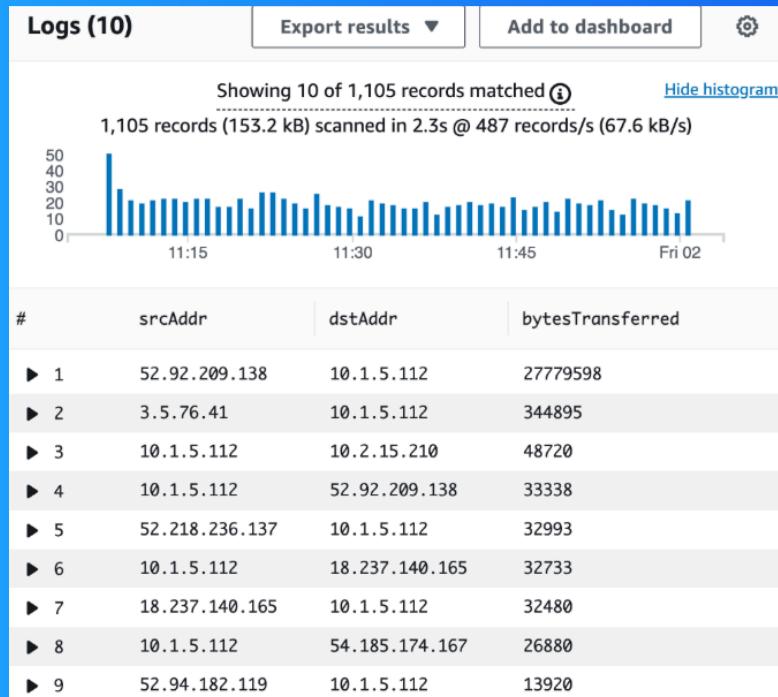




# VPC Monitoring with Flow Logs



samahae1530@gmail.com



# Introducing Today's Project!

## What is Amazon VPC?

Amazon VPC (Virtual Private Cloud) is a service that lets the user to create their own private network inside the AWS cloud. It's like having our own secure area where you can run your applications, store data, and control who can access it.

## How I used Amazon VPC in this project

first learnt to troubleshoot VPC peering connectivity issues. secondly learnt monitor network traffic using VPC flow logs.

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

log insights have a lot of built in queries that gives us many options on how we like to analyze network traffic.

## This project took me...

this project took me 2 hours and half an hour for documentation.

# In the first part of my project...

## Step 1 - Set up VPCs

In this step, I set up two VPCs from scratch within minutes. Network monitoring can also be performed using a single VPC and utilizes VPC peering for communication.

## Step 2 - Launch EC2 instances

In this step, two EC2 instances are launched, one in each VPC. These instances are essential for the next stages of the project, as they will generate traffic that VPC Flow Logs will monitor.

## Step 3 - Set up Logs

Setting up VPC flow logs to start monitoring traffic also setting up storage space for flow logs.

## Step 4 - Set IAM permissions for Logs

Providing VPC flow logs with permissions to create logs and upload them into log group in CloudWatch.

