😊 Interface Endpoints

AWS Interface Endpoints are a type of VPC endpoint that allows you to privately connect your VPC (Virtual Private Cloud) to AWS services. They use AWS PrivateLink, which enables you to access services over a private network rather than the public internet. Here's a breakdown of what they are and how they work:

Key Features:

1. **Private Connectivity**: Interface endpoints provide private connectivity between your VPC and the AWS service without requiring access over the internet. This means you can connect to AWS services such as S3, DynamoDB, or any other service that supports PrivateLink from within your VPC.
2. **Elastic Network Interfaces**: They work by creating an Elastic Network Interface (ENI) in your VPC. Each ENI has a private IP address and serves as an entry point to the service.
3. **Service Accessibility**: You can access the service as if it were part of your VPC network. This is useful for scenarios where you want to enhance security or improve network performance by avoiding internet traffic.
4. **Security and Control**: You can control access to your interface endpoints using security groups and policies, providing an additional layer of security and fine-grained access control.
5. **Integration with VPC**: Interface endpoints integrate seamlessly with other VPC features, such as route tables and security groups, allowing for flexible and secure network architectures.

Common Use Cases:

- **Enhanced Security**: Keep sensitive data secure by avoiding exposure to the public internet.
- **Improved Performance**: Potentially reduce latency and improve network performance by routing traffic within the AWS network.
- **Private Connectivity**: Access AWS services from within your VPC without needing NAT gateways or internet gateways.

How to Set Up:

1. **Create an Interface Endpoint**: In the AWS Management Console, navigate to the VPC service, select "Endpoints," and create a new endpoint. Choose the service you want to connect to.
2. **Specify VPC and Subnets**: Choose the VPC and subnets where you want to create the endpoint. AWS will automatically create the necessary ENIs.

3. **Configure Security Groups**: Attach security groups to the endpoint to control inbound and outbound traffic.
4. **Update Route Tables (if necessary)**: In some cases, you might need to update your route tables to ensure traffic is directed to the endpoint.

By leveraging AWS Interface Endpoints, you can ensure secure, efficient, and private access to AWS services from within your VPC, aligning with best practices for network security and performance.

🧐 What are we doing in this Lab?

In this exercise, you are setting up and using an **interface VPC endpoint** to securely connect to AWS services (specifically, the EC2 service) within a VPC without using the public internet. The key steps include creating an IAM role with EC2 read-only permissions, launching an EC2 instance with this role, and setting up the VPC endpoint. You then use the VPC endpoint to make requests to the EC2 service, observing network traffic to verify the data path.

Key Steps:

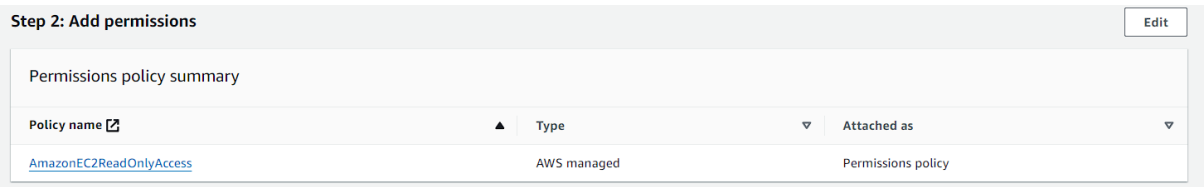
1. **Create IAM Role**: A role with EC2 read-only permissions is created for the EC2 instance.
2. **Launch EC2 Instance**: An instance is launched with the IAM role attached.
3. **Create VPC Endpoint**: An interface VPC endpoint for the EC2 service is created, allowing private communication within the VPC.
4. **Test Communication**: You use the instance to make a request to the EC2 service and monitor network traffic to verify that it routes through the VPC endpoint.

