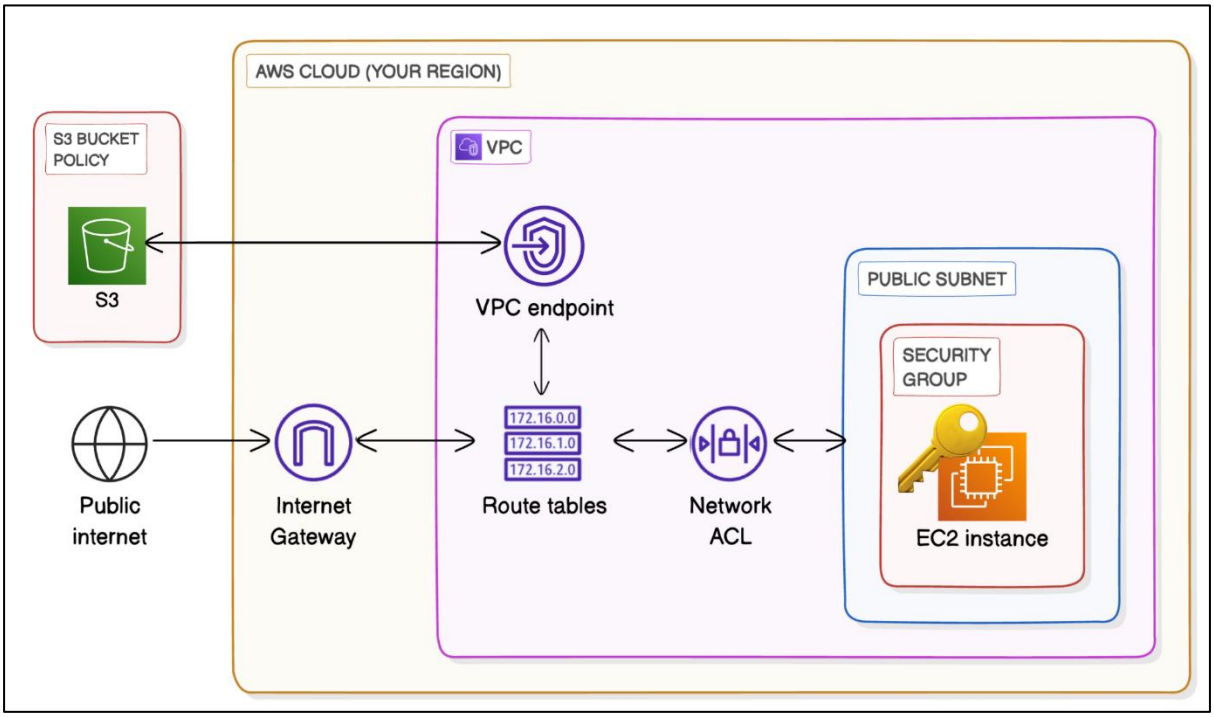
VPC Endpoints

VPC endpoints gives your VPC **private, direct access to other AWS services like S3**, so traffic doesn't need to go through the internet.

Just like how internet gateways are like your VPC's door to the internet, you can think of VPC endpoints as **private doors to specific AWS services.**

Get ready to:

1. Use VPC endpoints to directly access your S3 bucket from your VPC.
2. Test your endpoint setup using an S3 security tool called bucket policies!



Set up your Architecture

In this step, you're going to:

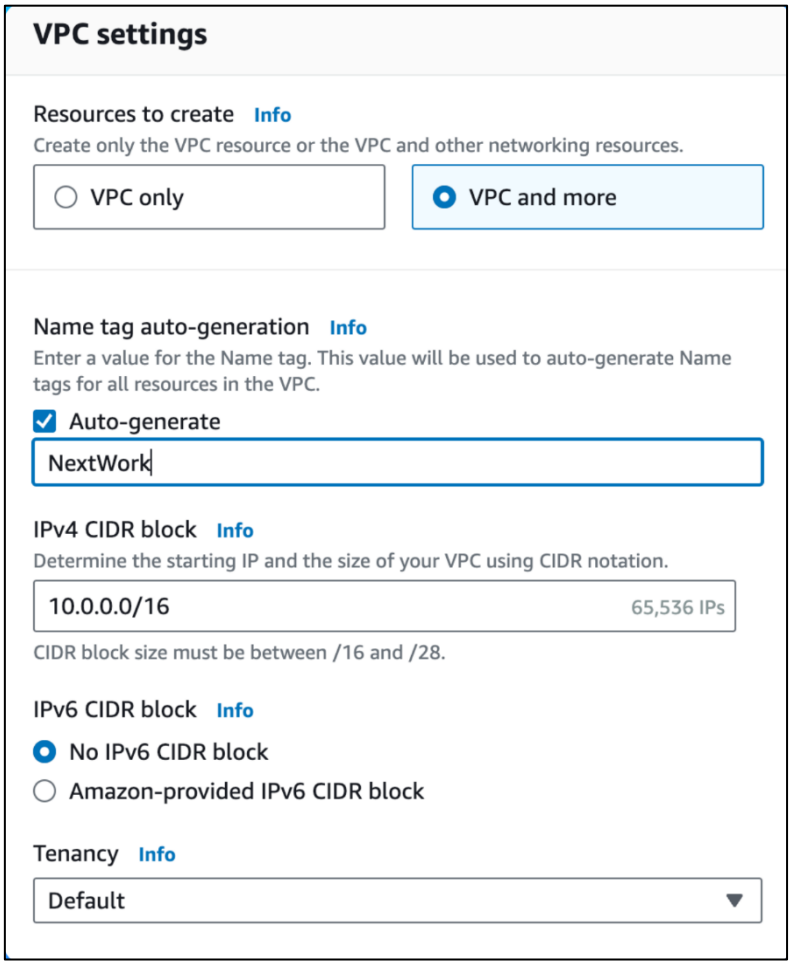
1. Create a VPC from scratch!
2. Launch an EC2 instance, which you'll connect to using EC2 Instance Connect later.
3. Set up an S3 bucket.

- Log in to your AWS Account.

Create Your VPC

- Head to your **VPC** console - search for VPC at the search bar at top of your page.
- From the left hand navigation bar, select **Your VPCs.**
- Select **Create VPC.**

- Select **VPC and more.**
- Under **Name tag auto-generation**, enter NextWork
- The VPC's **IPv4 CIDR block** is already pre-filled to 10.0.0.0/16 - we'll use this default CIDR block!



- For **IPv6 CIDR block**, we'll leave in the default option of **No IPv6 CIDR block.**
- For **Tenancy**, we'll keep the selection of **Default.**
- For **Number of Availability Zones (AZs)**, we'll use just **1** Availability Zone.
- Make sure the **Number of public subnets** chosen is **1**.
- For **Number of private subnets,** we'll keep thing simple today and go with **0** private subnets.
- For the **NAT gateways ($)** option, make sure you've selected **None.** As the dollar sign suggests, NAT gateways cost money!
- For the **VPC endpoints** option, select **None.**

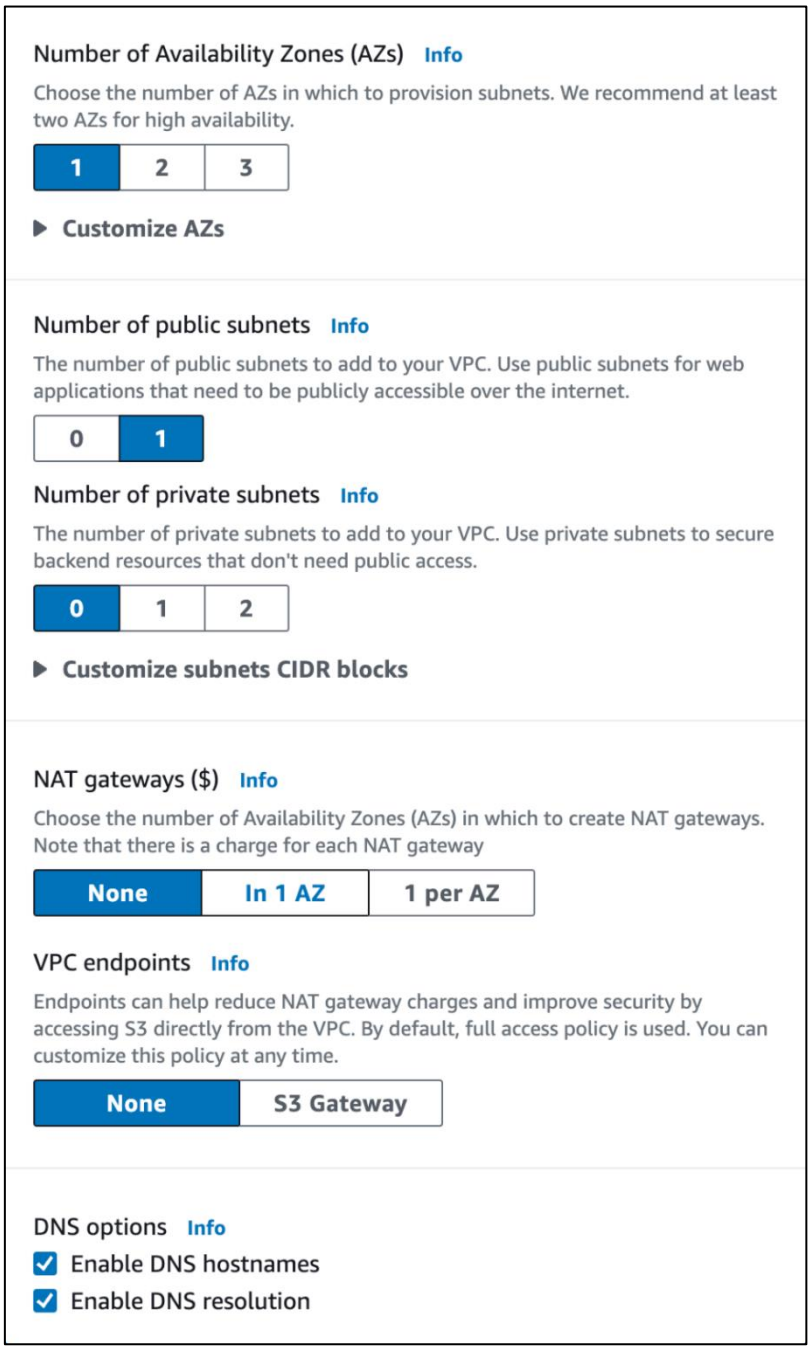
What are VPC endpoints?

Normally, to access some AWS services like S3 from your VPC, your traffic would go out to the public internet.

But, VPC endpoints let you connect your VPC privately to AWS services without using the public internet. This means your data stays within the AWS network, which can improve security and reduce data transfer costs.

We're selecting None here, so we can learn to set one up manually later!

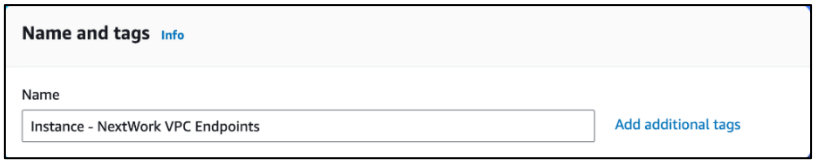
- You can leave the **DNS options** checked.



