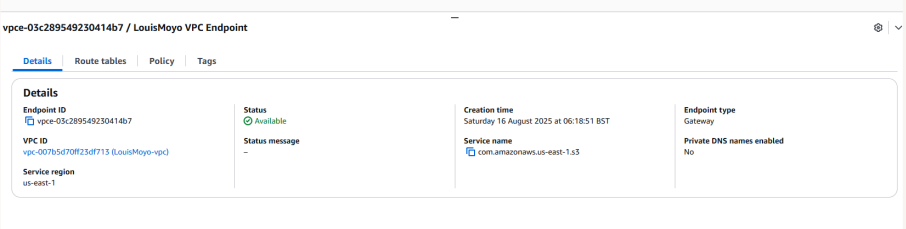
nextwork.org

VPC Endpoints

Louis Moyo



L
Louis Moyo
NextWork Student

Introducing Today's Project!

What is Amazon VPC?

Amazon VPC is a private virtual network in AWS where you can launch resources like EC2 and connect them securely. It's useful because it gives control over IP ranges, subnets, routing, and connectivity to other AWS services.

How I used Amazon VPC in this project

I used Amazon VPC to create a secure environment with an EC2 instance, then connected it to S3 using a VPC endpoint. This let my instance access S3 privately without sending traffic over the internet.

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

One thing I didn't expect in this project was that once I applied a restrictive bucket policy, even the AWS Management Console was blocked until the VPC endpoint was used.

This project took me...

This project took me about 75 minutes to complete, including creating the VPC, setting up the EC2 instance, creating the S3 bucket, and configuring the VPC endpoint with bucket policies.

In the first part of my project...

Step 1 - Architecture set up

In this step I'm setting up the foundations of my project by creating a new VPC, launching an EC2 instance that I can later connect to with EC2 Instance Connect, and creating an S3 bucket for storage.

Step 2 - Connect to EC2 instance

In this step I'm connecting directly to my EC2 instance using EC2 Instance Connect. This lets me open the terminal on the instance so I can run AWS CLI commands and test S3 access over the public internet.

Step 3 - Set up access keys

In this step I will create access keys to give my EC2 instance credentials, allowing it to securely connect to my AWS environment and interact with services like S3 using the CLI.

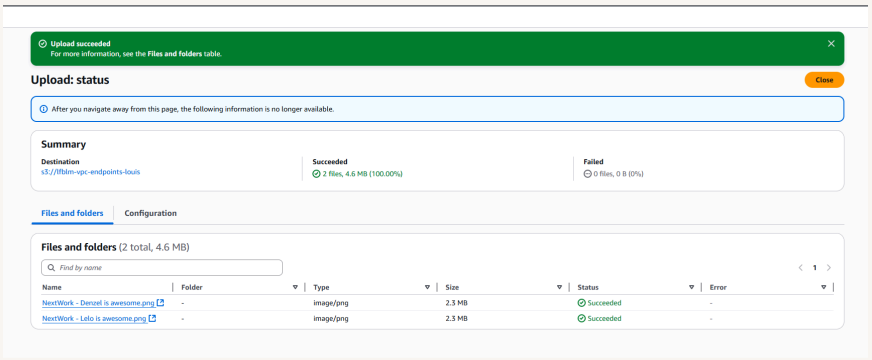
Step 4 - Interact with S3 bucket

In this step I'm heading back to my EC2 instance and using the AWS CLI to access my S3 bucket. This lets me confirm that the instance can securely connect and list objects in the bucket.

Architecture set up

I started my project by launching a new VPC named LouisMoyo with the default CIDR block 10.0.0.0/16, one public subnet, and an EC2 instance called Instance – LouisMoyo VPC Endpoints with a security group for SSH access.

I also set up an S3 bucket named lfblm-vpc-endpoints-louis and uploaded two files into it. This bucket will be used to practise accessing and listing objects from my EC2 instance.



L
Louis Moyo
NextWork Student

Access keys

Credentials

To set up my EC2 instance to interact with my AWS environment, I configured my access key ID, secret access key, default Region name, and default output format.

