



VPC Monitoring with Flow Logs

L

Louis Moyo

#	srcAddr	dstAddr	bytesTransferred
► 1	10.1.0.4	188.166.211...	853445
► 2	188.166.211.1...	10.1.0.4	730664
► 3	10.1.0.4	10.2.6.148	173964
► 4	10.1.0.4	18.206.107...	100950
► 5	18.206.107.27	10.1.0.4	54504
► 6	10.2.6.148	10.1.0.4	42420
► 7	18.206.107.29	10.1.0.4	13120
► 8	10.1.0.4	18.206.107...	11525
► 9	13.220.37.77	10.1.0.4	7453
► 10	13.217.79.194	10.1.0.4	7401



Introducing Today's Project!

What is Amazon VPC?

Amazon VPC is a secure, isolated section of the AWS cloud where you can launch resources inside a virtual network that you control. It's useful because it lets you customise IP ranges, subnets, route tables, and security settings, enabling secure and efficient communication between resources.

How I used Amazon VPC in this project

I used Amazon VPC to create two separate networks, set up a VPC peering connection, and monitor their traffic using VPC Flow Logs and CloudWatch Log Insights.

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

I didn't expect to see source and destination IPs swapped compared to the example logs - it highlighted how traffic direction depends on which instance initiates the connection.

This project took me...

About 2 hours from VPC setup to running Flow Logs queries and analysing the results.



In the first part of my project...

Step 1 - Set up VPCs

I'm going to use the VPC wizard to quickly create two new VPCs (unique CIDRs, one public subnet each). This sets a clean foundation to revisit peering and then enable Flow Logs + CloudWatch so I can monitor and analyse traffic between them.

Step 2 - Launch EC2 instances

I'm going to launch one EC2 instance in each VPC so they can communicate over the VPC peering connection later, generating network traffic that we can capture and analyse with VPC Flow Logs in CloudWatch.

Step 3 - Set up Logs

I'm going to enable VPC Flow Logs to capture all inbound and outbound traffic for my VPCs, and configure them to store the log records in CloudWatch so I can later search, filter, and analyse the network activity between my instances.



Step 4 - Set IAM permissions for Logs

I'm going to create an IAM policy and role that grants VPC Flow Logs permission to write log data to CloudWatch, then link that role to my subnet's flow log so network traffic records can be stored and analysed.