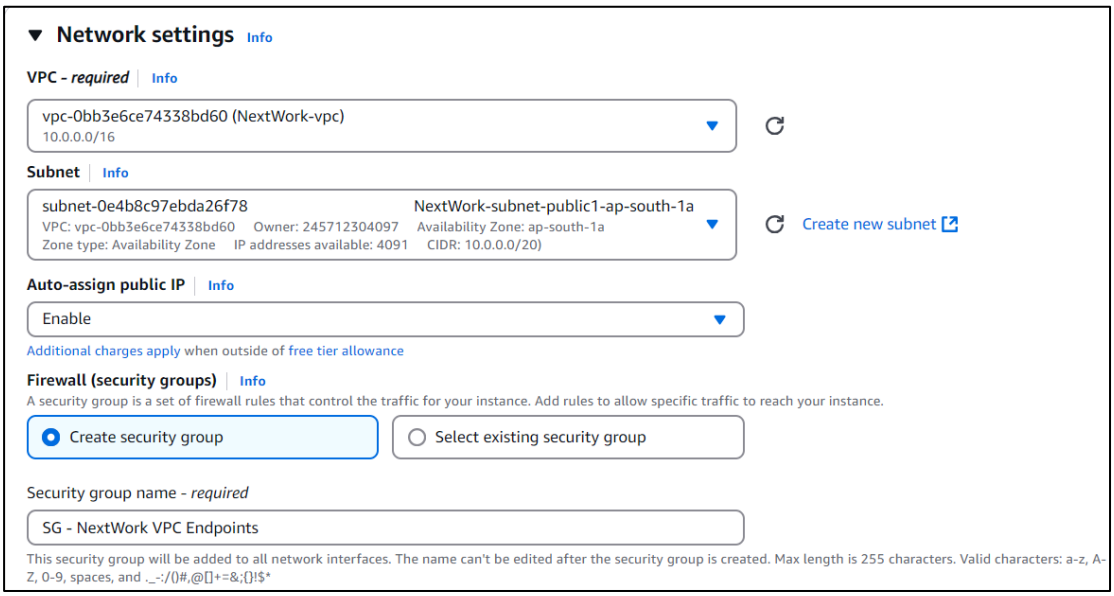
- Select **Create VPC.**

Launch an instance in your VPC

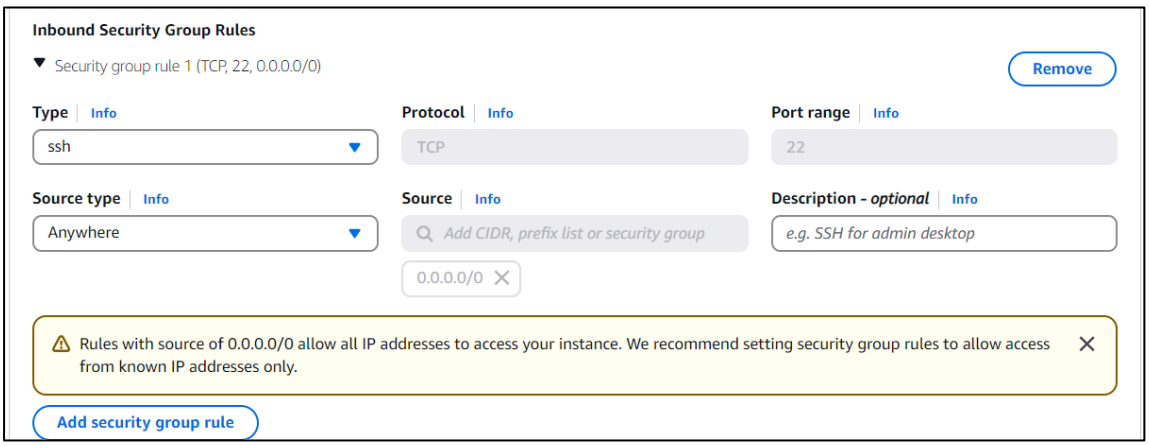
- Head to the **EC2 console** - search for EC2 in the search bar at the top of screen.
- Select **Instances** at the left hand navigation bar.
- Select **Launch instances**.
- Since your first EC2 instance will be launched in your first VPC, let's name it Instance - NextWork VPC Endpoints



- For the **Amazon Machine Image,** select **Amazon Linux 2023 AMI.**
- For the **Instance type,** select **t2.micro.**
- For the **Key pair (login)** panel, select **Proceed without a key pair (not recommended).**
- At the **Network settings** panel, select **Edit** at the right hand corner.
- Under **VPC**, select **NextWork-vpc**.
- Under **Subnet,** select your VPC's public subnet.
- Keep the **Auto-assign public IP**
- Select **Enable.**
- For the **Firewall (security groups)** setting, choose **Create security group.**
- Name your security group SG - NextWork VPC Endpoints



- That's it! No security group rules to add.

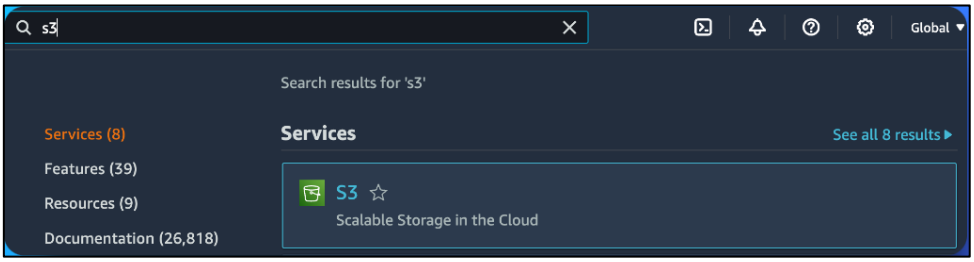


- Select **Launch instance.**

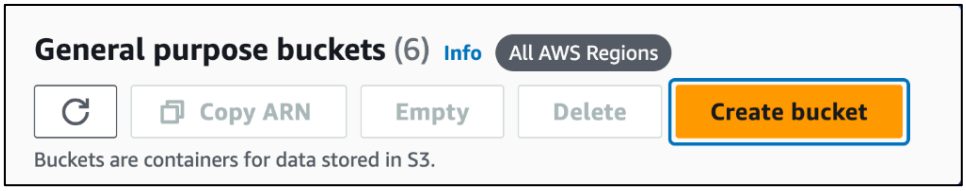
Launch a bucket in Amazon S3

Before we connect with our EC2 instance, let's set up a bucket in Amazon S3
After creating this bucket, we'll access it from our EC2 instance and do things like checking what objects are in the bucket.

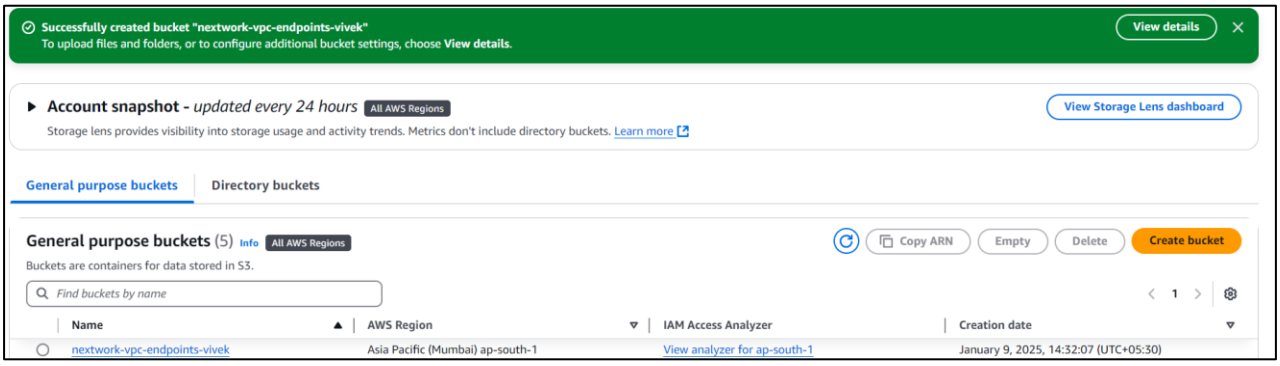
- Search for **S3** at the search bar at the top of your console.



- Make sure you're in the same Region as your NextWork VPC!
- Select **Create bucket.**

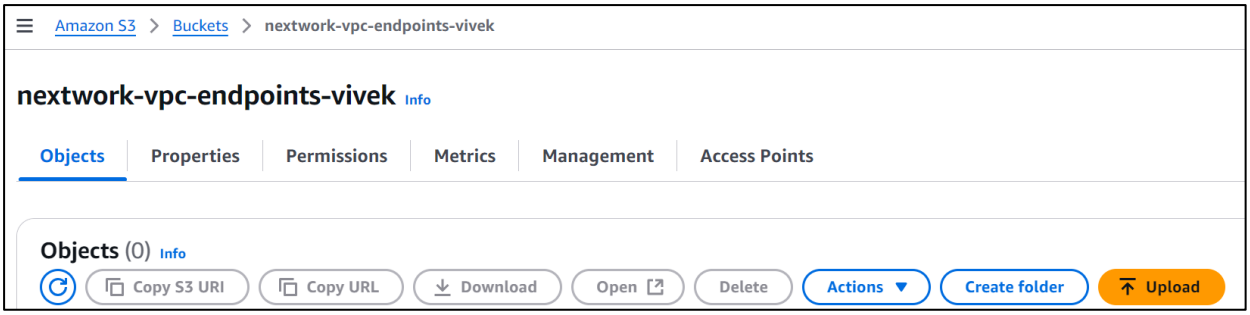


- Let's set up your bucket! Keep the **Bucket type** as **General purpose.**
- Your **bucket name** should be nextwork-vpc-endpoints-yourname
  - Don't forget to replace **yourname** with your first name! S3 bucket names need to be globally unique. In my case: nextwork-vpc-endpoints-vivek
- We'll leave all other default settings, go straight to **Create bucket**.

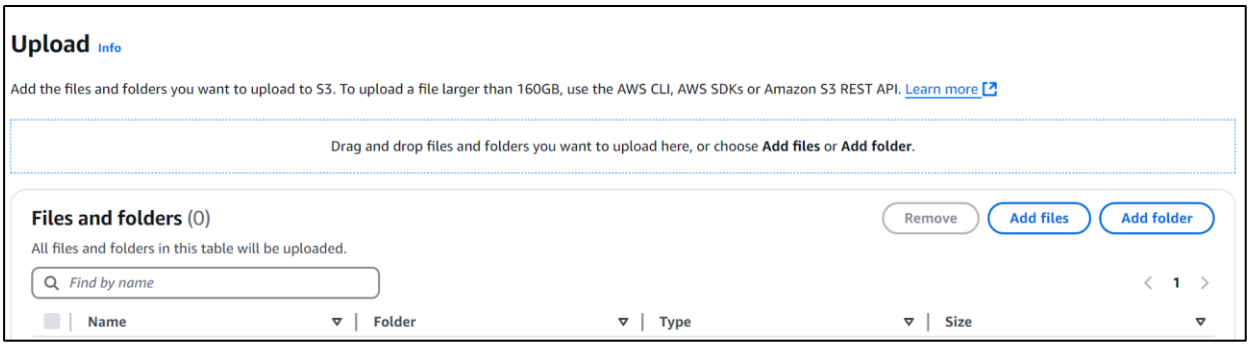


We'll finish set up by uploading two files into your shiny new bucket.
This can be **any** two files in your local computer!

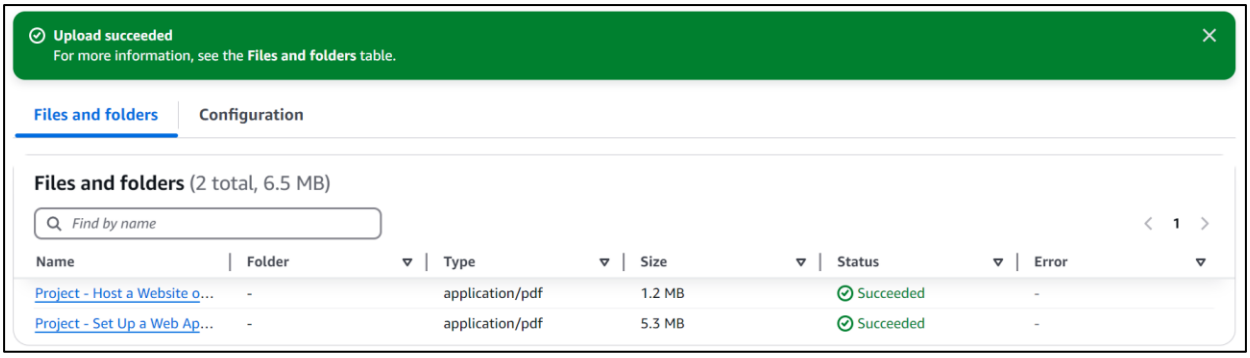
- Click into your bucket.
- Select **Upload**.



