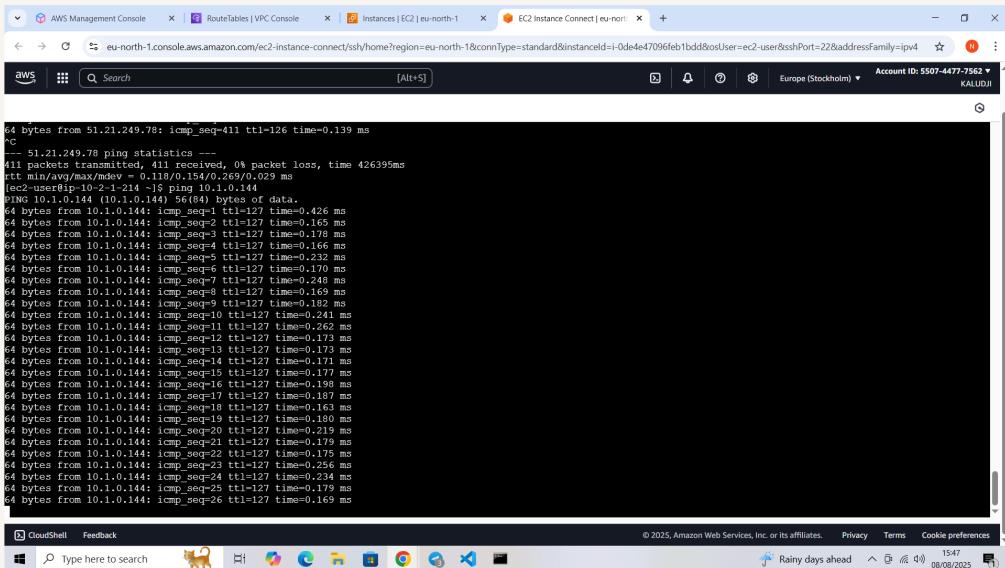
To solve this, I set up a peering connection between my VPCs

I also updated both VPCs' route tables because traffic in VPC 1 won't know how to get to resources in VPC 2 without a route in my route table. I need to set up a route that directs traffic bound for VPC 2 to the peering connection



# Connectivity troubleshooting

I received ping replies from Instance 2's private IP address! This means successfully resolved the connectivity issue by setting up a peering architecture between VPC 1 and VPC 2!. The EC2 instances can now communicate using their private IP addresse



A screenshot of a Windows desktop showing the AWS Management Console in a browser window. The terminal window displays the output of a 'ping' command. The output shows 411 packets transmitted, 411 received, 0% packet loss, and a round-trip time (RTT) minimum/average/max/stddev of 0.110/0.154/0.269/0.029 ms. The destination IP is 10.1.0.144. The terminal window has a black background with white text. The AWS logo and account information are visible at the top of the browser window.

```
64 bytes from 51.21.249.78: icmp_seq=411 ttl=126 time=0.139 ms
PING 10.1.0.144 (10.1.0.144) 56(84) bytes of data.
411 packets transmitted, 411 received, 0% packet loss, time 426395ms
rtt min/avg/max/stddev = 0.110/0.154/0.269/0.029 ms
[ec2-user@ip-10-2-1-214 ~]$ ping 10.1.0.144
PING 10.1.0.144 (10.1.0.144) 56(84) bytes of data.
411 packets transmitted, 411 received, 0% packet loss, time 426395ms
rtt min/avg/max/stddev = 0.110/0.154/0.269/0.029 ms
64 bytes from 10.1.0.144: icmp_seq=1 ttl=127 time=0.425 ms
64 bytes from 10.1.0.144: icmp_seq=2 ttl=127 time=0.165 ms
64 bytes from 10.1.0.144: icmp_seq=3 ttl=127 time=0.178 ms
64 bytes from 10.1.0.144: icmp_seq=4 ttl=127 time=0.166 ms
64 bytes from 10.1.0.144: icmp_seq=5 ttl=127 time=0.232 ms
64 bytes from 10.1.0.144: icmp_seq=6 ttl=127 time=0.170 ms
64 bytes from 10.1.0.144: icmp_seq=7 ttl=127 time=0.170 ms
64 bytes from 10.1.0.144: icmp_seq=8 ttl=127 time=0.165 ms
64 bytes from 10.1.0.144: icmp_seq=9 ttl=127 time=0.182 ms
64 bytes from 10.1.0.144: icmp_seq=10 ttl=127 time=0.241 ms
64 bytes from 10.1.0.144: icmp_seq=11 ttl=127 time=0.262 ms
64 bytes from 10.1.0.144: icmp_seq=12 ttl=127 time=0.173 ms
64 bytes from 10.1.0.144: icmp_seq=13 ttl=127 time=0.173 ms
64 bytes from 10.1.0.144: icmp_seq=14 ttl=127 time=0.171 ms
64 bytes from 10.1.0.144: icmp_seq=15 ttl=127 time=0.177 ms
64 bytes from 10.1.0.144: icmp_seq=16 ttl=127 time=0.198 ms
64 bytes from 10.1.0.144: icmp_seq=17 ttl=127 time=0.187 ms
64 bytes from 10.1.0.144: icmp_seq=18 ttl=127 time=0.163 ms
64 bytes from 10.1.0.144: icmp_seq=19 ttl=127 time=0.170 ms
64 bytes from 10.1.0.144: icmp_seq=20 ttl=127 time=0.219 ms
64 bytes from 10.1.0.144: icmp_seq=21 ttl=127 time=0.179 ms
64 bytes from 10.1.0.144: icmp_seq=22 ttl=127 time=0.175 ms
64 bytes from 10.1.0.144: icmp_seq=23 ttl=127 time=0.256 ms
64 bytes from 10.1.0.144: icmp_seq=24 ttl=127 time=0.234 ms
64 bytes from 10.1.0.144: icmp_seq=25 ttl=127 time=0.179 ms
64 bytes from 10.1.0.144: icmp_seq=26 ttl=127 time=0.169 ms
```