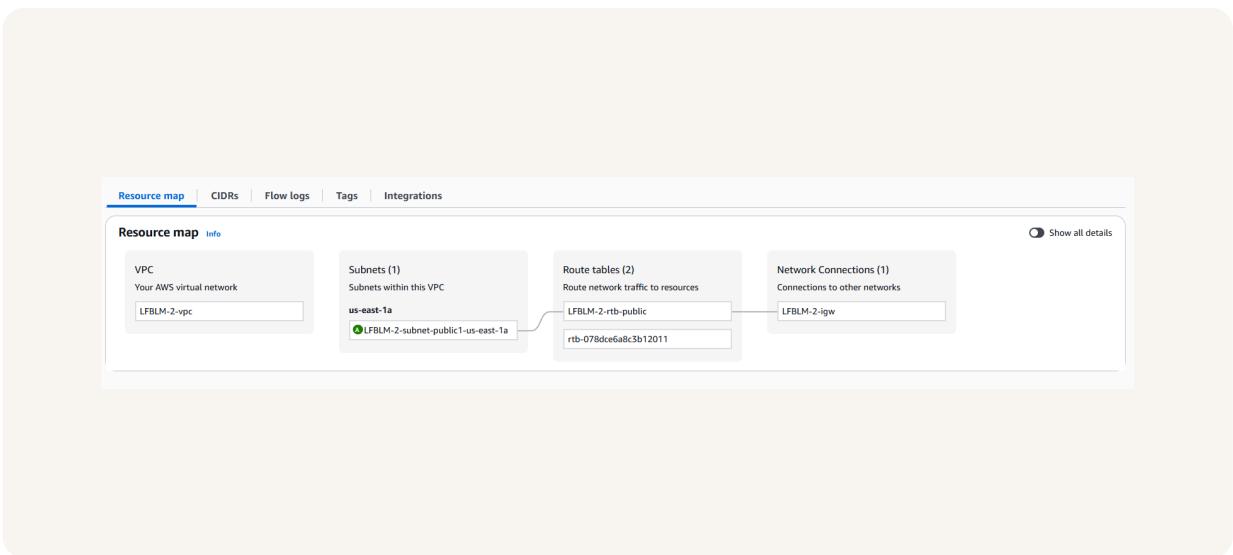
Multi-VPC Architecture

I started my project by launching two new VPCs from scratch using the VPC wizard, each with one public subnet, no private subnets, default tenancy, and no NAT gateways or endpoints.

The CIDR blocks for VPCs 1 and 2 are 10.1.0.0/16 and 10.2.0.0/16. They have to be unique because overlapping ranges would cause routing conflicts and prevent proper communication when connected via VPC peering

I also launched EC2 instances in each subnet

My EC2 instances' security groups allow ICMP from all IPs for Instance 1, but only from 10.1.0.0/16 for Instance 2. This is because Instance 1 needs to be reachable from anywhere, while Instance 2 only needs pings from VPC 1.





Logs

Logs are detailed records of events and activities in a system, such as network traffic, user actions, and errors, used to monitor performance, troubleshoot issues, and track security-related events.

Log groups are containers in CloudWatch that store and organise related logs from the same source or application, making it easier to manage, search, and analyse logs for monitoring and troubleshooting.

The screenshot shows the AWS VPC Flow Logs configuration interface. At the top, there is a table titled "Your VPCs (1/3) Info" showing three VPCs: LFBLM-1-vpc, LFBLM-2-vpc, and another unnamed VPC. The LFBLM-1-vpc row is selected. Below the table, a specific VPC configuration page is shown for "vpc-021ed852097ff6397 / LFBLM-1-vpc". This page includes tabs for "Details", "Resource map", "CIDRs", "Flow logs" (which is currently selected), "Tags", and "Integrations". Under the "Flow logs" tab, there is a table titled "Flow logs (1)" showing one entry: "LFBLMVPCTFlowLog". The "Destination type" is set to "cloud-watch-logs".



IAM Policy and Roles

I created an IAM policy because VPC Flow Logs needs explicit permission to create log groups, create log streams, and send log events to CloudWatch, and this policy grants those permissions for any log group or stream.

I also created an IAM role because VPC Flow Logs needs explicit permission to publish log data into CloudWatch, and creating a role with the right trust policy ensures only this service can assume it and perform that action.

A custom trust policy is a special IAM policy that specifies who or what can assume a role, rather than what actions they can take. In this case, it restricts access so only the `vpc-flow-logs.amazonaws.com` service can use the role.



The screenshot shows the AWS IAM Role configuration page for the role **VPCFlowLogs-Cloudwatch-1755247382140**. The page has a header with tabs for **Summary**, **Permissions**, **Trust relationships** (which is selected), **Tags**, **Last Accessed**, and **Revoke sessions**. On the right, there are buttons for **Delete** and **Edit**.

Summary section:

- Creation date:** August 15, 2025, 09:53 (UTC+01:00)
- Last activity:** 9 minutes ago
- ARN:** arn:aws:iam::756716632322:role/service-role/VPCFlowLogs-Cloudwatch-1755247382140
- Maximum session duration:** 1 hour

Permissions section:

Trusted entities section:

```
1+ [{
2+   "Version": "2012-10-17",
3+   "Statement": [
4+     {
5+       "Effect": "Allow",
6+       "Principal": {
7+         "Service": "vpc-flow-logs.amazonaws.com"
8+       },
9+       "Action": "sts:AssumeRole",
10+      "Condition": {
11+        "StringEquals": {
12+          "aws:SourceAccount": "756716632322"
13+        },
14+        "ArnLike": {
15+          "aws:SourceArn": "arn:aws:ec2:us-east-1:756716632322:vpc-flow-log/*"
16+        }
17+      }
18+    }
19+  ]
20+}]
```



In the second part of my project...