- Select **Add files** in the **Files and folders panel.**



- Select two files in your local computer to upload.
- Your files should show up in your **Files and folders** panel once they're uploaded!



Connect to Your EC2 Instance

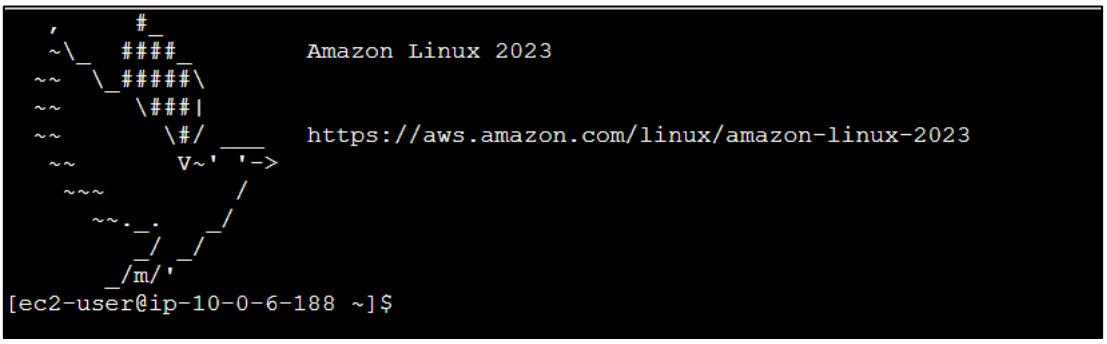
This project is all about VPC endpoint connections, so before we create that endpoint... let's see what things are like **without** endpoint connections in place.
In this step, we're gonna connect to your EC2 instance and try access S3 through the **public internet!**

In this step, you're going to:

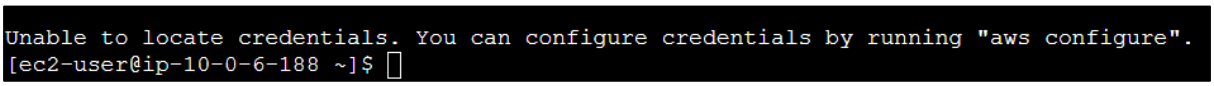
1. Connect directly to your EC2 instance.

- Head to your **EC2** console and the **Instances** page.
- Select the checkbox next to **Instance - NextWork VPC 1.**
- Select **Connect**.

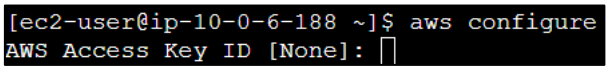
- In the EC2 Instance Connect set up page, select **Connect** again.



- Yay! Successfully connected.
- For your first command, try running aws s3 ls



- We need to provide credentials.
- Run aws configure



- The terminal is now asking us for an Access Key ID!

What's the difference between access keys and key pairs?
You might remember key pairs from launching EC2 instances!

- **Key pairs** are used specifically for logging into your EC2 instances through SSH.
- **Access keys**, which are what we're learning about now, are credentials for your applications and other servers to log into AWS and talk to your AWS services/resources.

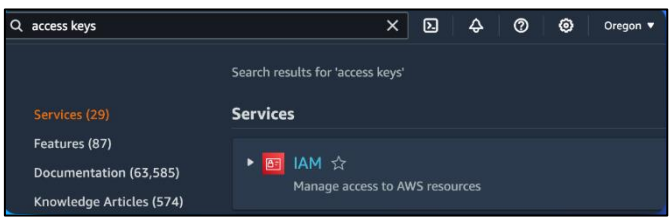
Create Access Keys

Your EC2 instance needs credentials to access your AWS services, so let's set up access keys right away

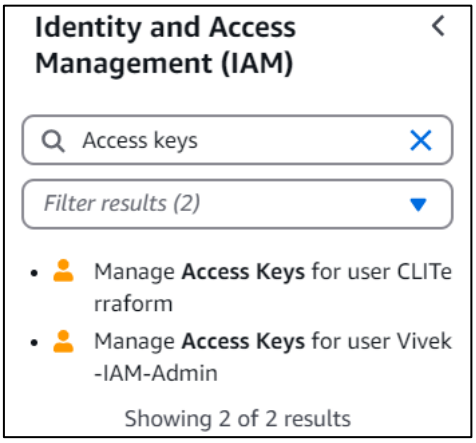
In this step, you're going to:

1. Give your EC2 instance access to your AWS environment.

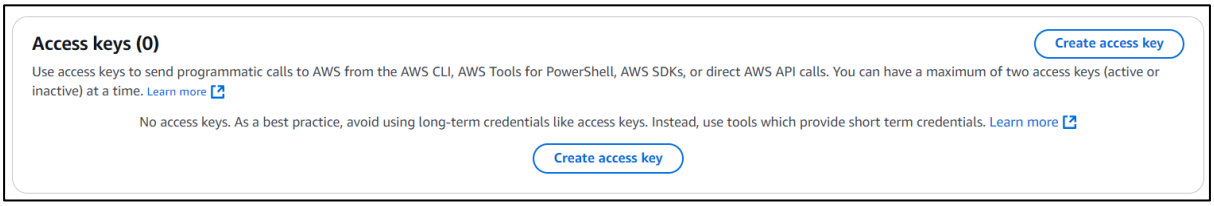
- Search for Access keys in the search bar.



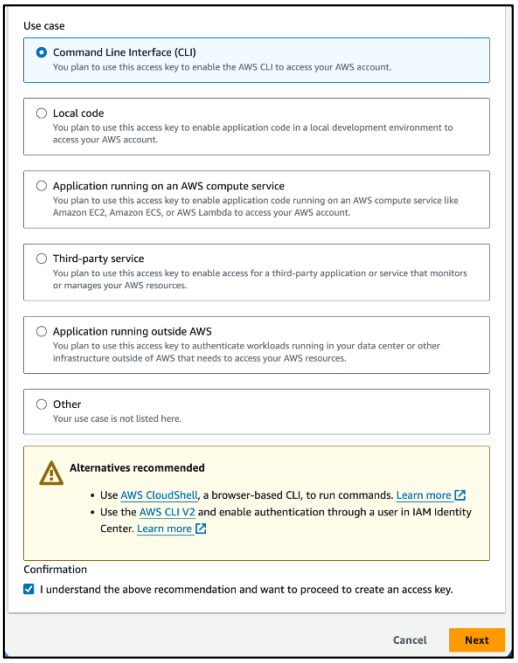
- it's the IAM console that helps us set this up. Select **IAM.**
- Search for Access keys again in the IAM console's left hand navigation panel.



- Choose the search result for managing an access key for your IAM Admin account. (in my case: user Vivek-IAM-Admin)
- Scroll Down and we find Access keys, Choose **Create access key** on the right-hand side.



- On the first set up page, select **Command Line Interface (CLI).**
- Select the checkbox that says **I understand the above recommendation and want to proceed to create an access key.**
- Select **Create access key.**



What is the yellow warning banner saying?

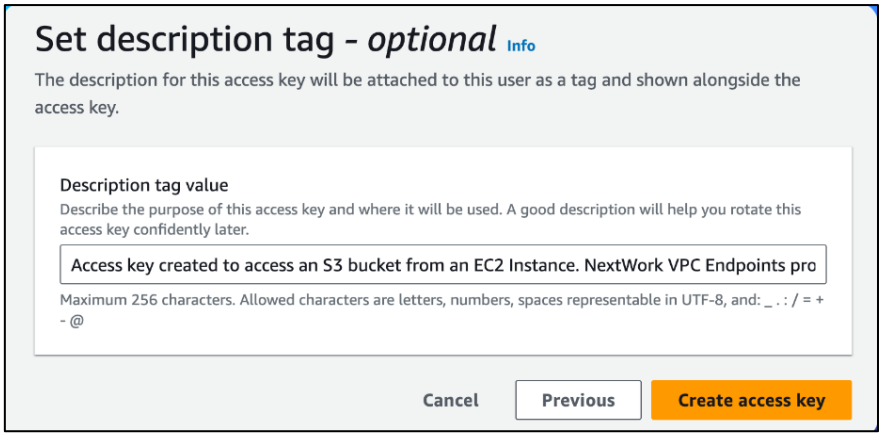
In this project we're creating access keys and manually applying them in our EC2 instance, but typically the recommended way is to create an **IAM role** with the necessary permissions and then **attaching** that role to your EC2 instance.
Your EC2 instance would inherit the permissions from the role, and this is best practise as you can easily attach and detach EC2 instances from roles to give and take away their credentials.
We aren't using this method so that we can learn about access keys, but roles are usually a better alternative for security.

Extra for Experts: What do the other options mean?

Let's break 'em down!

- Choose **Local code** if you're building an application in your local computer, and need the access key to connect your app with AWS services.
- Choose **Application running on an AWS compute service** if you're building an application that's running on an AWS compute service like EC2 or Lambda, and need the access key to connect your app with AWS services.
- Choose **Third-party service** if you're running an application from an external company (i.e. it wasn't built by yourself or AWS), but requires access to your AWS environment. For example, Crowdstrike is a third party security service that requires access to your AWS environment to protect them in real time!
- Choose **Application running outside AWS** if you're running an application built by yourself but hosted on a physical server or another cloud platform, and you need to integrate it with AWS services.

- Select **Next.**
- For the **Description tag value,** we'll write Access key created to access an S3 bucket from an EC2 Instance. NextWork VPC Endpoints project.



- Select **Create access key.**
- This is important! **STAY ON THIS PAGE**
  You've just created your access key, well done.
  This is the only time that your secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.
- Select **Download .csv file** near the bottom of the page.

What is the secret access key?

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

Secret is a key word here - anyone who has it can access your AWS account, so we need to keep this away from anyone else!

Connect to your S3 bucket

Access keys created

We shall try use our EC2 instance to access your S3 bucket now

In this step, you're going to:

1. Head back to your EC2 instance.
2. Get your EC2 instance to access your S3 bucket.

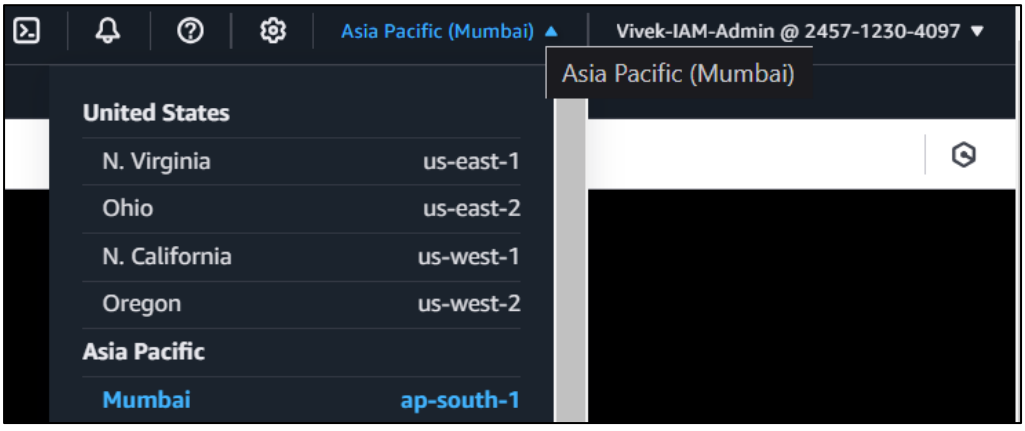
- Now back to your **EC2 Instance Connect** tab!
- Are you ready to enter your **Access keys details now?**
- Open the **AccessKeys.csv** file you've downloaded.
- Copy the **Access key ID.**
- Paste this into your EC2 instance's terminal, and press **Enter** on your keyboard.

```
[ec2-user@ip-10-0-6-188 ~]$ aws configure
AWS Access Key ID [None]: AKIATSNMYG7QVOXJNMNQ
AWS Secret Access Key [None]:
```

- Let's do the same for the **AWS Secret Access Key!**
- Copy the key from your .csv file, and paste it in the terminal. Press **Enter**.

```
[ec2-user@ip-10-0-6-188 ~]$ aws configure
AWS Access Key ID [None]: AKIATSNMYG7QVOXJNMNQ
AWS Secret Access Key [None]: xJ8UJXd/9VI53CJNzvK+sB6YirSQ93XLOUflsDvJ
Default region name [None]:
```

- Next, for your **Default region name,** open your **Region** dropdown on the top right hand corner of your AWS console.