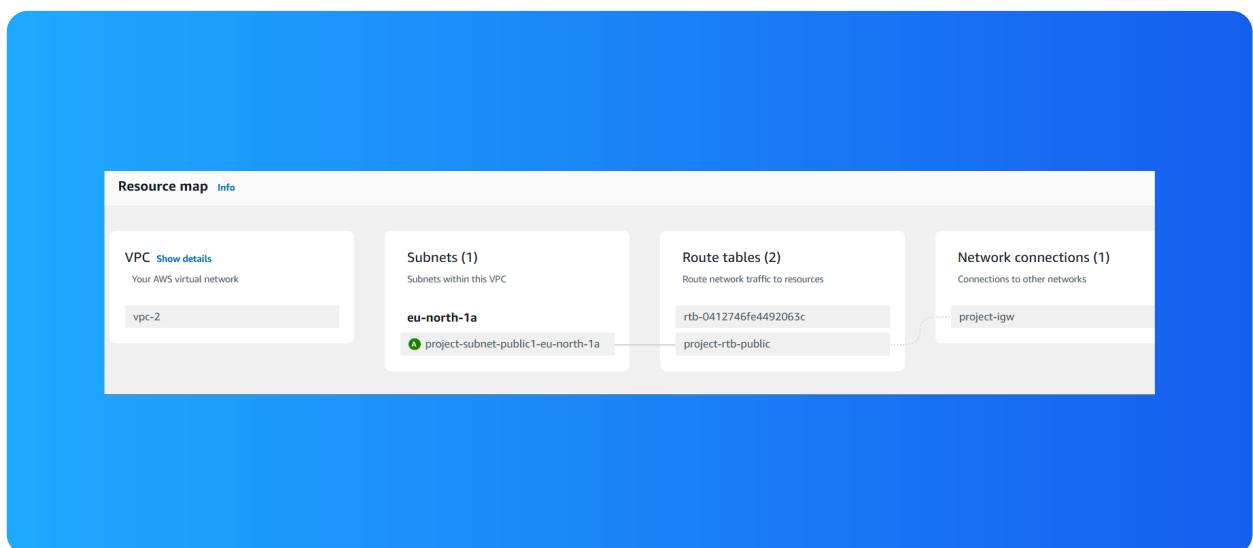
# Multi-VPC Architecture

I started my project by launching two public subnets ( one public subnet in each vpc) with no private subnets

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 having overlapping CIDR blocks will cause routing/traffic issues down the line when traffic when traffic is needing to go from one vpc to another.

## I also launched EC2 instances in each subnet

EC2 instances security groups allow SSH and ICMP type traffic. This is because EC2 instance connect will need to access EC2 instance using SSH traffic and need to allow ICMP traffic for connectivity tests later.



# Logs

Logs are records of events or activities that occur within applications or AWS services. They help to understand what's happening in the environment, troubleshoot issues, and monitor performance.

Log groups are a way to organize and manage logs in Amazon CloudWatch Logs. A log group is essentially a container for related logs, making it easier to manage and analyze them together.

Flow logs (1/1) <a href="#">Info</a>						
<input type="text"/> Search		Actions		<a href="#">Create flow log</a>		
<input checked="" type="checkbox"/>	Name	Flow log ID	Filter	Destination type	Destination name	IAM role ARN
<input checked="" type="checkbox"/>	vpcfloglogs 	fl-0667e1cf005e28cb7	ALL	cloud-watch-logs	vpcloggroup 	arn:aws:iam

# IAM Policy and Roles

I created an IAM policy because so that i can define a rule that allows policy holders e.g. vpc flow log service the ability to create log streams and upload them into cloudwatch.

I also created an IAM role because services like vpc flow logs have to be associated with a role instead of json. creating an IAM role will be necessary to give vpc flow logs the acess it needs to record and upload logs

A custom trust policy is a specific type of policy used for designing who/what allowed to access to the IAM role.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Statement1",
      "Effect": "Allow",
      "Principal": {
        "Service": "vpc-flow-logs.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

# In the second part of my project...

## Step 5 - Ping testing and troubleshooting

In this step, creating a network traffic this becomes important when communicating about cloudwork or cloud networking.

## Step 6 - Set up a peering connection

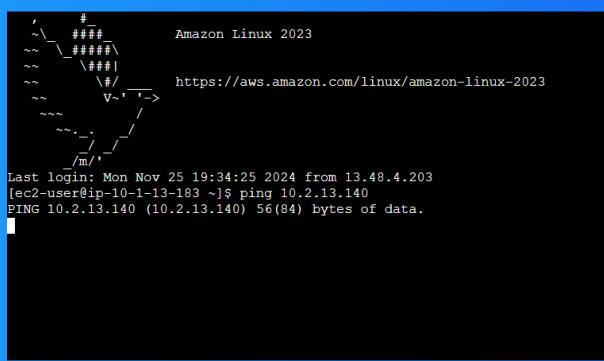
setting up peering connection so that vpc 1 and 2 can communicate directly to each other.

