

VPC Endpoints

Erik Gonzalez

The screenshot shows the AWS VPC Endpoints service interface. At the top, a green success message box displays: "Successfully created VPC endpoint vpce-0ac0600189fcf005c". Below this, the main "Endpoints (1/1)" table lists one entry:

Name	VPC endpoint ID	Endpoint type	Status	Service name
Nimbus VPC Endpoint	vpce-0ac0600189fcf005c	Gateway	Available	com.amazonaws.us-e...

Below the table, a detailed view for the endpoint "vpce-0ac0600189fcf005c / Nimbus VPC Endpoint" is shown. The "Details" section contains the following information:

Endpoint ID vpce-0ac0600189fcf005c	Status Available	Creation time Friday, December 5, 2025 at 10:11:54 EST	Endpoint type Gateway
VPC ID vpc-05204117a43f265d9 (Nimbus-vpc)	Status message -	Service name com.amazonaws.us-east-1.s3	Private DNS names enabled No
DNS record IP type service-defined	IP address type ipv4	Service region us-east-1	Private DNS preference -
Private DNS specified domains			

Introducing Today's Project!

What is Amazon VPC?

Amazon VPC is a tool in AWS that allows users to segment and isolate their network from the public internet to monitor and control their own private sector.

How I used Amazon VPC in this project

In today's project, I used Amazon VPC to create an EC2 instance, subnets, edit policies, communicate between networks, insert commands in CLI.

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

One thing I didn't expect in this project was the amount of errors I would get when editing policies. It was just a huge wall of red!

This project took me...

This project took me about two hours to complete while also taking a lunch break!

In the first part of my project...

Step 1 – Architecture set up

In this step, I will create a VPC from scratch, launch an EC2 instance, and set up an S3 bucket.

Step 2 – Connect to EC2 instance

In this step, I will connect directly to your EC2 instance.

Step 3 – Set up access keys

In this step, I will give my EC2 instance access to your AWS environment

Step 4 – Interact with S3 bucket

In this step, I will head back to the EC2 instance and get the EC2 instance to access the S3 bucket.

Architecture set up

I started my project by launching VPC to start my VPC and EC2 to start a brand new Instance!

I also set up an S3 bucket with two file uploads.

The screenshot shows an AWS S3 upload status page. At the top, a green banner displays a success message: "Upload succeeded. For more information, see the Files and folders table." On the right side of the banner is a "Close" button. Below the banner, the title "Upload: status" is followed by a note: "After you navigate away from this page, the following information is no longer available." A "Close" button is located on the right side of this note. The main area is titled "Summary". It shows the destination as "s3://nimbus1-vpc-project-eg". Under "Succeeded", it lists "2 files, 4.6 MB (100.00%)". Under "Failed", it lists "0 files, 0 B (0%)". Below the summary, there are tabs for "Files and folders" (which is selected) and "Configuration". The "Files and folders" tab shows a table with two rows of uploaded files. The columns are: Name, Folder, Type, Size, Status, Error, and a sorting icon. The first file is "NextWork - Denzel is awesome...L" (Folder), image/png, 2.3 MB, Succeeded, -. The second file is "NextWork - Lelo is awesome.pr..." (Folder), image/png, 2.3 MB, Succeeded, -.

Name	Folder	Type	Size	Status	Error
NextWork - Denzel is awesome...L	-	image/png	2.3 MB	Succeeded	-
NextWork - Lelo is awesome.pr...	L	image/png	2.3 MB	Succeeded	-

Access Keys

Credentials

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

Access keys are credentials for your applications and other servers to log into AWS and talk to your AWS services/resources.

The secret access key is like the password that pairs with your access key ID (your username). You need both to access AWS services.

Best practice

Although I'm using access keys in this project, a best practice alternative is to use IAM admin roles. This means necessary permission will be attached to an IAM role, and then the role will be associated with the relevant resources.

Connecting to my S3 Bucket

The command I ran was 'aws s3 ls' . This command is used to list all buckets in an AWS account.

The terminal responded with a list of my account's S3 bucket. This indicated that the access keys I set up correctly give my EC2 instance access to my AWS account and environment.

```
Default output format [None]:  
[ec2-user@ip-10-0-8-179 ~]$ aws s3 ls  
2025-12-05 13:59:47 nimbus1-vpc-project-eg  
[ec2-user@ip-10-0-8-179 ~]$ █
```

Connecting to my S3 Bucket

I also tested the command 'aws s3 ls s3://nimbus1-vpc-project-eg' , which returned a list of files I uploaded to my S3 bucket.

```
2025-12-05 13:59:47 nimbus1-vpc-project-eg
[ec2-user@ip-10-0-8-179 ~]$ aws s3 ls s3://nimbus1-vpc-project-eg
2025-12-05 14:03:45      2431554 NextWork - Denzel is awesome.png
2025-12-05 14:03:48      2399812 NextWork - Lelo is awesome.png
[ec2-user@ip-10-0-8-179 ~]$ █
```

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 a .txt file.