- Copy your **Region's code name.** In my case: ap-south-1
- Paste that in your EC2 instance's terminal. Press **Enter** on your keyboard.

```
[ec2-user@ip-10-0-6-188 ~]$ aws configure
AWS Access Key ID [None]: AKIATSNMYG7QVOXJNMNQ
AWS Secret Access Key [None]: xJ8UJXd/9VI53CJNzvK+sB6YirSQ93XLOUflsDvJ
Default region name [None]: ap-south-1
Default output format [None]:
```

- We don't have a default output format, so we can leave that empty. Press **Enter**.

What is a default output format?

In just a second, you'll be running a few CLI commands in your EC2 instance's terminal to use Amazon S3. The **default output format** is how the results of your will display!
Options for your output format options are...

- json, which is the default if you're not using another option.
- text for plain text.
- table for an formatted table.
- yaml, which is a format for writing data in a way that is easy for people to read and write.

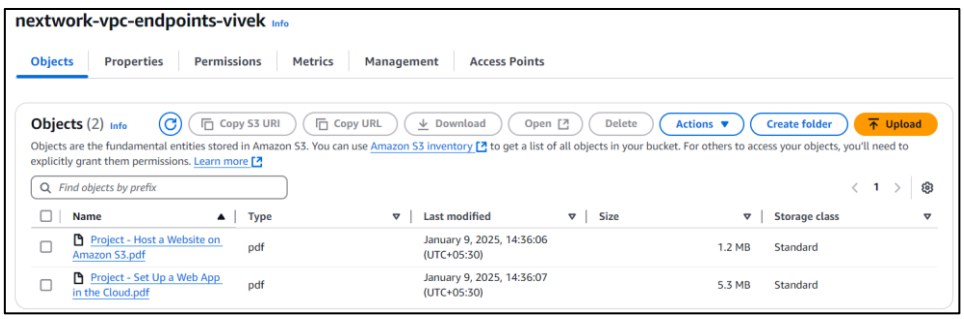
- Set up is done.
- Run the aws s3 ls command again to see a list of your account's S3 buckets.

```
[ec2-user@ip-10-0-6-188 ~]$ aws s3 ls
2025-01-09 09:02:08 nextwork-vpc-endpoints-vivek
2025-01-03 07:59:27 nextwork-website-project-vivek
2024-11-25 12:46:28 practicebin
2024-12-03 05:34:29 terraform-state-bucket-projectb114
2024-11-10 12:40:37 vivek114
[ec2-user@ip-10-0-6-188 ~]$
```

- We can now see the buckets specifically nextwork-vpc-endpoints-vivek (in mycase) which we created for our project.
- Next, let's run the command aws s3 ls s3://nextwork-vpc-endpoints-yourname. Make sure to replace nextwork-vpc-endpoints-yourname with your actual bucket name!

```
[ec2-user@ip-10-0-6-188 ~]$ aws s3 ls s3://nextwork-vpc-endpoints-vivek
2025-01-09 09:06:06    1299232 Project - Host a Website on Amazon S3.pdf
2025-01-09 09:06:07    5549573 Project - Set Up a Web App in the Cloud.pdf
[ec2-user@ip-10-0-6-188 ~]$
```

- we can even see the objects that are inside your bucket.



- you could upload files using AWS CLI too.
  - Run sudo touch /tmp/nextwork.txt to create a blank .txt file in your EC2 instance.

What does this command say?

Let's break it down!

- **sudo** stands for "superuser do" and it's used when you're running a command that typicaly needs elevated user permissions, like installing software or creating new files.
- **touch** is a standard command used to create an empty file if it doesn't exist. If it does exist, this command will update the file's timestamp (i.e. the last time it was accessed) instead.
- **/tmp/** is a common directory path (you can think of a directory path as layers and layers of folders) typically used for temporary files.
- **nextwork.txt** is the name of the empty file you're creating!

- Next, run aws s3 cp /tmp/nextwork.txt s3://nextwork-vpc-endpoints-yourname to upload that file into your bucket.

What does this command say?

Let's break it down!

- **aws s3 c:** is the command to **copy** files.
- **/tmp/nextwork.txt** is the **source file path.** It's saying that the file to copy is nextwork.txt, which exists inside a folder called tmp.
- **s3://nextwork-vpc-endpoints-yourname** is the **destination path**. It's saying that the file should be copied to the S3 bucket vpc-project-yourname!

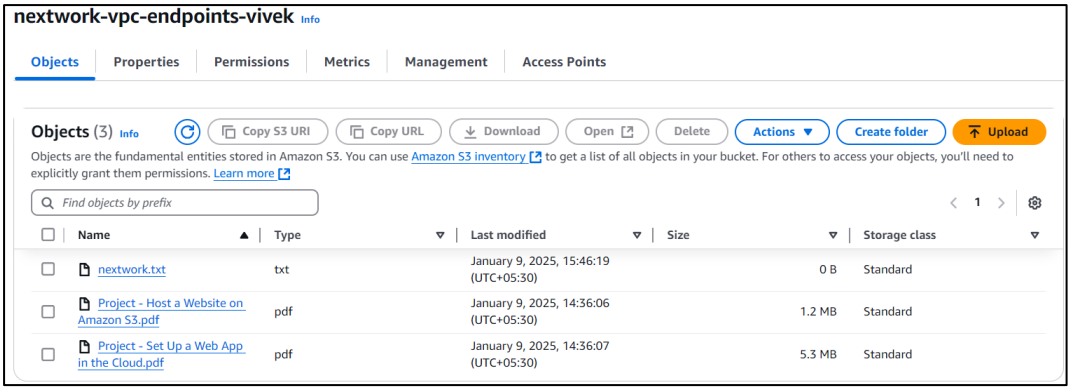
- Finally, run aws s3 ls s3://nextwork-vpc-endpoints-yourname again and observe what objects are listed in your bucket now.

```
[ec2-user@ip-10-0-6-188 ~]$ aws s3 ls s3://nextwork-vpc-endpoints-vivek
2025-01-09 09:06:06    1299232 Project - Host a Website on Amazon S3.pdf
2025-01-09 09:06:07    5549573 Project - Set Up a Web App in the Cloud.pdf
2025-01-09 10:16:19          0 nextwork.txt
```

What is the 0 next to nextwork.txt?

The numbers next to each file name is the size fo that file! Since nextwork.txt is an empty file with no data, it has 0 bytes.

- Switch tabs back to your **S3** console.
- Refresh the **Objects** tab for your S3 bucket.
- You see also see your new file there.



- A file you've created in your EC2 instance now lives in your S3 bucket too!

Create an endpoint

Your EC2 instance definitely has access to your bucket - your access key worked.

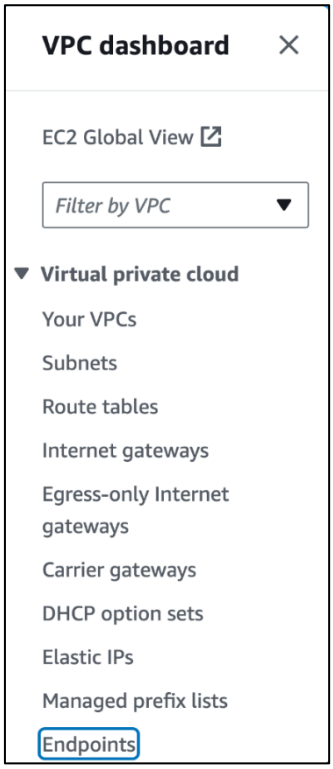
But your instance is connected to your bucket through the public internet.

This isn't the most secure way to communicate - external threats and attacks can easily intercept your commands and get access to your AWS environment or sensitive data.

lt's time we bring in VPC endpoints! **In this step, you're going to:**

1. Set up a way for your VPC and S3 to communicate direclty.

    - **In a new tab,** head back to your **VPC** console.
    - Select **Endpoints** from the left hand navigation panel.



- Select **Create endpoint.**

What's an endpoint?

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.

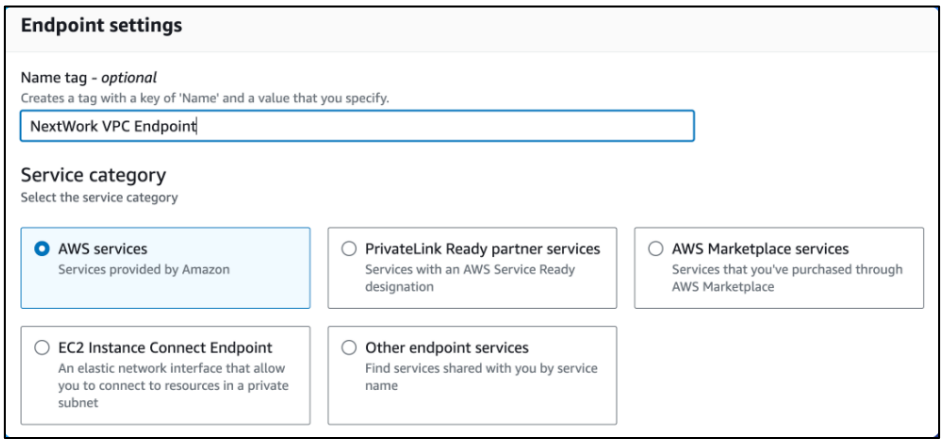
I thought all of my resources get launched inside my VPC, so how are some connections through the internet?

Not all AWS resources get launched inside your VPC! While your EC2 instances live within your VPC, S3 buckets and some other AWS services exist outside your VPC because they're designed to be highly available and accessible from anywhere.

For example, the commands you're putting through AWS CLI now will be communicated to your S3 bucket through the public internet.

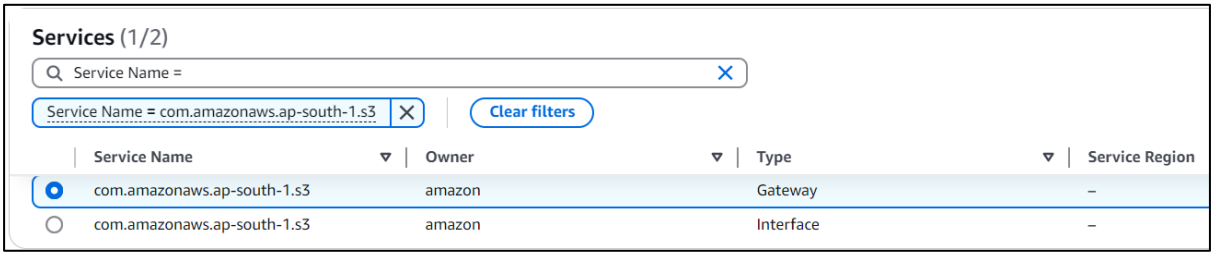
That's why VPC endpoints exist to create a **private** connection between your VPC and AWS services. Having a VPC endpoint means your instances can now access services like S3 directly without routing through the public internet, which makes sure your data stays within the AWS network for security.

- For the **Name tag**, let's use NextWork VPC Endpoint
- Keep the **Service category** as **AWS services.**



- In the **Services** panel, search for S3.
- Select the filter result that just ends with **s3.**

- Select the row with the **Type** set to **Gateway**.

