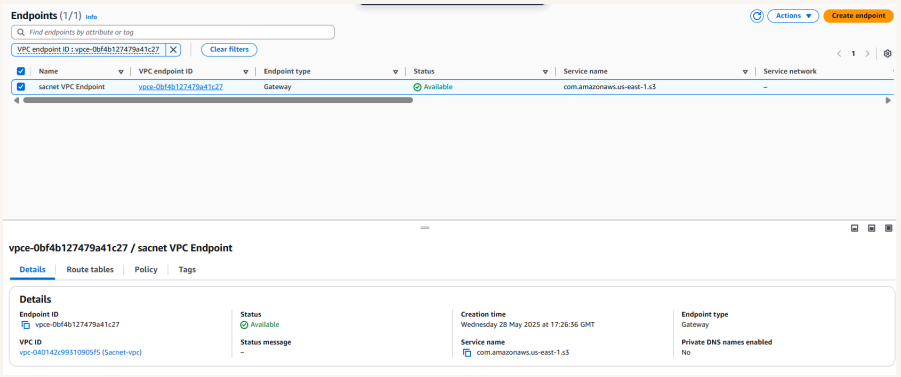
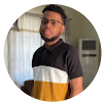
nextwork.org

VPC Endpoints

Jonathan Nutsugah





Jonathan Nutsugah
NextWork Student

Introducing Today's Project!

What is Amazon VPC?

Amazon VPC (Virtual Private Cloud) lets you create a secure, isolated network in AWS where you control IP ranges, subnets, and traffic. It's useful for customizing network setup, enhancing security, and supporting hybrid cloud deployments.

How I used Amazon VPC in this project

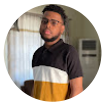
Through the management console, I used it to virtually isolate a space to keep my resources away from external access.

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

How easy it is to navigate AWS.

This project took me...

2 hours



Jonathan Nutsugah
NextWork Student

In the first part of my project...

Step 1 - Architecture set up

In this step, I'm going to create a new VPC, launch an EC2 instance inside it, and set up an S3 bucket. This setup will form the foundation of my project, allowing me to later test secure, direct access between my EC2 instance and the S3 bucket.

Step 2 - Connect to EC2 instance

In this step, I'm going to connect directly to my EC2 instance using EC2 Instance Connect. This allows me to securely access the instance's terminal so I can run commands and test the connection to my S3 bucket later.

Step 3 - Set up access keys

In this step, I'm going to create access keys so my EC2 instance can use credentials to access AWS services. This lets my instance interact with AWS resources securely while I set up and test my environment.

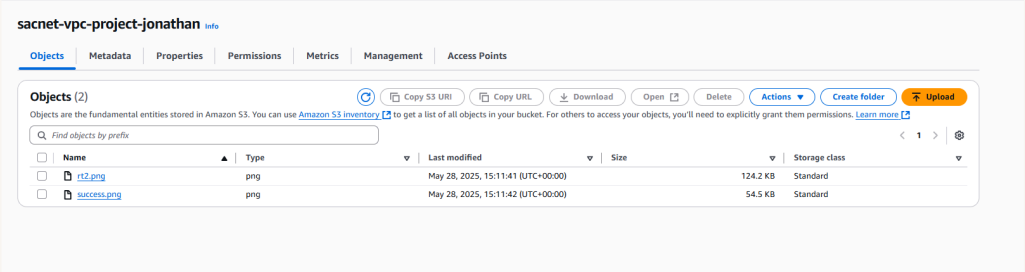
Step 4 - Interact with S3 bucket

In this step, I'm going back to my EC2 instance to enable it to access my S3 bucket. This lets me test that my instance can securely interact with the storage service and manage files as needed.

Architecture set up

I started my project by launching an EC2 instance, which I'll later use to test connectivity to my S3 bucket through a VPC endpoint.

I also set up an S3 bucket with the General Purpose type, which I'll use to store files and test secure access from my EC2 instance using a VPC endpoint.



Access keys

Credentials

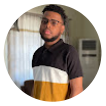
To set up my EC2 instance to interact with my AWS environment, I configured the Access Key ID, the Secret Access Key, the default AWS region, and left the default output format empty.

Access keys are a pair of credentials (an access key ID and a secret access key) that allow programmatic access to AWS services. They are used by tools like the AWS CLI to authenticate and authorize actions in your AWS account.

Secret access keys are a confidential part of AWS access keys used to securely sign API requests. They work with the access key ID to verify your identity and grant permissions when accessing AWS services programmatically.