Access keys are a pair of credentials (an access key ID and a secret access key) that allow secure programmatic access to AWS services through the CLI or SDKs.

Secret access keys are like the password paired with an access key ID (the username). Together they provide programmatic access to AWS services and must be kept secure.

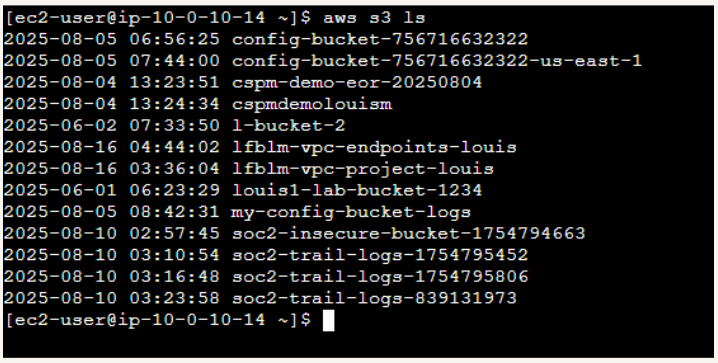
Best practice

Although I'm using access keys in this project, a best practice alternative is to use IAM roles with temporary credentials, which are automatically managed and more secure.

Connecting to my S3 bucket

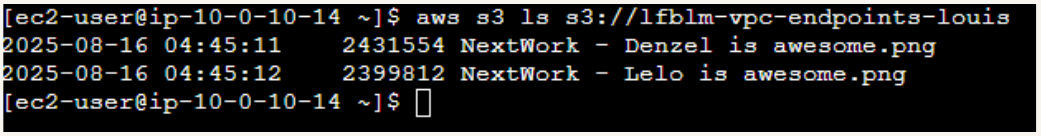
The command I ran was aws s3 ls. This command is used to list all the S3 buckets in my account and confirm that my EC2 instance can connect to AWS services through the CLI.

The terminal responded with a list of my S3 buckets. This indicated that the access keys I set up were working correctly and my EC2 instance was successfully connected to my AWS environment.



Connecting to my S3 bucket

I also tested the command aws s3 ls s3://lfblm-vpc-endpoints-louis which returned the objects inside my S3 bucket, confirming that my EC2 instance could access the files.

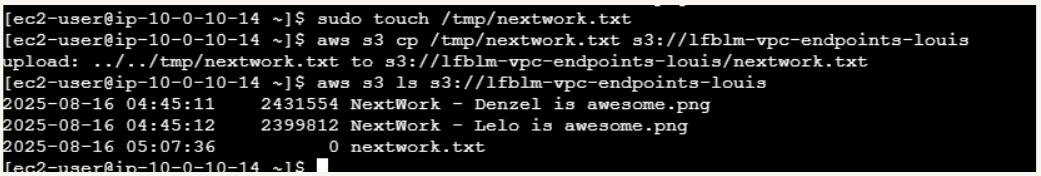


Uploading objects to S3

To upload a new file to my bucket, I first ran the command sudo touch /tmp/nextwork.txt. This command creates an empty text file in the /tmp directory to be uploaded to S3.

The second command I ran was aws s3 cp /tmp/nextwork.txt s3://lfblm-vpc-endpoints-louis/nextwork.txt. This command will copy my new text file into the S3 bucket.

The third command I ran was aws s3 ls s3://lfblm-vpc-endpoints-louis. This validated that the new file nextwork.txt was successfully uploaded and appeared in my S3 bucket.



In the second part of my project...

Step 5 - Set up a Gateway

I'm creating a VPC endpoint for S3 so my VPC talks to S3 privately instead of the public internet.

Step 6 - Bucket policies

I'm locking down my S3 bucket with a policy that only allows access via my S3 VPC endpoint and blocks everything else. This proves traffic stays within AWS.

Step 7 - Update route tables

I'm testing my VPC endpoint setup by trying to access my S3 bucket again from my EC2 instance. This confirms that my instance can still reach S3 privately through the endpoint, even though public access is blocked.