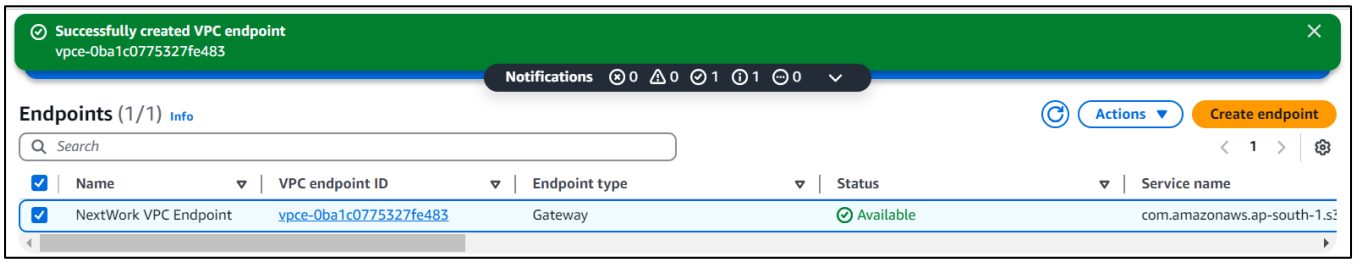


What's does Gateway mean?

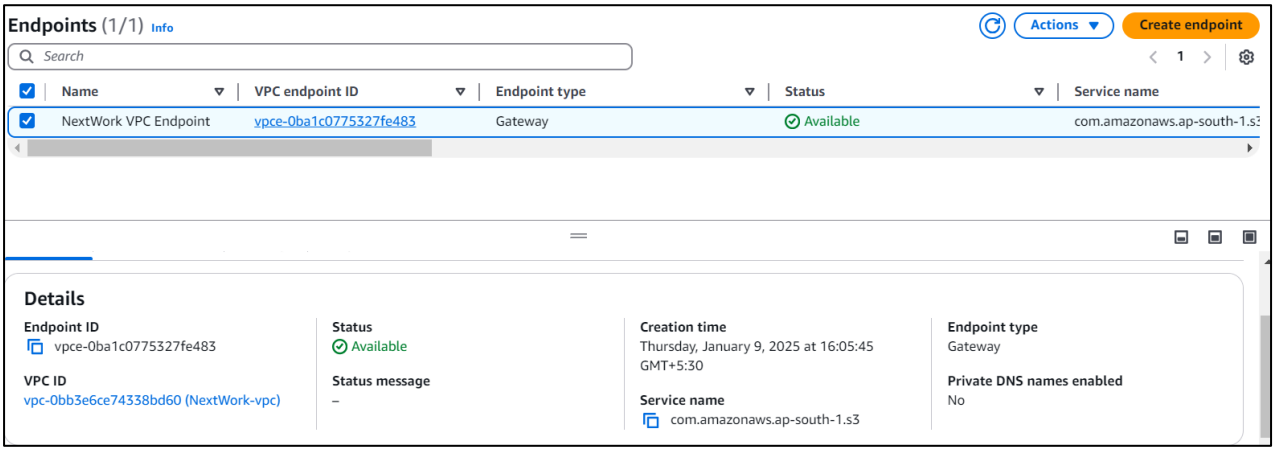
A **Gateway** is a type of endpoint used specifically for Amazon S3 and DynamoDB (DynamoDB is an AWS database service).
Gateways work by simply adding a route to your VPC route table that directs traffic bound for S3 or DynamoDB to head straight for the Gateway instead of the internet.

Extra for Experts: As AWS evolved and more companies started using it for diverse and complex tasks, there was a need for more detailed control over how data moves within AWS networks. Gateway endpoints they could manage traffic but didn't offer detailed settings - this led to AWS creating Interface endpoints, which offer more security settings.

- Next, at the **VPC** panel, select **NextWork-vpc.**
- We'll leave the remaining default values and select **Create endpoint.**
- Endpoint created



- Select the checkbox next to your endpoint's name.



Create a super secure bucket policy

You've just set up a VPC endpoint.
You might be wondering:

- How do I know that my endpoint is truly providing direct access to S3?
- How can I confirm that my VPC isn't still communicating with S3 through the public internet?
To answer these questions, we'll validate our endpoint connection by doing the following:

We're going to block off your S3 bucket from ALL traffic... except traffic coming from the endpoint.

It's the ultimate test - if your endpoint was truly set up properly, then your EC2 instance should still be able to access S3. Otherwise, your instance's access is blocked!

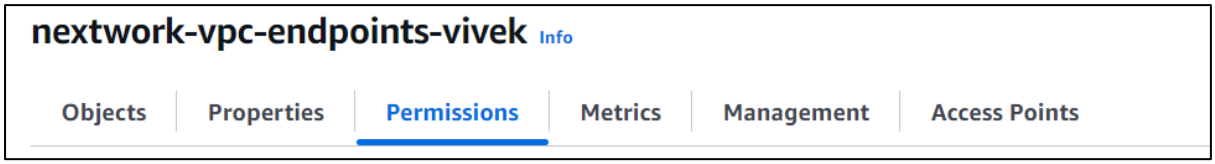
In this step, you're going to:

1. Limit your S3 bucket access's to only traffic from your endpoint.

In your S3 bucket, we're going make access super secure.

we want to make this bucket completely private to ALL access... **except** access through the VPC endpoint you've set up.

- In the **S3** console, select **Buckets** from the left hand navigation panel.
- Click into your bucket nextwork-**vpc-endpoints-yourname**. **(in my case:** nextwork-vpc-endpoints-vivek)
- Select the **Permissions** tab.



- Scroll to the **Bucket policy** panel, and select **Edit**.

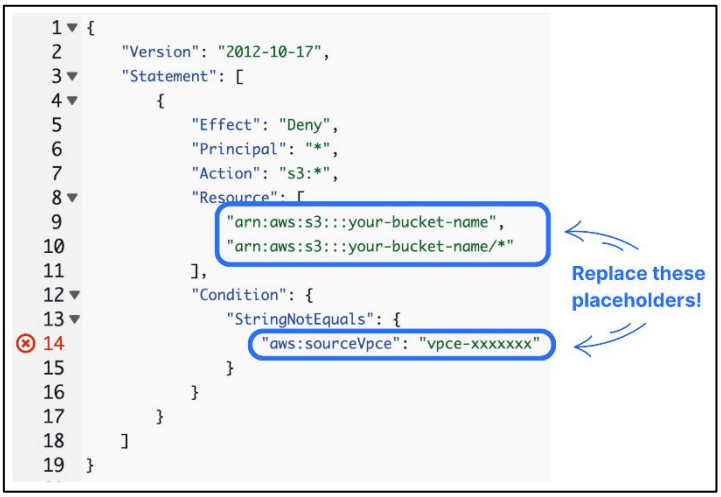
What's a bucket policy?
A bucket policy is a type of IAM policy designed for setting access permissions to an S3 bucket. Using bucket policies, you get to decide who can access the bucket and what actions they can perform with it.

- Add the below bucket policy to the policy editor.

```
{
 "Version": "2012-10-17",
 "Statement": [
   {
    "Effect": "Deny",
    "Principal": "*",
    "Action": "s3:*",
    "Resource": [
     "arn:aws:s3:::your-bucket-name",
     "arn:aws:s3:::your-bucket-name/*"
    ],
    "Condition": {
     "StringNotEquals": {
      "aws:sourceVpce": "vpce-xxxxxxx"
     }
    }
   }
 ]
}
```

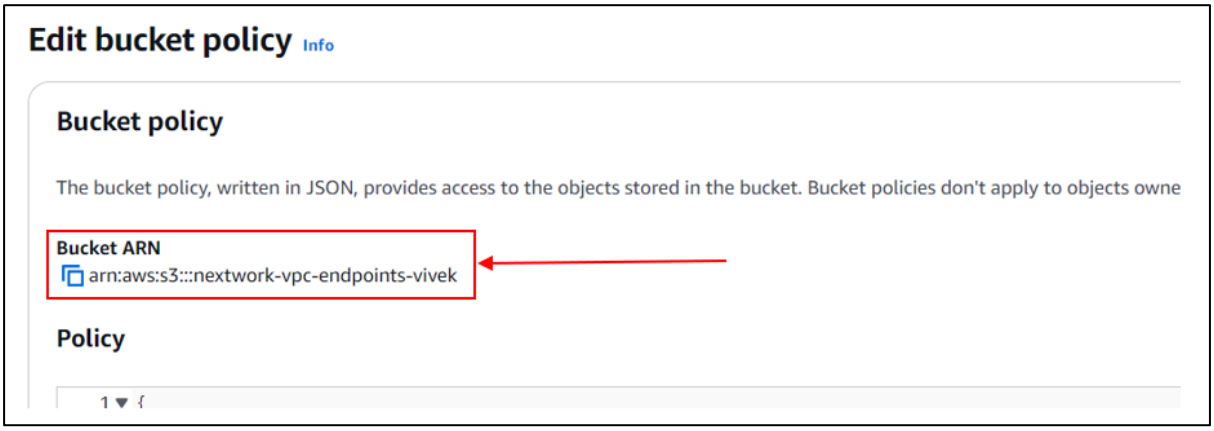
What does this policy do?
This policy **denies** all actions (s3:*) on your S3 bucket and its objects to everyone (Principal: "*")... unless the access is from the VPC endpoint with the ID defined in aws:sourceVpce.
In other words, only traffic coming from your VPC endpoint can get any access to your S3 bucket!

Before you save this policy – we have to do some changes to the script

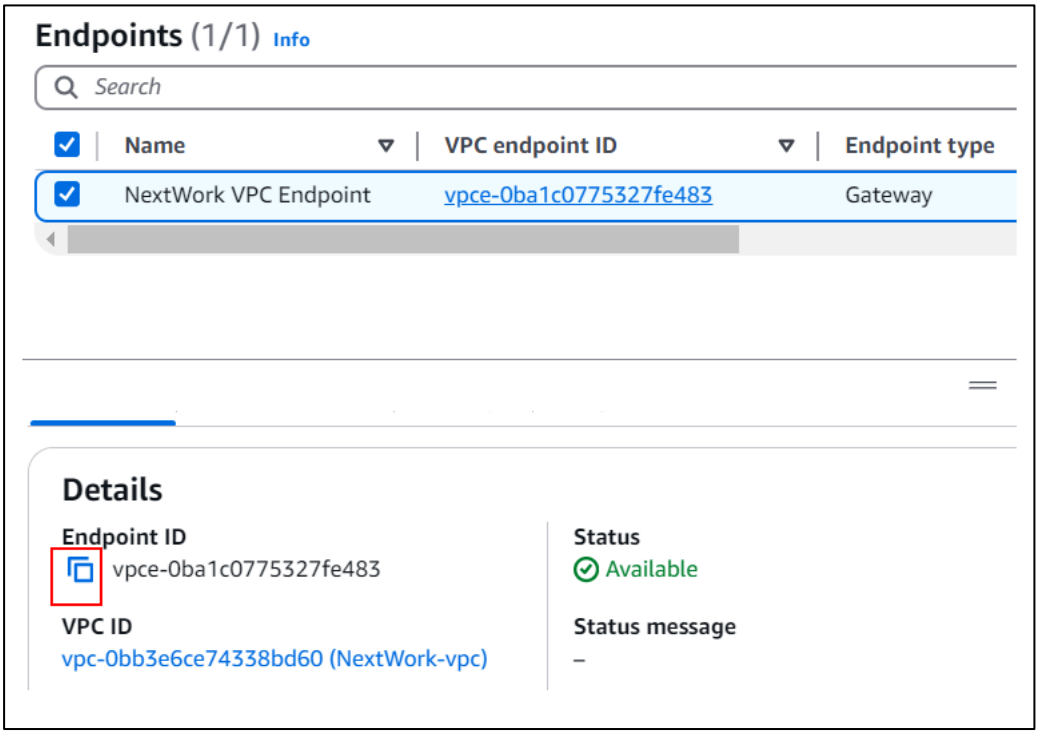


Don't forget to replace:

1. BOTH instances of **arn:aws:s3:::your-bucket-name** with your actual bucket ARN.
    - Handy tip: you can find your Bucket ARN right above the Policy window



- Replace **vpce-xxxxxxx** with your VPC endpoint's ID.
    - To find this ID, you'll have to switch back to your **Endpoints** tab and copy your endpoint's ID. Make sure it starts with vpce-

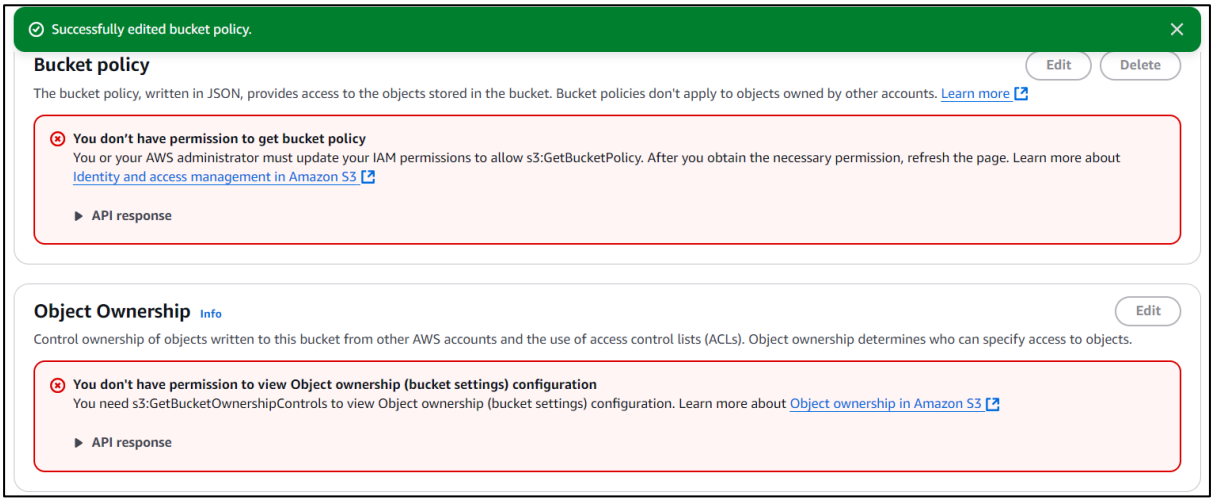


- After replacing the default values with your resources' ARN or ID, double check that you haven't deleted the quote marks around any of your statements. Also check that you still have /* at the second Resource line!
- Your code will look as follows:

Policy

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::nextwork-vpc-endpoints-vivek",
        "arn:aws:s3:::nextwork-vpc-endpoints-vivek/*"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:sourceVpce": "vpce-0ba1c0775327fe483"
        }
      }
    }
  ]
}
```

- Select **Save changes.**
- Once you've saved your changes, panels have turned red all across the screen.



- **Your policy denies all actions unless they come from your VPC endpoint. This means any attempt to access your bucket from other sources, including the AWS Management Console, is blocked!**

Access your S3 bucket Again!

Now that your S3 bucket's access is only limited to your endpoint, can your EC2 instance still access your bucket?

✅ If it can, you've set up your endpoint connection perfectly.

❌ If it can't, something's gone wrong with our VPC endpoint set up...

In this step, you're going to:

1. Test your VPC endpoint set up.
2. Troubleshoot a connectivity issue.

- Head back to **EC2 Instance Connect.**
- Try running aws s3 ls s3://nextwork-vpc-endpoints-yourname **(in my case:** nextwork-vpc-endpoints-vivek) again.

```
[ec2-user@ip-10-0-6-188 ~]$ aws s3 ls s3://nextwork-vpc-endpoints-vivek

An error occurred (AccessDenied) when calling the ListObjectsV2 operation: User: arn:aws:iam::2457123040
n resource: "arn:aws:s3:::nextwork-vpc-endpoints-vivek" with an explicit deny in a resource-based policy
[ec2-user@ip-10-0-6-188 ~]$
```

- Access denied. It looks like the bucket policy had stopped us.

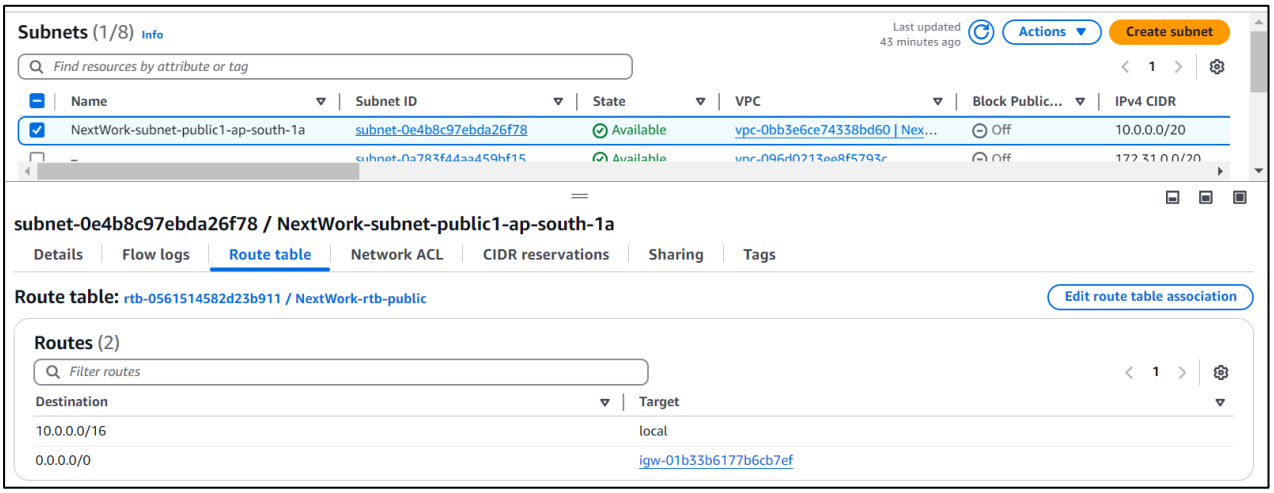
Wait a second... why is our EC2 instance also denied access?

We're supposed to get exclusive access to this S3 bucket... after all, the policy denies everyone except traffic through the endpoint!

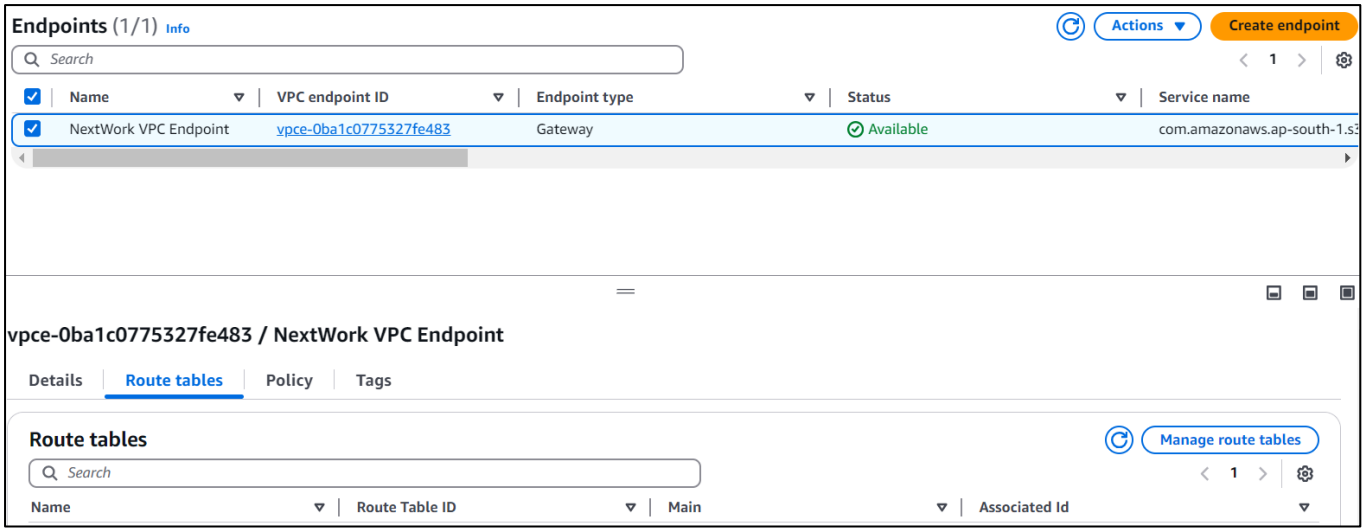
We did set up an endpoint already between your VPC and your S3, didn't we?

Troubleshoot this error by heading to your VPC console.

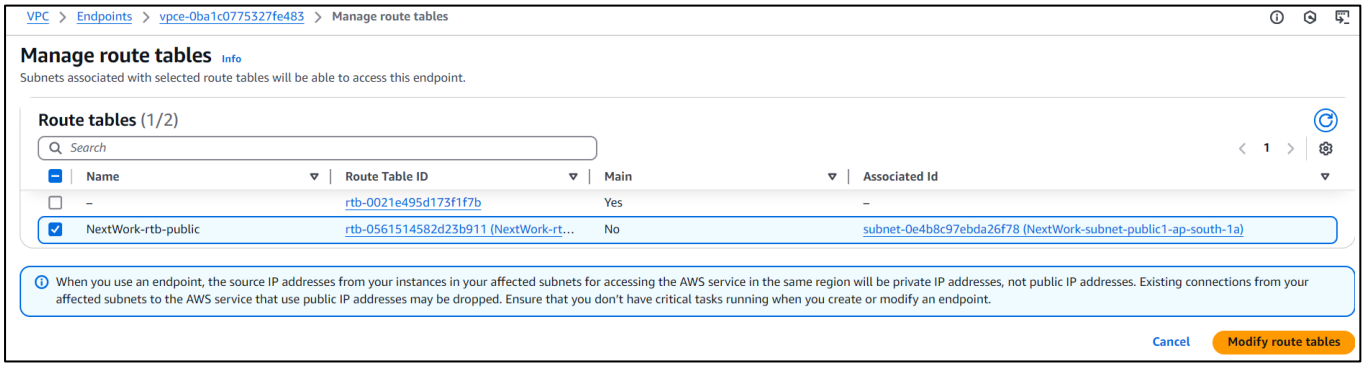
- Select **Subnets** from the left hand navigation panel.
- Select your public subnet, which starts with **NextWork-subnet-public1.**
- Select the **Route table** tab.



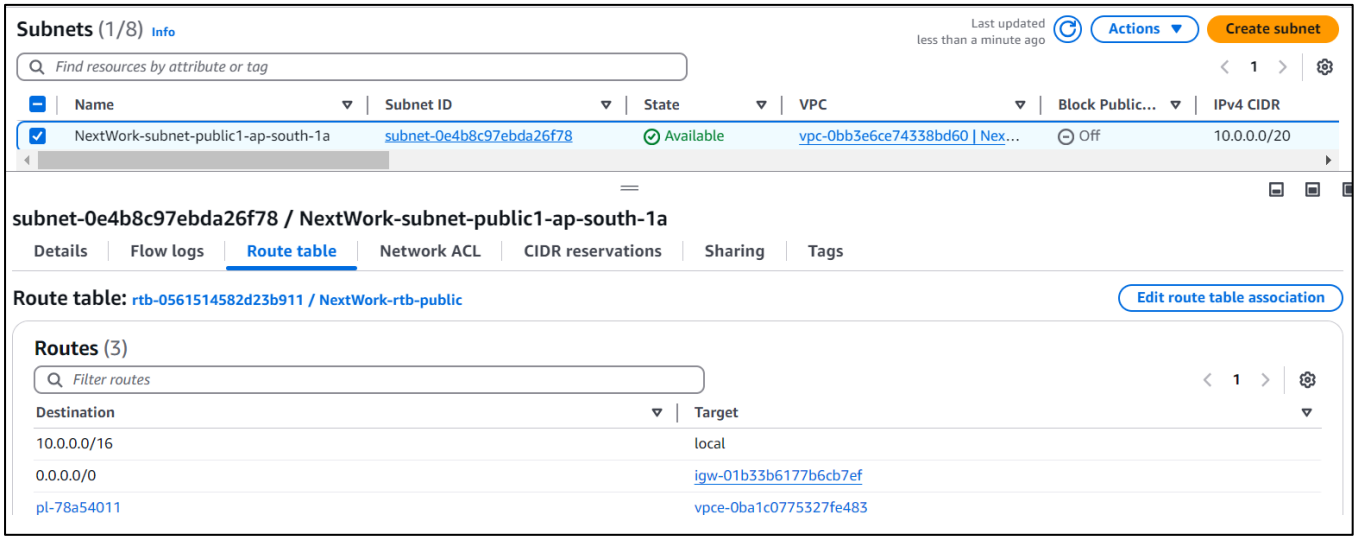
- Let's make sure your public subnet has a route to your endpoint(as we can see clearly that no route to our end point).
- Select **Endpoints** from the left hand navigation panel.
- Select the checkbox next to your endpoint, and select **Route tables.**
- Select **Manage route tables.**



- Select the checkbox next to your public route table.
- Select **Modify route tables.**



- Head back to the **Subnets** page.
- Select the refresh button next to the **Actions** dropdown.
- Check the **Route table** tab for your public subnet again.



- There's a route to the VPC endpoint now.

Access your S3 bucket Again

Our VPC endpoint set up is now!

To validate your work, let's get your EC2 instance to interact with your S3 bucket one last time.

In this step, you're going to:

1. Test your VPC endpoint set up (again).
2. Restrict your VPC's acccess to your AWS environment.

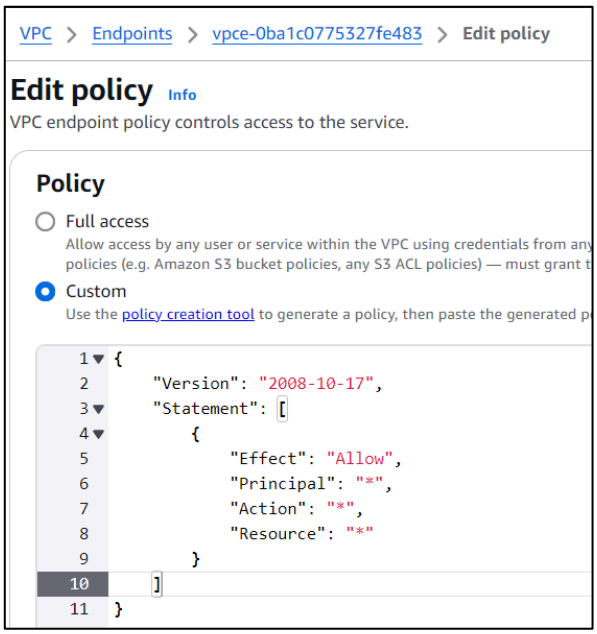
- Back in your EC2 Instance Connect tab, run aws s3 ls s3://nextwork-vpc-endpoints-yourname **(in my case:** nextwork-vpc-endpoints-vivek) again.

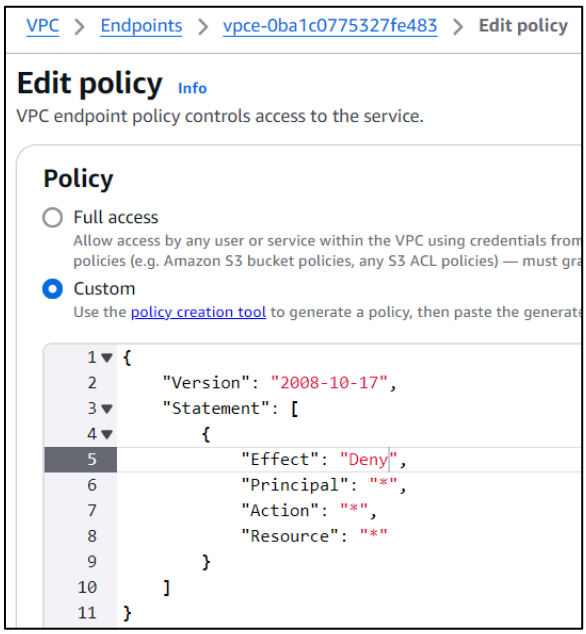
```
[ec2-user@ip-10-0-6-188 ~]$ aws s3 ls s3://nextwork-vpc-endpoints-vivek
2025-01-09 09:06:06    1299232 Project - Host a Website on Amazon S3.pdf
2025-01-09 09:06:07    5549573 Project - Set Up a Web App in the Cloud.pdf
2025-01-09 10:16:19          0 nextwork.txt
[ec2-user@ip-10-0-6-188 ~]$
```

- We have successfully connected back via VPC Endpoint Connection.

VPC endpoints can be configured using policies too?!

- We'll do a simple configuration - switch to your **Endpoints** tab.
- Select the checkbox next to your policy.
- Select the **Policy** tab.
- Ooo! By default, your VPC endpoint allows access to all AWS services.
- Select **Edit policy.**
- Change the line **"Effect": "Allow"** to **"Effect": "Deny"**!



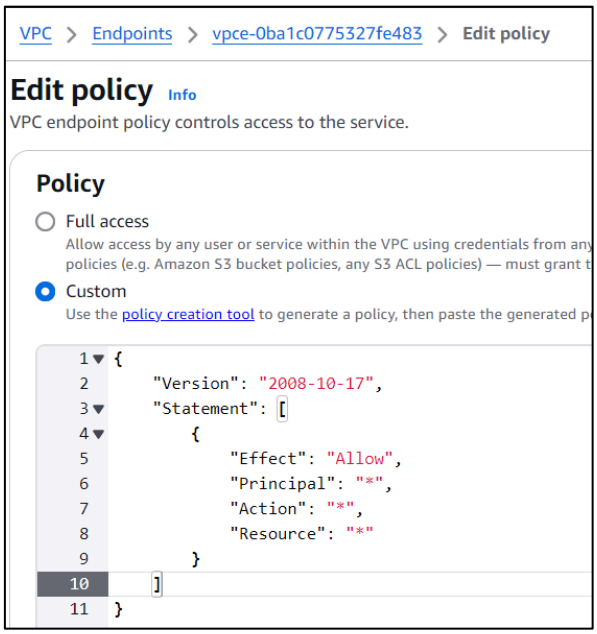


- Click **Save.**
- Switch back to your EC2 Instance Connect tab.
- Run aws s3 ls s3://nextwork-vpc-endpoints-yourname **(in my case:** nextwork-vpc-endpoints-vivek)