Best practice

Although I'm using access keys in this project, a best practice alternative is to use AWS Cloudshell.



Jonathan Nutsugah
NextWork Student

Connecting to my S3 bucket

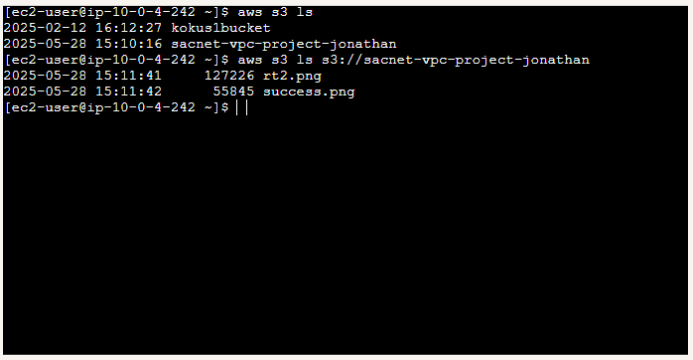
The command I ran was aws s3 ls. This command is used to list all the S3 buckets available in my AWS account, confirming that my instance can access S3.

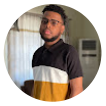
When I ran the aws s3 ls command, the terminal responded with a list of my S3 buckets. This confirmed that my EC2 instance was successfully configured to access my AWS environment and could interact with S3.

```
Default output format [None]:
[ec2-user@ip-10-0-4-242 ~]$ aws s3 ls
2025-02-12 16:12:27 kokus1bucket
2025-05-28 15:10:16 sacnet-vpc-project-jonathan
[ec2-user@ip-10-0-4-242 ~]$
```

Connecting to my S3 bucket

I also tested the command aws s3 ls s3://sacnet-vpc-project-yourname, which returned a list of the objects stored inside my S3 bucket. This confirmed that my EC2 instance could successfully access and view the contents of the bucket.





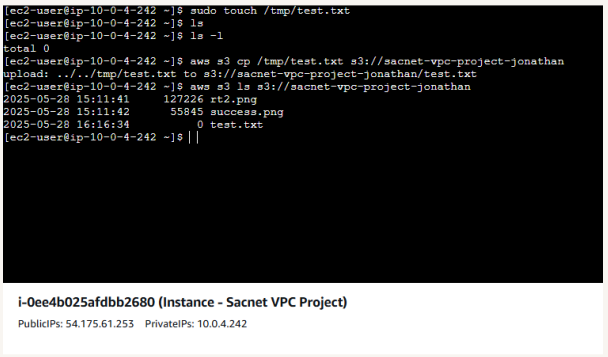
Jonathan Nutsugah
NextWork Student

Uploading objects to S3

To upload a new file to my bucket, I first ran the command sudo touch /tmp/test.txt. This command creates an empty file named test.txt in the /tmp directory on my EC2 instance.

The second command I ran was aws s3 cp /tmp/test.txt s3://nextwork-vpc-project-yourname. This command will copy the test.txt file from my EC2 instance to my S3 bucket, uploading it to the cloud.

The third command I ran was aws s3 ls s3://nextwork-vpc-project-yourname, which validated that my file was successfully uploaded to the S3 bucket by listing it in the output.



In the second part of my project...

Step 5 - Set up a Gateway

In this step, I'm setting up a VPC endpoint so my VPC can communicate directly with my S3 bucket—without going through the public internet. This makes access faster, more secure, and keeps traffic within the AWS network.

Step 6 - Bucket policies

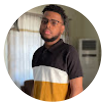
In this step, I'm going to update my S3 bucket's permissions to only allow access from my VPC endpoint. This adds an extra layer of security by making sure that only traffic coming through my private network can interact with the bucket.

Step 7 - Update route tables

In this step, I'm going to test whether my EC2 instance can access my S3 bucket through the VPC endpoint I set up. This helps confirm that my private connection between the VPC and S3 is working as expected.

Step 8 - Validate endpoint conection

In this step, I'm going to test my VPC endpoint setup again to make sure it's working correctly. Then, I'll restrict my VPC's access to my AWS environment to improve security by limiting what resources and traffic can interact with my VPC.