End Goal:

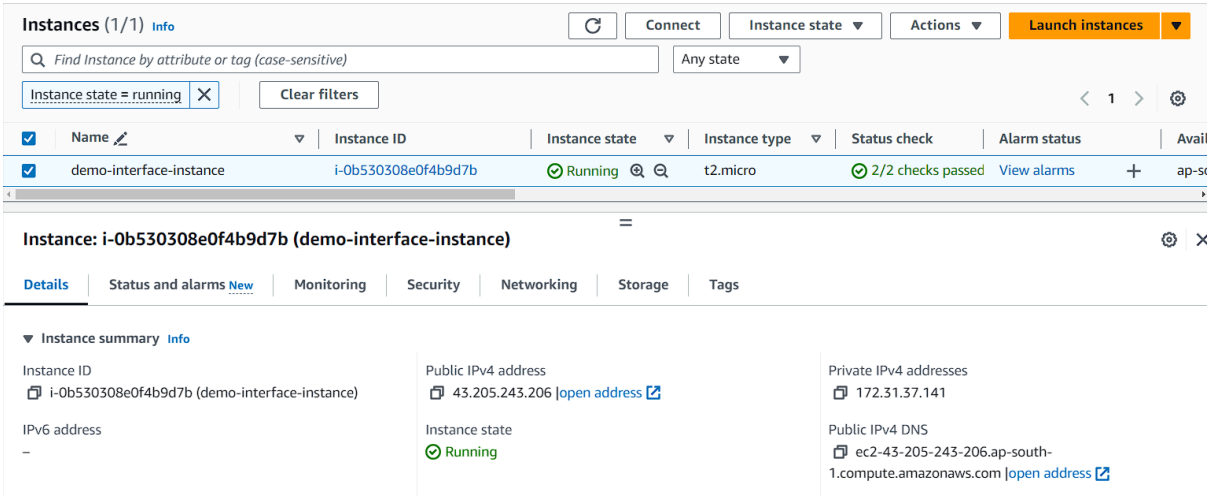
The main goal of this exercise is to show how you can connect to AWS services, like checking your EC2 instances, securely within your private network (VPC) without using the internet. This setup keeps your data safe by ensuring it doesn't leave the protected environment of your VPC, making it more secure and potentially cheaper. It's like having a private, secure tunnel inside your cloud environment to access AWS services.

😄 To begin with the Lab:

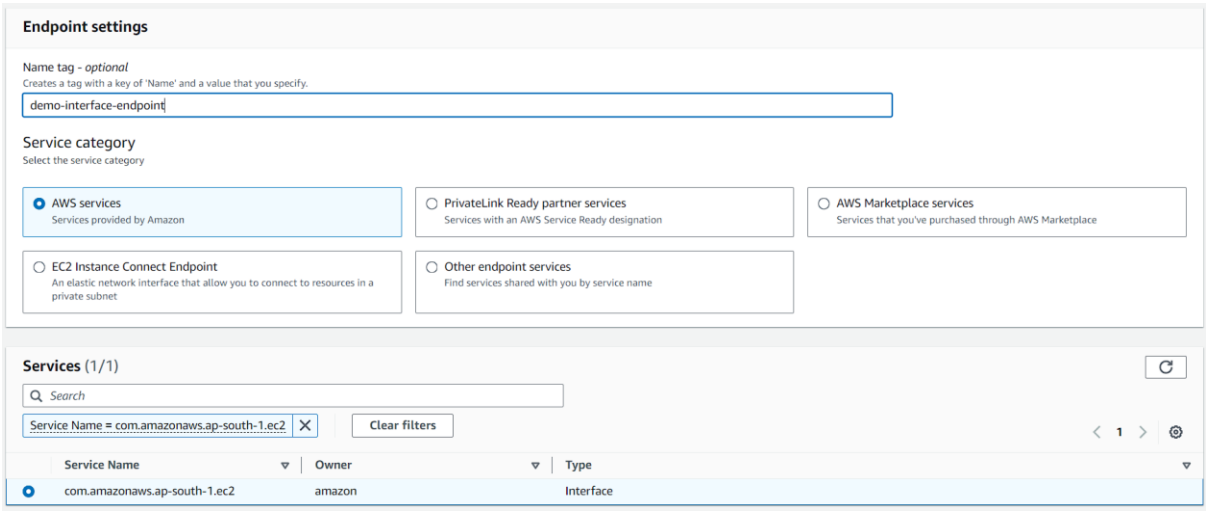
1. Login to AWS Console and navigate to IAM. There you are going to create a role for EC2.
2. You have to attach EC2 read-only permission to it and then create it.