```
[ec2-user@ip-10-0-6-188 ~]$ aws s3 ls s3://nextwork-vpc-endpoints-vivek

An error occurred (AccessDenied) when calling the ListObjectsV2 operation: User: arn:aws:iam::2457123040
n resource: "arn:aws:s3:::nextwork-vpc-endpoints-vivek" with an explicit deny in a VPC endpoint policy
[ec2-user@ip-10-0-6-188 ~]$
```

- The endpoint policy now stops us from accessing our S3 bucket.
- Switch tabs back to your **Endpoints** page.
- Select **Edit Policy.**
- Change the policy back to "Effect": "Allow" **- this will be important when it comes to deleting your resources later!**



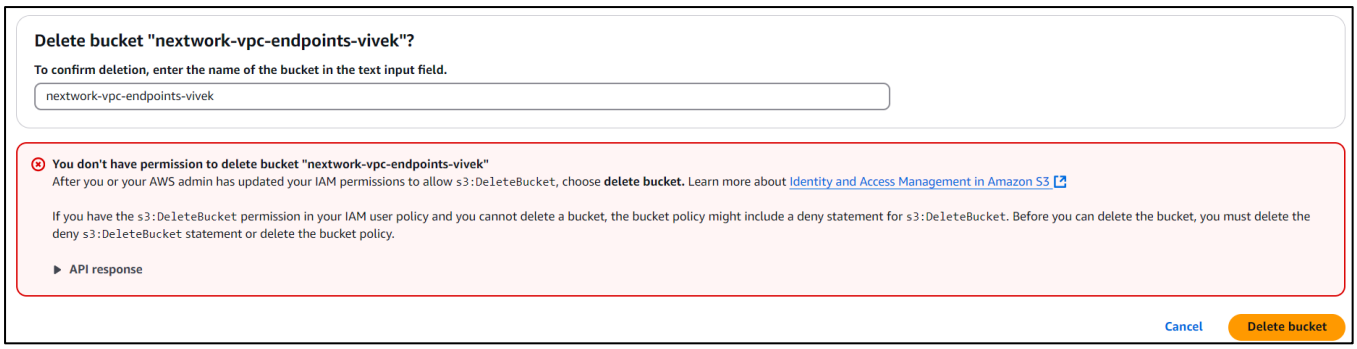
- Select **Save changes.**

We have completed the project and **learnt how to set up direct, private access to S3 from your VPC.**

Delete Your Resources

Delete Your S3 Buckets

- Head to your **S3** console.
- Select the **Buckets** page.
- Select your **nextwork-vpc-endpoints-yourname (in my case:** nextwork-vpc-endpoints-vivek), and select **Delete**.
- Enter your bucket name, and select **Delete bucket**.



- Deleting your bucket from the console is not possible - don't forget, your S3 bucket policy literally denies everyone else (**except** traffic from the VPC endpoint) access to the bucket.
- Our only option is to delete our bucket through the EC2 instance!
- Switch tabs to your instance's terminal.
- Run the command aws s3 rb s3://nextwork-vpc-endpoints-yourname **(in my case:** nextwork-vpc-endpoints-vivek)

What does this command do?

rb is a command that deletes your bucket!

- This means rb s3://nextwork-vpc-endpoints-yourname is used to delete your S3 bucket nextwork-vpc-endpoints-yourname.

```
[ec2-user@ip-10-0-6-188 ~]$ aws s3 rb s3://nextwork-vpc-endpoints-vivek
remove_bucket failed: s3://nextwork-vpc-endpoints-vivek An error occurred (BucketNotEmpty)
```

- The delete failed - we need to make sure our bucket is empty first.
- To delete everything in your bucket, run the command aws s3 rm s3://nextwork-vpc-endpoints-yourname --recursive **(in my case:** nextwork-vpc-endpoints-vivek)

What does this command do?

- rm stands for "remove" and is used to remove things inside your bucket.
- --recursive means the remove command should include **all** objects and subdirectories inside your bucket. This is a super helpful command to run before delete a non-empty bucket.

```
[ec2-user@ip-10-0-6-188 ~]$ aws s3 rm s3://nextwork-vpc-endpoints-vivek --recursive
delete: s3://nextwork-vpc-endpoints-vivek/Project - Set Up a Web App in the Cloud.pdf
delete: s3://nextwork-vpc-endpoints-vivek/Project - Host a Website on Amazon S3.pdf
delete: s3://nextwork-vpc-endpoints-vivek/nextwork.txt
```

- Run the command aws s3 rb s3://nextwork-vpc-endpoints-yourname **(in my case:** nextwork-vpc-endpoints-vivek) again.