## Step 7 - Analyze flow logs

In this step, tracking the network data that's been collected from vpc and then analyze that data to extract insights.

# Connectivity troubleshooting

My first ping test between my EC2 instances had no replies which means ICMP traffic could be blocked by security group/network ACLS or maybe traffic is routed to a wrong path.



A terminal window on an Amazon Linux 2023 instance. The window title is 'Amazon Linux 2023' and the URL is 'https://aws.amazon.com/linux/amazon-linux-2023'. The terminal shows a login message: 'Last login: Mon Nov 25 19:34:25 2024 from 13.48.4.203 [ec2-user@ip-10-1-13-183 ~]\$'. A command is run: 'ping 10.2.13.140'. The output shows the ping command and its parameters: 'PING 10.2.13.140 (10.2.13.140) 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 second instance is allowing ICMP traffic.

# Connectivity troubleshooting

Looking at VPC 1's route table, I identified that the ping test with Instance 2's private address failed because the vpc doesn't have route in vpc's route table that directs traffic from one vpc to another.

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

updated both VPCs' route tables so that traffic from one of the vpc's and heading to other vpc's ipv4 address can get directed to go through peering connection instead of the public internet.

Routes (3)				Both	Edit routes
Destination	Target	Status	Propagated		
0.0.0.0/0	<a href="#">igw-010f67bdbbe16c5d6</a>	Active	No		
10.1.0.0/16	local	Active	No		
10.2.0.0/16	<a href="#">pxx-0b6f35fb379380e9e</a>	Active	No		

# Connectivity troubleshooting

received ping replies from Instance 2's private IP address! This means setting up the peering connection and the route table solved the connectivity error of vpc's traffic not being able to navigate from one vpc to the other.

```
64 bytes from 10.2.13.140: icmp_seq=1720 ttl=127 time=0.189 ms
64 bytes from 10.2.13.140: icmp_seq=1721 ttl=127 time=0.185 ms
64 bytes from 10.2.13.140: icmp_seq=1722 ttl=127 time=0.171 ms
64 bytes from 10.2.13.140: icmp_seq=1723 ttl=127 time=0.252 ms
64 bytes from 10.2.13.140: icmp_seq=1724 ttl=127 time=0.218 ms
64 bytes from 10.2.13.140: icmp_seq=1725 ttl=127 time=0.211 ms
64 bytes from 10.2.13.140: icmp_seq=1726 ttl=127 time=0.180 ms
64 bytes from 10.2.13.140: icmp_seq=1727 ttl=127 time=0.208 ms
64 bytes from 10.2.13.140: icmp_seq=1728 ttl=127 time=0.173 ms
64 bytes from 10.2.13.140: icmp_seq=1729 ttl=127 time=0.184 ms
64 bytes from 10.2.13.140: icmp_seq=1730 ttl=127 time=0.368 ms
64 bytes from 10.2.13.140: icmp_seq=1731 ttl=127 time=0.175 ms
64 bytes from 10.2.13.140: icmp_seq=1732 ttl=127 time=0.207 ms
64 bytes from 10.2.13.140: icmp_seq=1733 ttl=127 time=0.187 ms
64 bytes from 10.2.13.140: icmp_seq=1734 ttl=127 time=0.241 ms
64 bytes from 10.2.13.140: icmp_seq=1735 ttl=127 time=0.175 ms
64 bytes from 10.2.13.140: icmp_seq=1736 ttl=127 time=0.199 ms
64 bytes from 10.2.13.140: icmp_seq=1737 ttl=127 time=0.179 ms
64 bytes from 10.2.13.140: icmp_seq=1738 ttl=127 time=0.201 ms
64 bytes from 10.2.13.140: icmp_seq=1739 ttl=127 time=0.239 ms
64 bytes from 10.2.13.140: icmp_seq=1740 ttl=127 time=0.199 ms
64 bytes from 10.2.13.140: icmp_seq=1741 ttl=127 time=0.270 ms
64 bytes from 10.2.13.140: icmp_seq=1742 ttl=127 time=0.226 ms
64 bytes from 10.2.13.140: icmp_seq=1743 ttl=127 time=0.182 ms
64 bytes from 10.2.13.140: icmp_seq=1744 ttl=127 time=0.201 ms
64 bytes from 10.2.13.140: icmp_seq=1745 ttl=127 time=0.186 ms
64 bytes from 10.2.13.140: icmp_seq=1746 ttl=127 time=0.179 ms
64 bytes from 10.2.13.140: icmp_seq=1747 ttl=127 time=0.191 ms
64 bytes from 10.2.13.140: icmp_seq=1748 ttl=127 time=0.173 ms
```