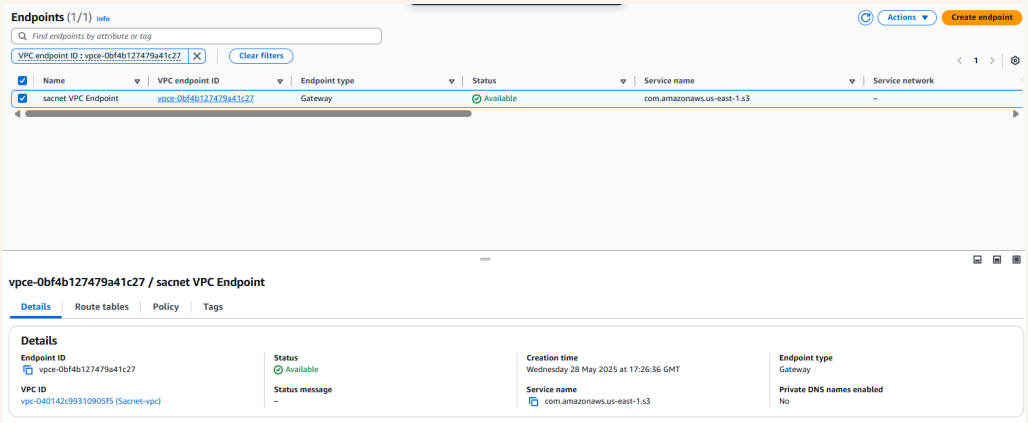
Jonathan Nutsugah
NextWork Student

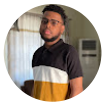
Setting up a Gateway

I set up an S3 Gateway, which is a type of VPC endpoint that allows my VPC to privately connect to Amazon S3 without using the public internet. This makes data transfer more secure and efficient by keeping traffic within the AWS network.

What are endpoints?

An endpoint is a connection point that lets your VPC privately connect to supported AWS services, like S3, without using the public internet. It helps improve security and performance by keeping traffic within the AWS network.



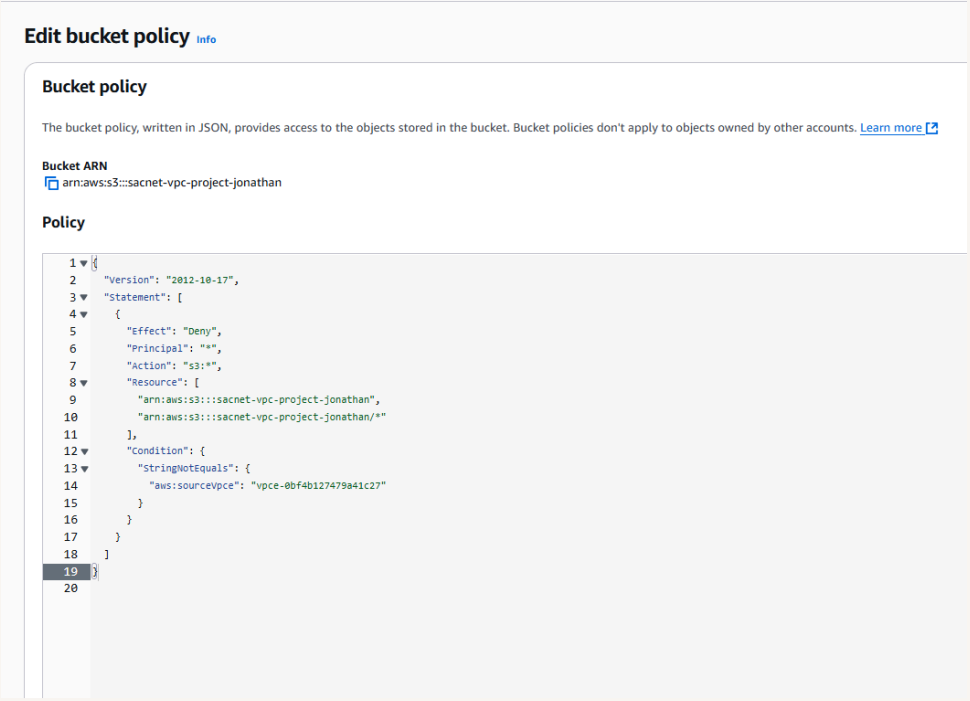


Jonathan Nutsugah
NextWork Student

Bucket policies

A bucket policy is a set of rules, written in JSON, that defines who can access an Amazon S3 bucket and what actions they can perform.