Step 5 - Ping testing and troubleshooting

I'm going to connect to Instance 1 and try sending test messages to Instance 2's private IP address. This will both confirm that the VPC peering connection works and generate traffic that should appear in the flow logs for monitoring.

Step 6 - Set up a peering connection

I'm going to create a VPC peering connection between my two VPCs and update their route tables so they can communicate directly using private IP addresses, ensuring faster, more secure traffic without going over the public internet.

Step 7 - Analyze flow logs

I'm going to review VPC Flow Logs for VPC 1's public subnet to see what network activity was recorded and analyse the traffic for useful insights.

Connectivity troubleshooting

My first ping test between my EC2 instances had no replies, which means the connection over the VPC peering link isn't working yet, likely due to missing or incorrect network permissions, such as NACL rules or route table entries.

```
'`#`          Amazon Linux 2023
`~\`####`      https://aws.amazon.com/linux/amazon-linux-2023
`~\`####`\\
`~\`###`|
`~\`##/`|`->
`~\`V~'`|`-->
`~~`/`|`/
`~~`-.`|`/
`~~`-/`|`/
`~~`/m/`|`/
[ec2-user@ip-10-1-0-4 ~]$ ping 10.2.6.148
PING 10.2.6.148 (10.2.6.148) 56(84) bytes of data.
```

I could receive ping replies if I ran the ping test using the other instance's public IP address, which means Instance 2 is set up to respond to ICMP requests and Instance 1 can reach it over the public internet, though this doesn't confirm VPC peering works.



Connectivity troubleshooting

The issue is VPC 1's route table lacks a route to 10.2.0.0/16 via the peering connection, so private IP traffic goes over the internet. Adding this route ensures traffic stays on the private AWS network, improving speed and security.

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

I also updated both VPCs' route tables so that traffic destined for the other VPC's private IP range is routed through the VPC peering connection, allowing direct private communication without going over the public internet.

The screenshot shows two parts of the AWS Route Tables interface:

Route tables (1/5) [rtb]

Name	Route table ID	Explicit subnet assoc...	Edge associations	Main	VPC	Owner ID
LFBLM-1-rtb-public	rtb-04647fffc2b08e2105	subnet-0f92af951b1aa...	-	No	vpc-021e8522097983937 (LFBL...	750716632322
LFBLM-2-rtb-public	rtb-0151337a0b621f1	-	-	Yes	vpc-04d664f980-1820843	750716632322
LFBLM-1-rtb-peering	rtb-0987b5b2c0de4574	subnet-0a8a9f9210eff...	-	No	vpc-0e0aef1c7a71405db (LFBL...	750716632322
LFBLM-2-rtb-peering	rtb-079a4e4b3b132011	-	-	Yes	vpc-0e0aef1c7a71405db (LFBL...	750716632322
LFBLM-1-rtb-internet	rtb-055d9443c1d929	-	-	Yes	vpc-021e8522097983937 (LFBL...	750716632322

rtb-04647fffc2b08e2105 / LFBLM-1-rtb-public

Routes (3)					
Destination	Target	Status	Propagated	Route Origin	Actions
0.0.0.0	gw-0706bf50012ba1a28	Active	No	Create Route	
10.1.0.0/24	local	Active	No	Create Route Table	
10.2.0.0/24	vpc-09540218219e0419b	Active	No	Create Route	