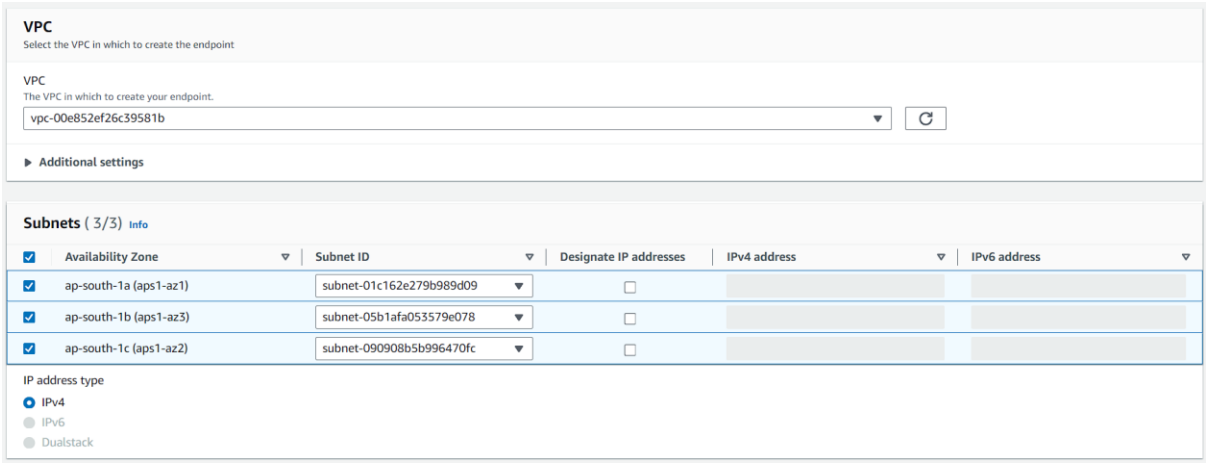
3. Now navigate to EC2 and create an instance. Remember to attach your IAM role with it.



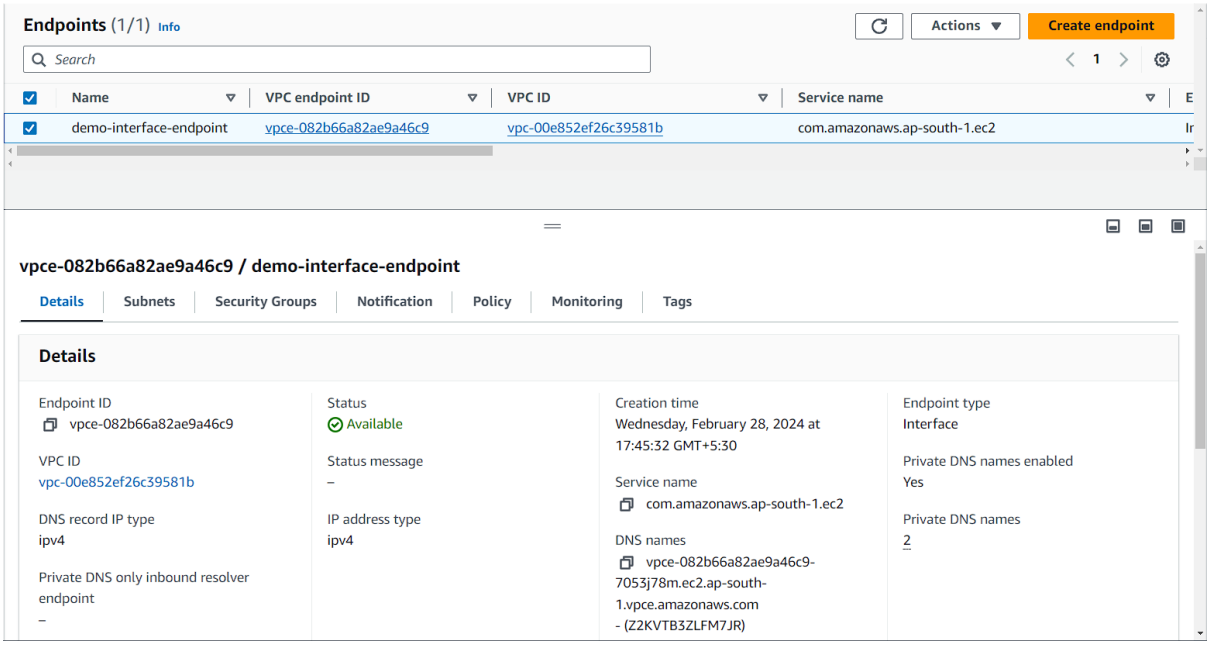
4. Now you have to navigate to VPC and from the left pane select Endpoints and click on Create Endpoints.
5. Here you have to give it a name and then select AWS Services. There are multiple services available, but you have to search for EC2 and select that.