The second command I ran was 'aws s3 cp /tmp/nextwork.txt s3://nextwork-vpc project-eg' This command will upload the .txt file to my S3 bucket.

The third command I ran was 'aws s3 ls s3://nextwork-vpc-project-eg' which validated that the .txt file was successfully uploaded.

```
The user-provided path /tmp/nextwork.txt does not exist.  
[ec2-user@ip-10-0-8-179 ~]$ sudo touch /tmp/nextwork.txt  
[ec2-user@ip-10-0-8-179 ~]$ aws s3 cp /tmp/nextwork.txt s3://nimbus1-vpc-project-eg  
upload: ../../tmp/nextwork.txt to s3://nimbus1-vpc-project-eg/nextwork.txt  
[ec2-user@ip-10-0-8-179 ~]$ aws s3 ls s3://nimbus1-vpc-project-eg  
2025-12-05 14:03:45      2431554 NextWork - Denzel is awesome.png  
2025-12-05 14:03:48      2399812 NextWork - Lelo is awesome.png  
2025-12-05 14:45:19          0 nextwork.txt  
[ec2-user@ip-10-0-8-179 ~]$ █
```

In the second part of my project...

Step 5 – Set up a Gateway

In this step, I will set up a way for your VPC and S3 to communicate directly.

Step 6 – Bucket policies

In this step, I will be testing my endpoint connection by blocking off all traffic to my S3 bucket, except for traffic coming from my endpoint.

Step 7 – Update route tables

In this step, I will test the VPC endpoint set up and troubleshoot a connectivity issue.

Step 8 – Validate endpoint connection

In this step, I will test my VPC endpoint set up (again) and restrict my VPC's access to my AWS environment.

Setting up a Gateway

I set up an S3 Gateway, which is a type of endpoint used specifically for Amazon S3 and DynamoDB (DynamoDB is an AWS database service). Gateways work by simply adding a route to your VPC route table that directs traffic bound for S3 or DynamoDB to head straight for the Gateway instead of the internet.

What are endpoints?

An endpoint in AWS is a service that allows private connections between your VPC and other AWS services without needing the traffic to go over the internet.

The screenshot shows two related AWS VPC Endpoint management pages. The top page is a list view of endpoints, and the bottom page is a detailed view of a specific endpoint.

Endpoints (1/1) - Actions - Create endpoint

vpce-0ac0600189fc005c / Nimbus VPC Endpoint - Endpoints

Details

Endpoint ID	Status	Creation time	Endpoint type
vpce-0ac0600189fc005c	Available	Friday, December 5, 2025 at 10:11:54 EST	Gateway
VPC ID	Status message	Service name	Private DNS names enabled
vpc-05204117a43f265d9 (Nimbus-vpc)	-	com.amazonaws.us-east-1.s3	No
DNS record IP type	IP address type	Service region	Private DNS preference
service-defined	ipv4	us-east-1	-
Private DNS specified domains			

Bucket Policies

A bucket policy is a type of IAM policy designed for setting access permissions to an S3 bucket. Using bucket policies, you get to decide who can access the bucket and what actions they can perform with it.

My bucket policy will deny all traffic EXCEPT from the VPC.

The screenshot shows the AWS S3 Bucket Policy editor interface. On the left, there's a navigation sidebar with sections like 'Amazon S3', 'Buckets', 'Access management and security', 'Storage management and insights', and 'AWS Marketplace for S3'. The main area is titled 'Bucket policy' and contains the following JSON code:

```
1  {
2    "Version": "2012-10-17",
3    "Statement": [
4      {
5        "Effect": "Deny",
6        "Principal": "*",
7        "Action": "s3:*",
8        "Resource": [
9          "arn:aws:s3:::nimbus1-vpc-project-eg",
10         "arn:aws:s3:::nimbus1-vpc-project-eg/*"
11       ],
12       "Condition": {
13         "StringNotEquals": {
14           "aws:sourceVpce": "vpce-0ac0600189cf005c"
15         }
16       }
17     ]
18   }
19 }
```

To the right of the policy code, there are two panels: 'Edit statement' and 'Select a statement'. The 'Edit statement' panel has a button '+ Add new statement'. The 'Select a statement' panel has a placeholder 'Select an existing statement in the policy or add a new statement.'