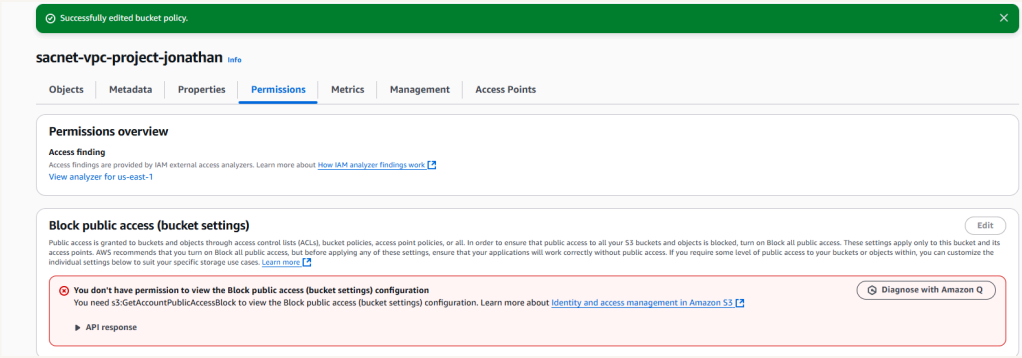
My bucket policy will deny all access to the bucket unless the request comes from a specific VPC endpoint. It uses a condition that checks if the request's source VPC endpoint ID matches the one I specified.

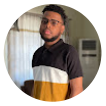


Bucket policies

My bucket policy denies all access unless the request comes from my VPC endpoint. This means any attempt to access my bucket from other sources, including the AWS Management Console, is blocked.

I also had to update my route table because my VPC needed a way to know how to reach the S3 service through the VPC endpoint. Without this route, traffic from my EC2 instance wouldn't be directed through the private connection to S3.



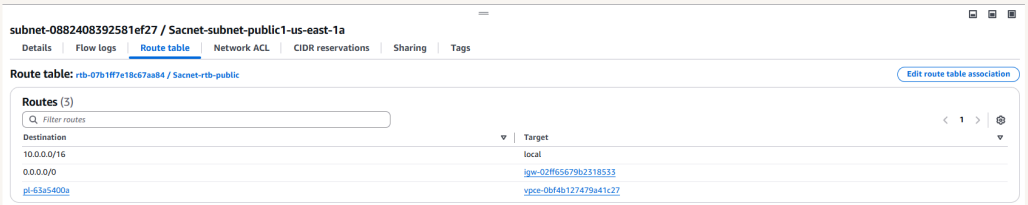


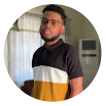
Jonathan Nutsugah
NextWork Student

Route table updates

To update my route table, I added a route in my public subnet's route table that directs traffic destined for the S3 service to go through my VPC endpoint instead of the public internet.

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



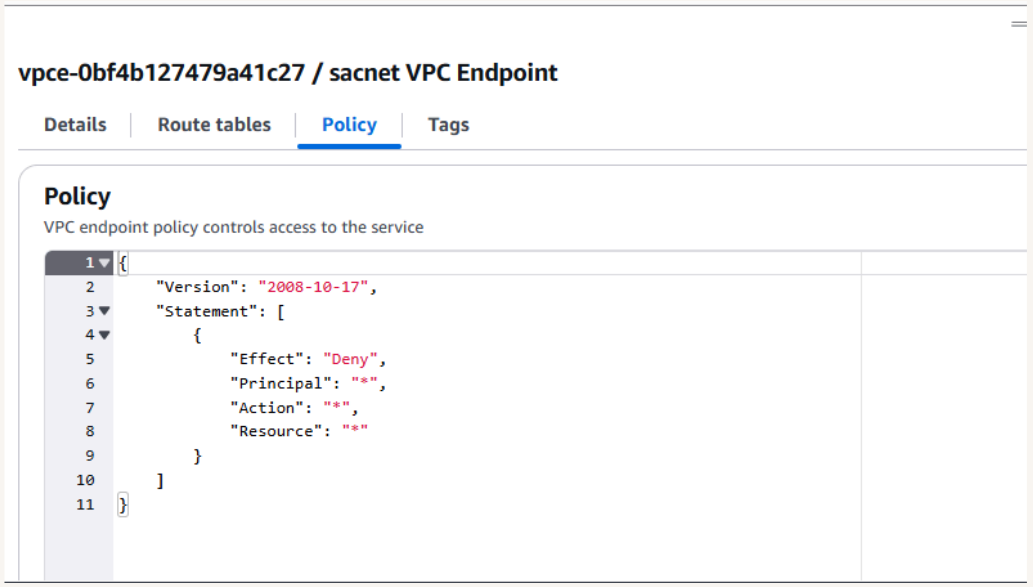


Jonathan Nutsugah
NextWork Student

Endpoint policies

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

I updated my endpoint's policy by modifying it to allow only specific actions or restrict access to certain S3 buckets. I could see the effect of this right away, because my EC2 instance was either granted or denied access to those resources.



The place to learn & showcase your skills

Check out nextwork.org for more projects