Step 8 - Validate endpoint conection

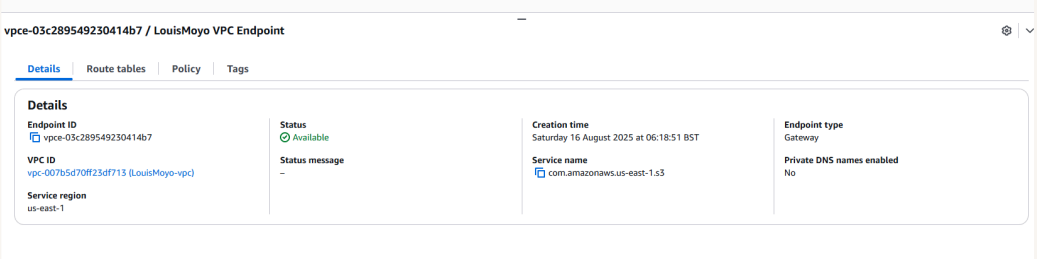
I'm validating my setup by accessing my S3 bucket from my EC2 instance one last time. This confirms that my VPC endpoint is working correctly and that bucket access is fully restricted to the private AWS network.

Setting up a Gateway

I set up an S3 Gateway, which is a type of VPC endpoint that routes traffic between my VPC and Amazon S3 directly over the AWS network.

What are endpoints?

An endpoint is a private connection that lets resources in a VPC securely access AWS services without using the public internet.



Bucket policies

A bucket policy is a JSON-based resource policy that defines permissions and access rules for an S3 bucket and the objects inside it.

My bucket policy will block all access to my S3 bucket except traffic that comes through my VPC endpoint, ensuring secure private access only.