# Analyzing flow logs

Flow logs tell us about the source and destination of network traffic, the amount of data being transferred, whether the traffic was accepted or rejected.

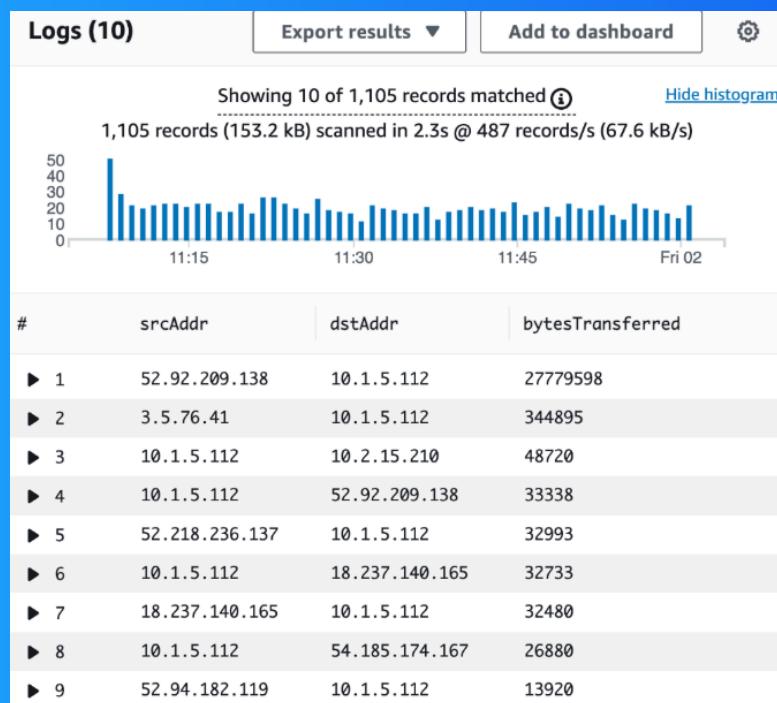
For example, the flow log I've captured tells us that the traffic went from 18.237.140.165 to 10.1.5.112.

```
# 2024-08-01T23:45:42.000Z      2 471112976395 eni-08a0e21a6bb867b64 18.237.140.165 10.1.5.112 46093 22 6 4 344 172255..  
2 471112976395 eni-08a0e21a6bb867b64 18.237.140.165 10.1.5.112 46093 22 6 4 344 1722555942 1722556001 ACCEPT OK
```

# Logs Insights

logs insight is a special tool within amazon cloudwatch that help us with analysing logs and creating visual graphs and chart through queries.

I ran the query the top 10 byte transfers by source and destination IP addresses. This query analyzes the flow logs collected on EC2 instance 1.





NextWork.org

# Everyone should be in a job they love.

Check out nextwork.org for  
more projects