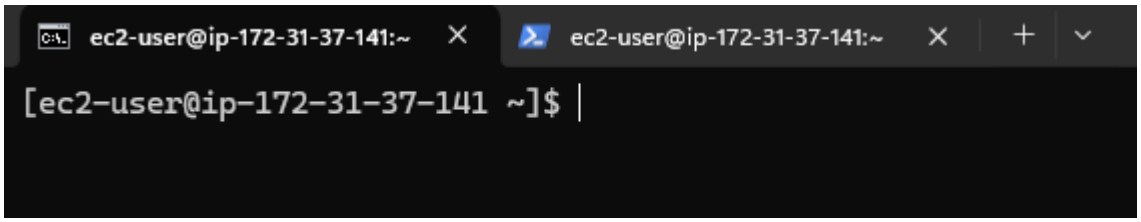
6. Then you have to select your VPC and the subnets along with the subnet IDs.



7. Then choose any of the Security group or you can create new security group for this lab. Attach all traffic from everywhere in the inbound rules.
8. Then just create your endpoint.
9. Once your endpoint is created and the status is available.



10. Then you have to SSH into your instance. Use Command prompt for it because you will be connecting your instance in the two tabs like shown below.



11. Now you are going to run commands in both of them. Now in the first tab run this command. Your instance will start listening.