Policy

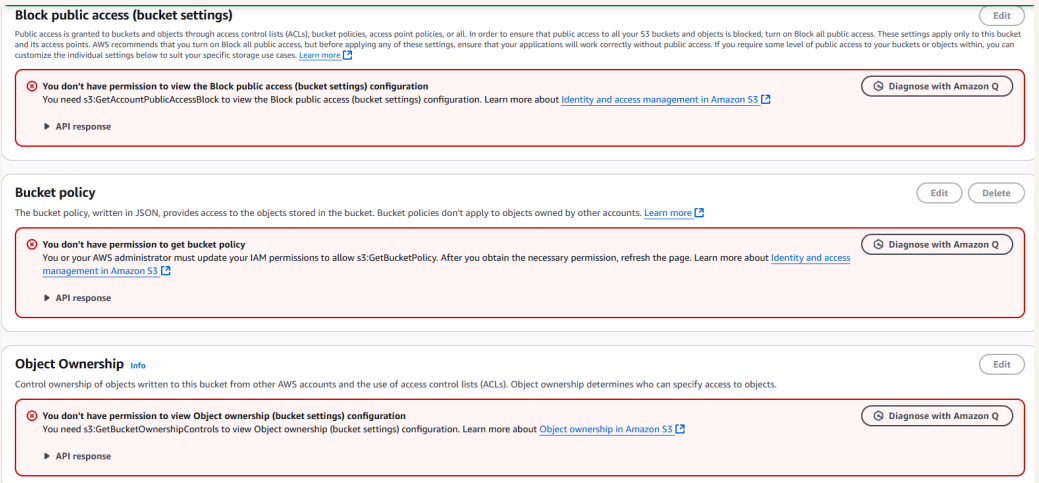
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::lfblm-vpc-endpoints-louis",
        "arn:aws:s3:::lfblm-vpc-endpoints-louis/*"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:sourceVpce": "vpce-03c289549230414b7"
        }
      }
    }
  ]
}
```

Louis Moyo
NextWork Student

Bucket policies

Right after saving my bucket policy, my S3 bucket page showed 'denied access' warnings. This was because my policy denies all access unless it comes through my VPC endpoint, which blocks even the AWS Console.

I also had to update my route table because without a route to my VPC endpoint, my EC2 instance was trying to reach S3 over the public internet instead of through the private AWS network.

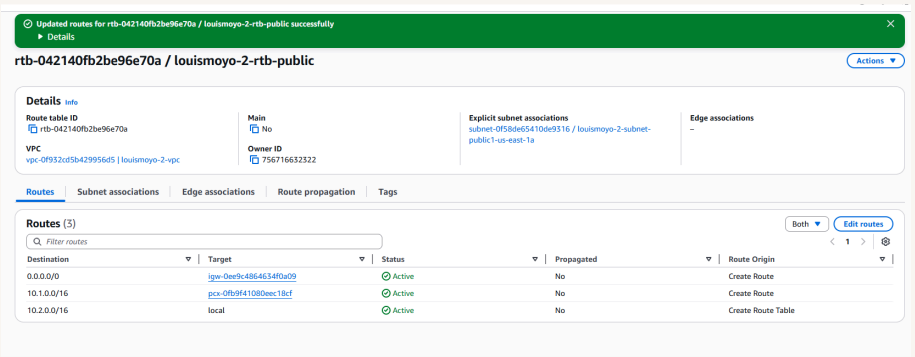


Louis Moyo
NextWork Student

Route table updates

To update my route table, I associated my public subnet's route table with the S3 VPC endpoint, so traffic from my EC2 instance is now directed securely to S3 via the endpoint.

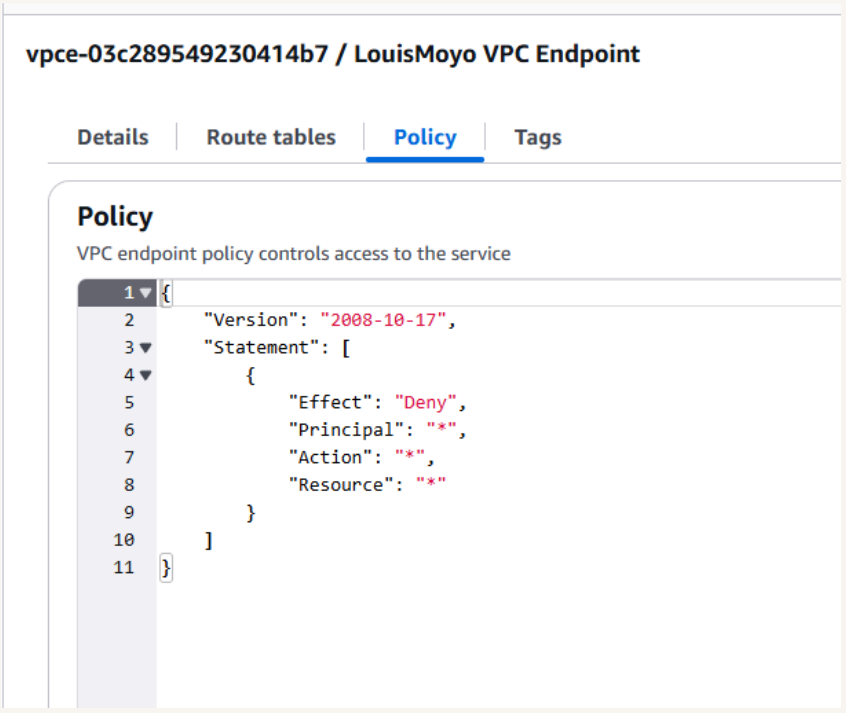
After updating my public subnet's route table, my terminal could return the list of objects inside my S3 bucket, confirming that the VPC endpoint connection was working correctly.



Endpoint policies

An endpoint policy is a resource-based JSON policy attached to a VPC endpoint that controls which AWS services and actions can be accessed through that endpoint.

I updated my endpoint's policy by changing the effect from "Allow" to "Deny". I could see the effect of this right away, because my EC2 instance was no longer able to access my S3 bucket when I ran the aws s3 ls command.



The place to learn & showcase your skills

Check out nextwork.org for more projects