Connectivity troubleshooting

I received ping replies from Instance 2's private IP address! This means the VPC peering connection and route tables are now correctly configured, allowing direct communication between the two instances over their private IPs without going through the public internet.

```
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

last login: Fri Aug 15 10:10:47 2025 from 18.206.107.27
ec2-user@ip-10-0-4-0: ~$ ping 10.2.6.148
PING 10.2.6.148 (10.2.6.148) 56(84) bytes of data.
64 bytes from 10.2.6.148: icmp_seq=1 ttl=127 time=1.02 ms
64 bytes from 10.2.6.148: icmp_seq=2 ttl=127 time=0.486 ms
64 bytes from 10.2.6.148: icmp_seq=3 ttl=127 time=1.12 ms
64 bytes from 10.2.6.148: icmp_seq=4 ttl=127 time=0.907 ms
64 bytes from 10.2.6.148: icmp_seq=5 ttl=127 time=0.699 ms
64 bytes from 10.2.6.148: icmp_seq=6 ttl=127 time=0.443 ms
64 bytes from 10.2.6.148: icmp_seq=7 ttl=127 time=0.440 ms
64 bytes from 10.2.6.148: icmp_seq=8 ttl=127 time=0.986 ms
64 bytes from 10.2.6.148: icmp_seq=9 ttl=127 time=0.538 ms
64 bytes from 10.2.6.148: icmp_seq=10 ttl=127 time=0.881 ms
64 bytes from 10.2.6.148: icmp_seq=11 ttl=127 time=0.476 ms
64 bytes from 10.2.6.148: icmp_seq=12 ttl=127 time=0.511 ms
64 bytes from 10.2.6.148: icmp_seq=13 ttl=127 time=0.649 ms
64 bytes from 10.2.6.148: icmp_seq=14 ttl=127 time=0.824 ms
64 bytes from 10.2.6.148: icmp_seq=15 ttl=127 time=0.433 ms
64 bytes from 10.2.6.148: icmp_seq=16 ttl=127 time=0.539 ms
64 bytes from 10.2.6.148: icmp_seq=17 ttl=127 time=0.713 ms
64 bytes from 10.2.6.148: icmp_seq=18 ttl=127 time=0.498 ms
64 bytes from 10.2.6.148: icmp_seq=19 ttl=127 time=1.02 ms
64 bytes from 10.2.6.148: icmp_seq=20 ttl=127 time=0.525 ms
64 bytes from 10.2.6.148: icmp_seq=21 ttl=127 time=0.455 ms
64 bytes from 10.2.6.148: icmp_seq=22 ttl=127 time=0.662 ms
64 bytes from 10.2.6.148: icmp_seq=23 ttl=127 time=0.529 ms
64 bytes from 10.2.6.148: icmp_seq=24 ttl=127 time=0.850 ms
64 bytes from 10.2.6.148: icmp_seq=25 ttl=127 time=0.481 ms
64 bytes from 10.2.6.148: icmp_seq=26 ttl=127 time=0.831 ms
64 bytes from 10.2.6.148: icmp_seq=27 ttl=127 time=0.441 ms
64 bytes from 10.2.6.148: icmp_seq=28 ttl=127 time=1.23 ms
64 bytes from 10.2.6.148: icmp_seq=29 ttl=127 time=0.575 ms
```



Analyzing flow logs

Flow logs tell us about network traffic in and out of an interface — version, AWS account ID, network interface ID, source/destination IPs, ports, protocol, packets, bytes, timestamps, action (ACCEPT/REJECT), and log status (OK).

For example, the flow log I've captured tells us that 176 bytes of TCP traffic (port 22) were successfully sent from 18.206.107.27 to 10.1.0.4, and the connection was allowed by both the NACL and SG.

The screenshot shows a terminal window displaying a list of log events. The logs are timestamped and show various network interactions between IP addresses 18.206.107.27 and 10.1.0.4. The logs indicate successful connections (ACCEPT OK) and rejected connections (REJECT OK). The logs also mention specific ports like 22 and 2222, and protocols like TCP. The interface includes a search bar, a filter button, and time selection buttons for 1m, 30m, and 1h.