# Analyzing flow logs

Flow logs tell us about - Flow log version - Account ID - ID of the ENI - Source & Destination IP address - Source & Destination port number - Protocol number - Number of packets & bytes transferred - Start and End time of the flow - ACCEPT or REJECT

For example, the flow log I've captured tells us;

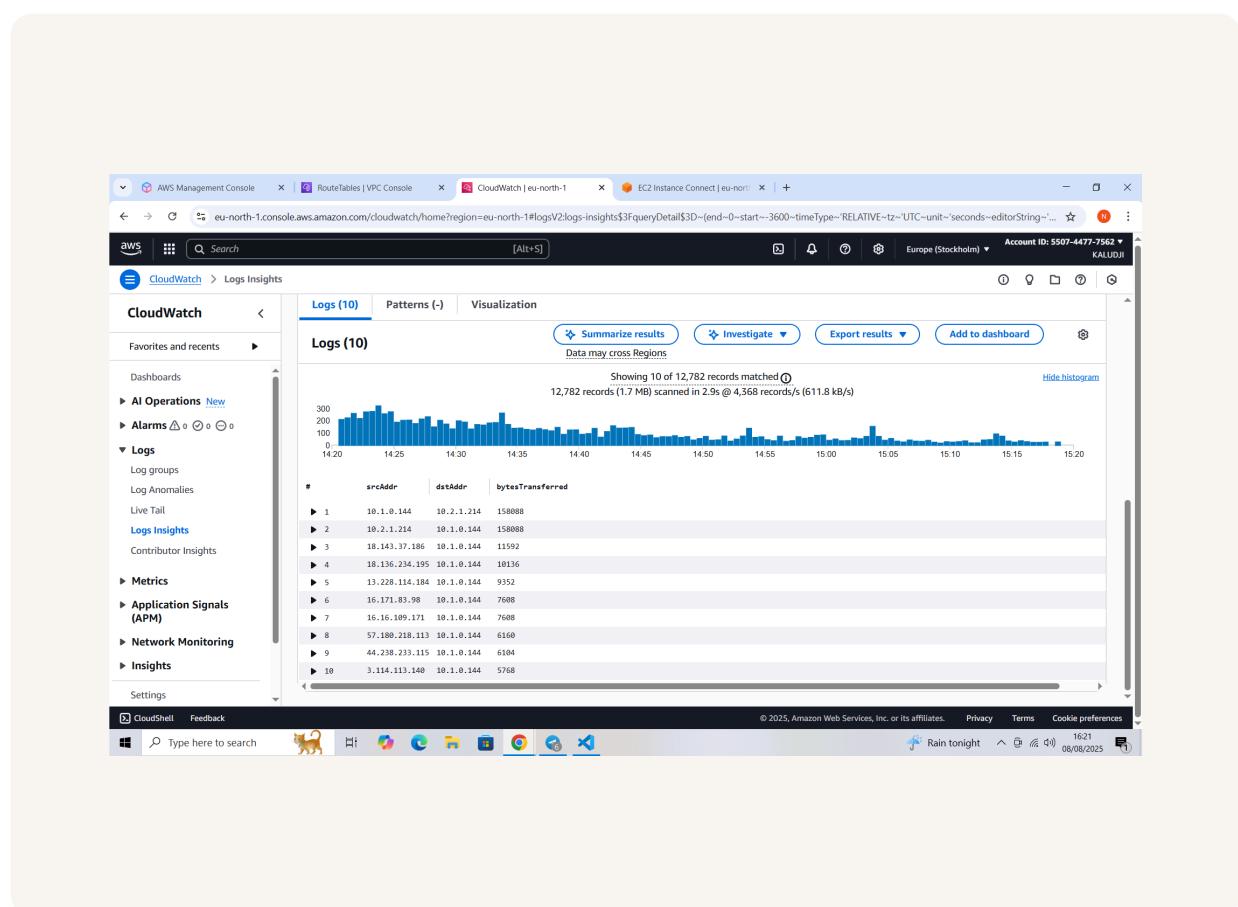
```
2 550744777562 eni-0ce974ff0935ef106 54.168.120.175 10.1.0.144 9993 61465 17 1 56 1754665392
1754665424 REJECT OK
```

The screenshot shows the AWS CloudWatch Log Events interface. The left sidebar navigation includes 'CloudWatch' (selected), 'Logs' (selected), 'Log groups', 'Log Stream names', 'Log Analytics', 'Log Tail', 'Log Insights', and 'CloudWatch 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 several rows of log entries, each starting with a timestamp like '2023-08-08T11:51:00Z' and followed by a detailed log message. The log messages describe network traffic between specific source and destination IP addresses, ports, and protocols (e.g., TCP, UDP). Some entries include packet counts (e.g., '1 56') and byte counts (e.g., '1754665392'). The interface also includes a search bar, filter buttons ('Actions', 'Start tailing', 'Create metric filter'), and a time range selector ('Clear', '1m', '30m', '1h', '12h', 'Customs', 'UTC timezone'). At the bottom right, there's a 'Back to top' button.

# Logs Insights

Logs Insights is a CloudWatch feature that analyzes your logs. In Log Insights, you use queries to filter, process and combine data to help you troubleshoot problems or better understand your network traffic!

I ran the query "Top 10 byte transfers by source and destination IP addresses". This query analyzes the ten pairs of source and destination IP addresses that transferred the most data between them.





[nextwork.org](https://nextwork.org)

# The place to learn & showcase your skills

Check out [nextwork.org](https://nextwork.org) for more projects