    sudo tcpdump dst port 443

```
ec2-user@ip-172-31-37-141:~   ec2-user@ip-172-31-37-141:~
[ec2-user@ip-172-31-37-141 ~]$ sudo tcpdump dst port 443
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enX0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

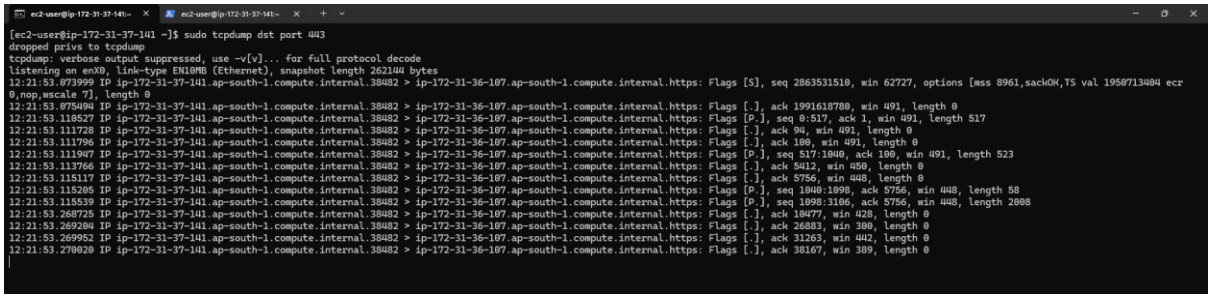
12. After that in the second tab you have to run this command. You will get results or say output as expected.

    aws ec2 describe-instances --region ap-south-1

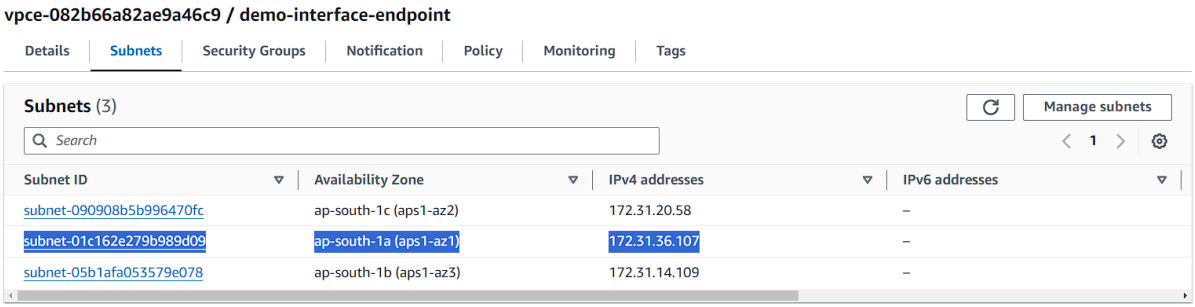
```
ec2-user@ip-172-31-37-141:~   ec2-user@ip-172-31-37-141:~
[ec2-user@ip-172-31-37-141 ~]$ aws ec2 describe-instances --region ap-south-1
{
    "Reservations": [
        {
            "Groups": [],
            "Instances": [
                {
                    "AmiLaunchIndex": 0,
                    "ImageId": "ami-03f48787554349777f",
                    "InstanceId": "i-0b2d66f874cd0af2f",
                    "InstanceType": "t2.micro",
                    "KeyName": "dockerdemo",
                    "LaunchTime": "2024-02-17T04:07:58+00:00",
                    "Monitoring": {
                        "State": "disabled"
                    },
                    "Placement": {
                        "AvailabilityZone": "ap-south-1a",
                        "GroupName": "",
                        "Tenancy": "default"
                    },
                    "PrivateDnsName": "ip-172-31-41-113.ap-south-1.compute.internal",
                    "PrivateIpAddress": "172.31.41.113",
                    "ProductCodes": [],
                    "PublicDnsName": "",
                    "State": {
                        "Code": 80,
                        "Name": "stopped"
                    },
                    "StateTransitionReason": "User initiated (2024-02-17 04:39:00 GMT)",
                    "SubnetId": "subnet-01c162e279b989d09",
                    "VpcId": "vpc-00e852ef26c39581b",
                    "Architecture": "x86_64",
                    "BlockDeviceMappings": [
                        {
                            "DeviceName": "/dev/sda1",
                            "Ebs": {
                                "AttachTime": "2024-02-08T11:43:44+00:00",
                                "DeleteOnTermination": true,
                                "Status": "attached",
                                "VolumeId": "vol-01d6027664d4375fe"
                            }
                        }
                    ],
                    "ClientToken": "9276adb3-b876-44bd-9822-0c2a2167789a",
                    "EbsOptimized": false,
                    "EnaSupport": true,
                    "Hypervisor": "xen",
                    "IamInstanceProfile": {
                        "Arn": "arn:aws:iam::878893308172:instance-profile/demo-ec2-flowlogs",
                        "Id": "AIPA4ZIQ7TEGG7DIOEHP7"
```

13. Now if you go back to the first tab you will see that data is generated.

14. Here you will see that the request from your instance is going to the interface endpoint IP address.



15. Now to check which endpoint IP address is this you can visit to endpoint and if you will open subnets, you will see that IP address and you can match them.
16. Here you can see that the highlighted IP address is the one which our instance is sending requests to.



17. When you are done just delete your endpoint and terminate your instance.