Timestamp	Message
2025-08-15T10:54:28.000Z	2.76751645322 41+0x0000000000000000 154.189.40.130 10.1.0.4 54235 13905 6 1 40 1751254000 1755254010 REJECT OK
2025-08-15T10:54:28.000Z	2.76751645322 41+0x0000000000000000 162.219.159.214 10.1.0.4 52048 40361 6 1 40 1755254000 1755254010 REJECT OK
2025-08-15T10:54:28.000Z	2.76751645322 41+0x0000000000000000 112.31.149.18 10.1.0.4 57948 9430 6 1 40 1755254000 1755254010 REJECT OK
2025-08-15T10:54:28.000Z	2.76751645322 41+0x0000000000000000 147.135.133.78 10.1.0.4 52022 40396 6 1 40 1755254000 1755254010 REJECT OK
2025-08-15T10:54:28.000Z	2.76751645322 41+0x0000000000000000 147.135.133.244 10.1.0.4 53299 1098 6 1 40 1751254000 1755254010 REJECT OK
2025-08-15T10:54:28.000Z	2.76751645322 41+0x0000000000000000 162.216.149.128 10.1.0.4 54654 10554 6 1 40 1751254000 1755254010 REJECT OK
2025-08-15T10:54:28.000Z	2.76751645322 41+0x0000000000000000 35.203.211.87 10.1.0.4 57232 49695 6 1 40 1755254010 1755254019 REJECT OK
2025-08-15T10:54:28.000Z	2.76751645322 41+0x0000000000000000 162.216.149.149 10.1.0.4 54983 49597 6 1 40 1751254000 1755254010 REJECT OK
2025-08-15T10:54:28.000Z	2.76751645322 41+0x0000000000000000 162.216.149.232 10.1.0.4 54763 49462 6 1 40 1751254000 1755254010 REJECT OK
2025-08-15T10:54:28.000Z	2.76751645322 41+0x0000000000000000 119.235.245.20 10.1.0.4 58728 2000 6 1 40 1755254010 1755254019 REJECT OK
2025-08-15T10:54:28.000Z	2.76751645322 41+0x0000000000000000 287.99.246.12 10.1.0.4 32429 6469 6 1 40 1755254010 1755254019 REJECT OK
2025-08-15T10:54:28.000Z	2.76751645322 41+0x0000000000000000 45.79.151.51 10.1.0.4 40005 69 6 1 40 1750254010 1755254019 REJECT OK
2025-08-15T10:54:28.000Z	2.76751645322 41+0x0000000000000000 154.189.40.221 10.1.0.4 53202 23706 6 1 40 1751254000 1755254019 REJECT OK
2025-08-15T10:54:28.000Z	2.76751645322 41+0x0000000000000000 147.135.133.189 10.1.0.4 54983 49598 6 1 40 1751254000 1755254019 REJECT OK
2025-08-15T10:54:28.000Z	2.76751645322 41+0x0000000000000000 18.206.107.27 10.1.0.4 31246 22 6 1 40 1755254010 1755254019 ACCEPT OK
2025-08-15T10:54:28.000Z	2.76751645322 41+0x0000000000000000 10.1.0.4 18.206.107.27 22 17300 6 2 1N 1755254010 1755254019 ACCEPT OK
2025-08-15T10:54:28.000Z	2.76751645322 41+0x0000000000000000 18.1.0.4 18.206.107.27 22 17300 6 2 1N 1755254010 1755254019 ACCEPT OK
2025-08-15T10:54:28.000Z	2.76751645322 41+0x0000000000000000 162.219.159.152 10.1.0.4 51262 40368 6 1 40 1755254010 1755254019 REJECT OK
2025-08-15T10:54:28.000Z	2.76751645322 41+0x0000000000000000 119.251.69.114 10.1.0.4 54109 21098 6 1 40 1751254000 1755254010 REJECT OK

Logs Insights

Logs Insights is an interactive tool in CloudWatch that lets you search, analyse, and visualise log data using a query language, helping you quickly identify patterns, troubleshoot issues, and gain insights from your logs in near real-time.

I ran the query to return the top 10 byte transfers by source and destination IP addresses. This query analyses the flow logs to identify the largest data transfers, helping pinpoint high-traffic connections for monitoring or troubleshooting.

#	srcAddr	dstAddr	bytesTransferred
► 1	10.1.0.4	188.166.211...	853445
► 2	188.166.211.1...	10.1.0.4	730664
► 3	10.1.0.4	10.2.6.148	173964
► 4	10.1.0.4	18.206.107...	100950
► 5	18.206.107.27	10.1.0.4	54504
► 6	10.2.6.148	10.1.0.4	42420
► 7	18.206.107.29	10.1.0.4	13120
► 8	10.1.0.4	18.206.107...	11525
► 9	13.220.37.77	10.1.0.4	7453
► 10	13.217.79.194	10.1.0.4	7401



nextwork.org

The place to learn & showcase your skills

Check out nextwork.org for more projects