```
[ec2-user@ip-10-0-6-188 ~]$ aws s3 rb s3://nextwork-vpc-endpoints-vivek
remove_bucket: nextwork-vpc-endpoints-vivek
```

- S3 bucket has been successfully removed.

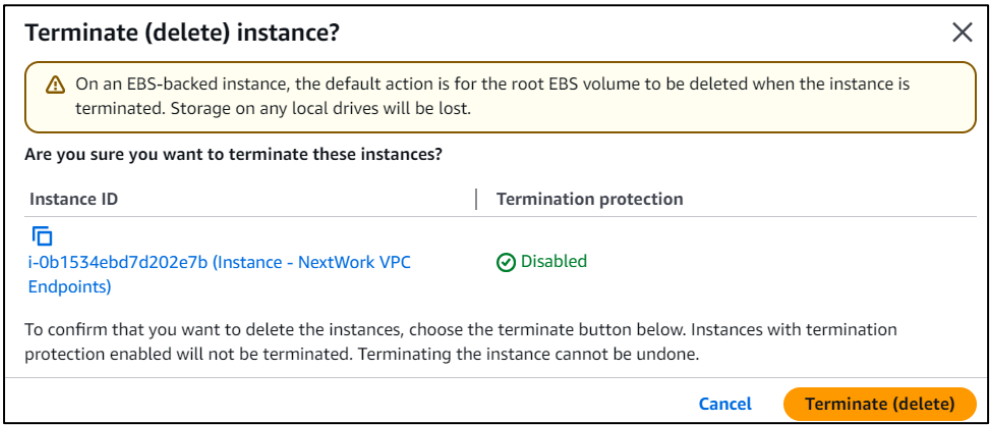
- Let's check it's gone by running aws s3 ls one last time to verify if our bucket still show in the list of results.

```
[ec2-user@ip-10-0-6-188 ~]$ aws s3 ls
2025-01-03 07:59:27 nextwork-website-project-vivek
2024-11-25 12:46:28 practicebin
2024-12-03 05:34:29 terraform-state-bucket-projectb114
2024-11-10 12:40:37 vivek114
```

- We can see that our bucket nextwork-vpc-endpoints-yourname **(in my case:** nextwork-vpc-endpoints-vivek) no longer exists.

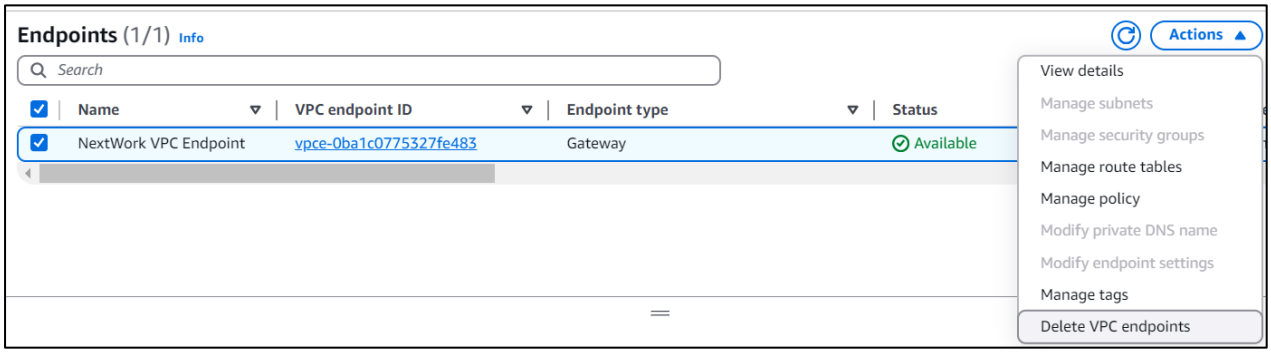
Delete your EC2 Instance

- Head back to the **Instances** page of your EC2 console.
- Select the checkboxes next to **Instance - NextWork VPC Endpoint.**
- Select **Instance state,** then select **Terminate Instance.**
- Select **Terminate**.



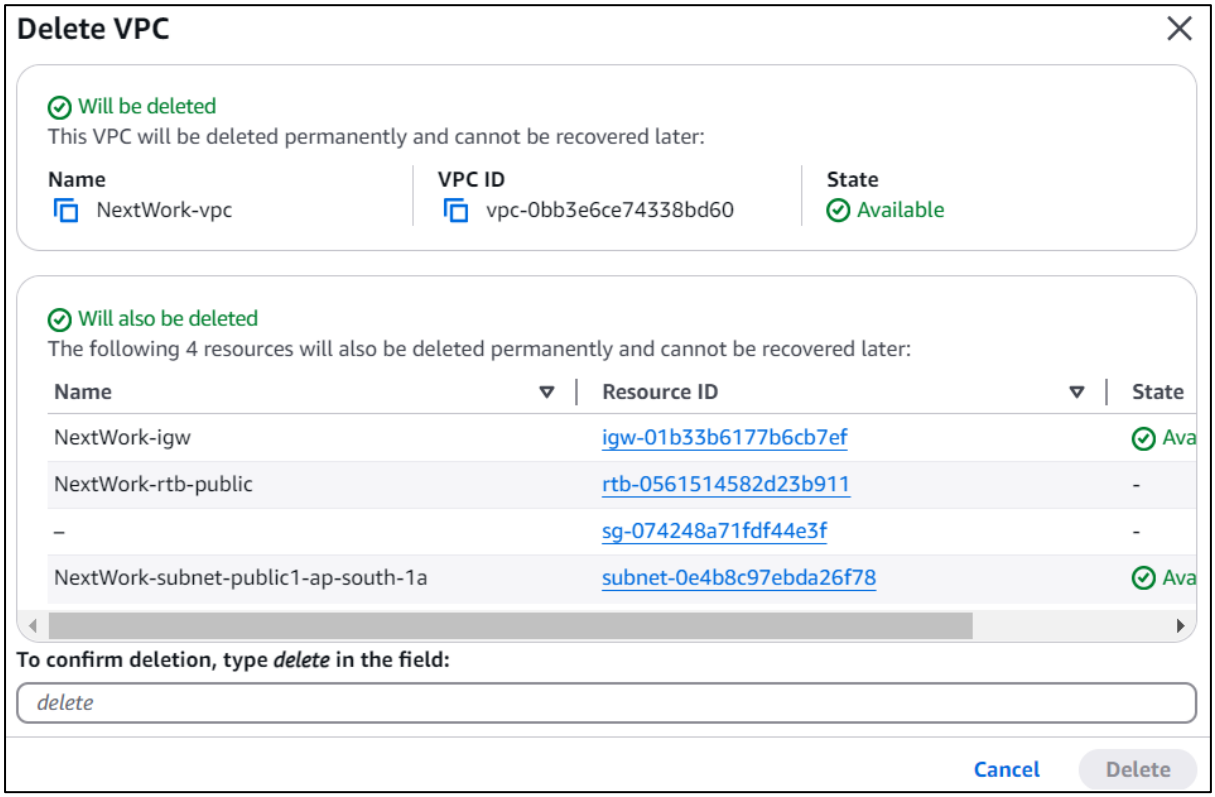
Delete your VPC Endpoint

- Head back to the **Endpoints** page of your VPC console.
- Select the checkbox next to your endpoint.
- Select the **Actions** dropdown.
- Select **Delete VPC endpoints.**



Delete your VPCs

- Select **Your VPCs** from your left hand navigation panel.
- Select **NextWork-vpc,** then **Actions**, and **Delete VPC.**



- Type delete in the text box and click **Delete.**
- Note: if you get stopped from deleting your VPC because **network interfaces** are still attached to your VPC - delete all the attached network interfaces first!

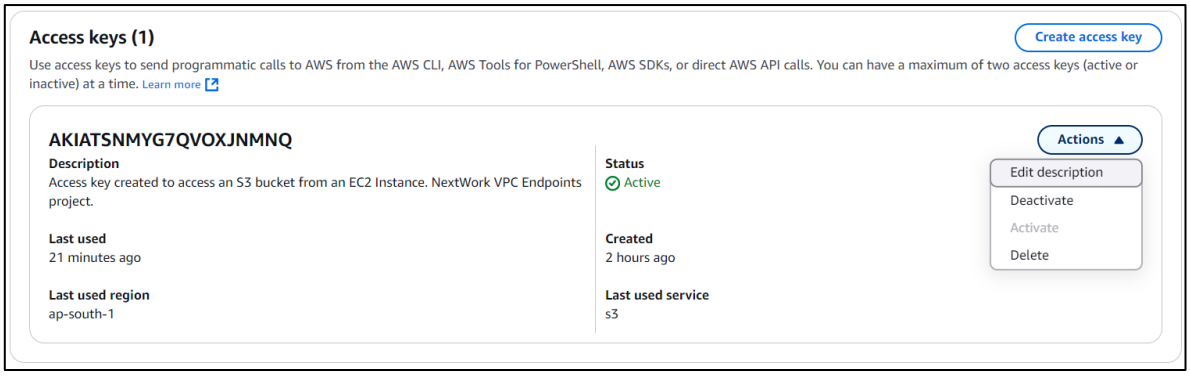
Other network components should be automatically deleted with your VPC, but it's always a good idea to check anyway.

Don't forget to **refresh** each page before checking if these resources are still in your account:

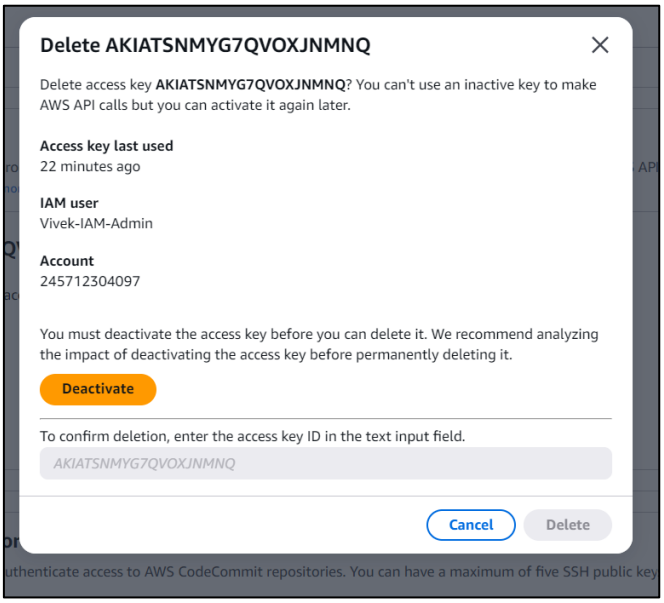
1. Subnets
2. Route tables
3. Internet gateways
4. Network ACLs
5. Security groups

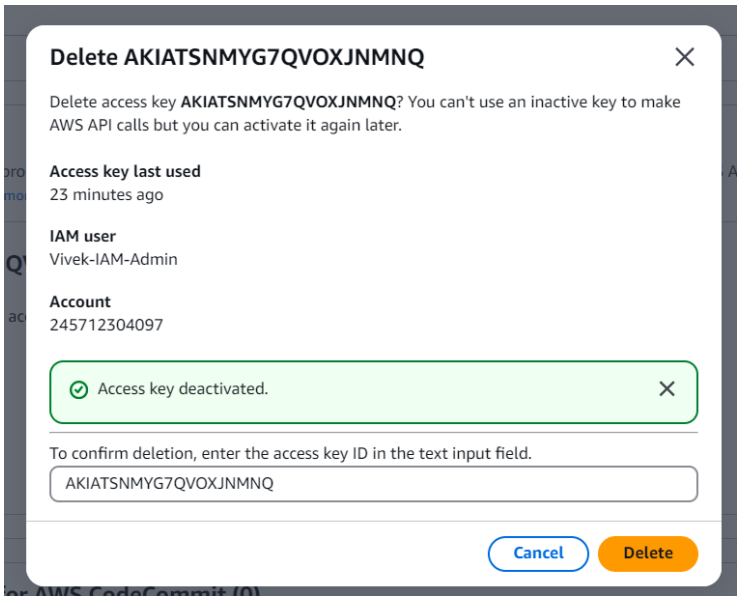
Delete Your IAM Access Keys

- Head to your **IAM** console.
- Select **Users**.
- Select your **Security credentials** tab.
- Scroll down to your **Access keys** panel and select the **Actions** drop down.
- Select **Delete.**



- At the popup panel, select **Deactivate**, and enter your access key ID into the text field.





- Select **Delete.**
- Last but definitely not least - don't forget to delete the local access key **.csv file** saved on your local computer!