Bucket policies

Right after saving my bucket policy, my S3 bucket page showed 'denied access' warnings. This was because my policy denies all actions unless they come from my VPC endpoint.

I also had to update my route table because my route table, by default, didn't provide a route for traffic in my public subnet to the VPC endpoint.

Block public access (bucket settings)

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more ↗](#)

Access denied

API response

```
User: arn:aws:iam::312831649097:user/erik-admin is not authorized to perform: s3:GetBucketPublicAccessBlock on resource: "arn:aws:s3:::nimbus1-vpc-project-eg" with an explicit deny in a resource-based policy
```

Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more ↗](#)

You don't have permission to get bucket policy

You or your AWS administrator must update your IAM permissions to allow s3:GetBucketPolicy. After you obtain the necessary permission, refresh the page. Learn more about [Identity and access management in Amazon S3 ↗](#)

API response

Route table updates

To update my route table, I visited the 'Endpoints' page of my VPC console and I modified the route table from there to associate my VPC's Public subnet.

After updating my public subnet's route table, my terminal could return my S3 bucket files that were uploaded.

The screenshot shows the AWS VPC Route Table configuration for a specific subnet. The top navigation bar displays the subnet ID and name: `subnet-046a3a01fa1f63cb8 / Nimbus-subnet-public1-us-east-1a`. Below this, the route table ID is listed as `rtb-0e687a33f1bec6144 / Nimbus-rtb-public`. On the right side of the screen, there is a button labeled `Edit route table association`. The main content area is titled `Routes (3)` and contains a table with three entries:

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	igw-0e7676f75358d6bb6
pl-63a5400a	vpce-0ac0600189fc005c

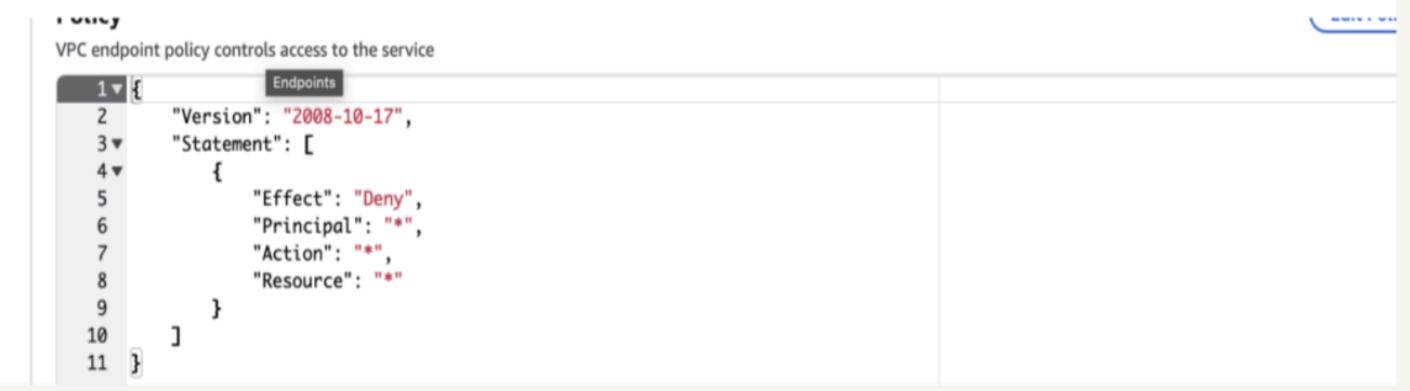
A search bar labeled `Filter routes` is located above the table. Navigation controls for the list (page 1 of 1) and a settings gear icon are also present.

Endpoint policies

An endpoint policy is a type of policy designed for specifying the range of resources and actions permitted by an 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 again denied access to S3 when I tried to run another AWS S3 command.

vpce-0ac0600189fc005c / Nimbus VPC Endpoint



The screenshot shows the AWS Lambda function editor with the title "vpce-0ac0600189fc005c / Nimbus VPC Endpoint". Below the title, it says "VPC endpoint policy controls access to the service". The policy code is displayed in a code editor:

```
1 {  
2     "Version": "2008-10-17",  
3     "Statement": [  
4         {  
5             "Effect": "Deny",  
6             "Principal": "*",  
7             "Action": "*",  
8             "Resource": "*"  
9         }  
10    ]  
11 }
```