Assignment

1. Setup Secure Central VPC Networking to Share APIs Hosted in Multiple VPCs within AWS Account.

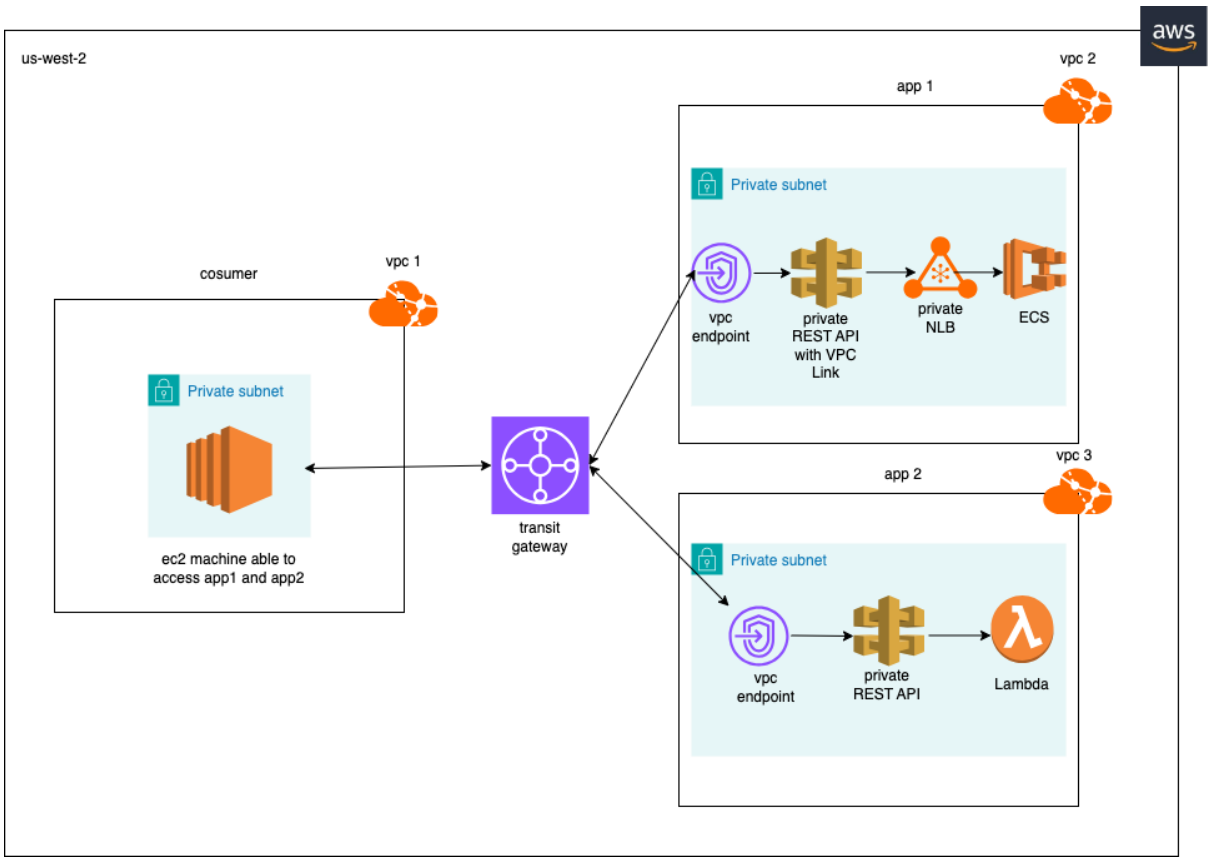
Solution

Tech Stack

1. AWS (S3, VPC, ECS, ECR, Lambda, EC2, NLB, API Gateway. Transit Gateway)
2. Terraform (Infra Provision)
3. Github (Source Code)
4. Docker (for ECS)
5. Java, python

Repo -> https://github.com/venkat-raju0492/freyr/tree/main/1

Architecture Diagram



VPC

Created three VPC's below are CIDR Ranges

VPC1

CIDR: 10.0.0.0/20

public_subnet_cidr = ["10.0.0.0/23","10.0.2.0/23","10.0.4.0/23"]

private_subnet_cidr = ["10.0.6.0/23","10.0.8.0/23","10.0.10.0/23"]

VPC2

CIDR: 10.0.32.0/20

public_subnet_cidr = ["10.0.32.0/23","10.0.34.0/23","10.0.36.0/23"]

private_subnet_cidr = ["10.0.38.0/23","10.0.40.0/23","10.0.42.0/23"]

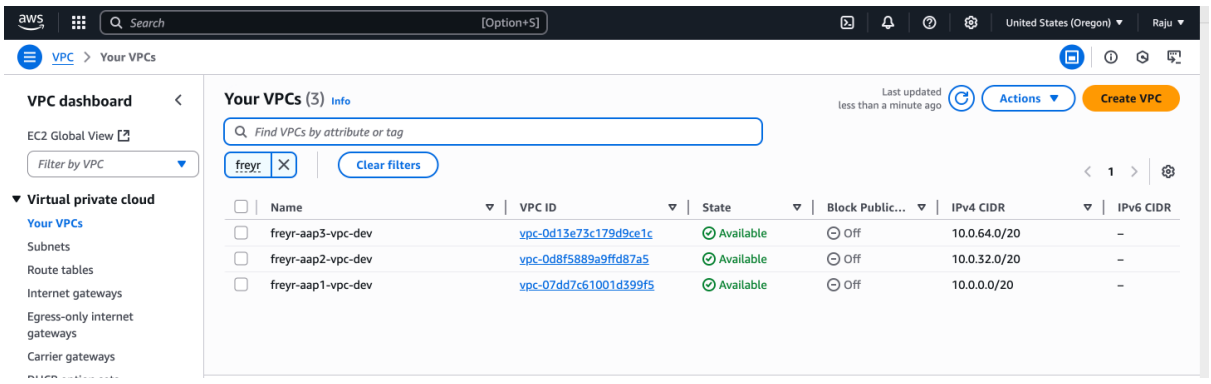
VPC 3

CIDR: 10.0.64.0/20

public_subnet_cidr = ["10.0.64.0/23","10.0.66.0/23","10.0.68.0/23"]

private_subnet_cidr = ["10.0.70.0/23","10.0.72.0/23","10.0.74.0/23"]

Terraform code -> https://github.com/venkat-raju0492/freyr/tree/main/1/vpc



Each VPC has respective subnets, route tables, internet & NAT gateways

App1

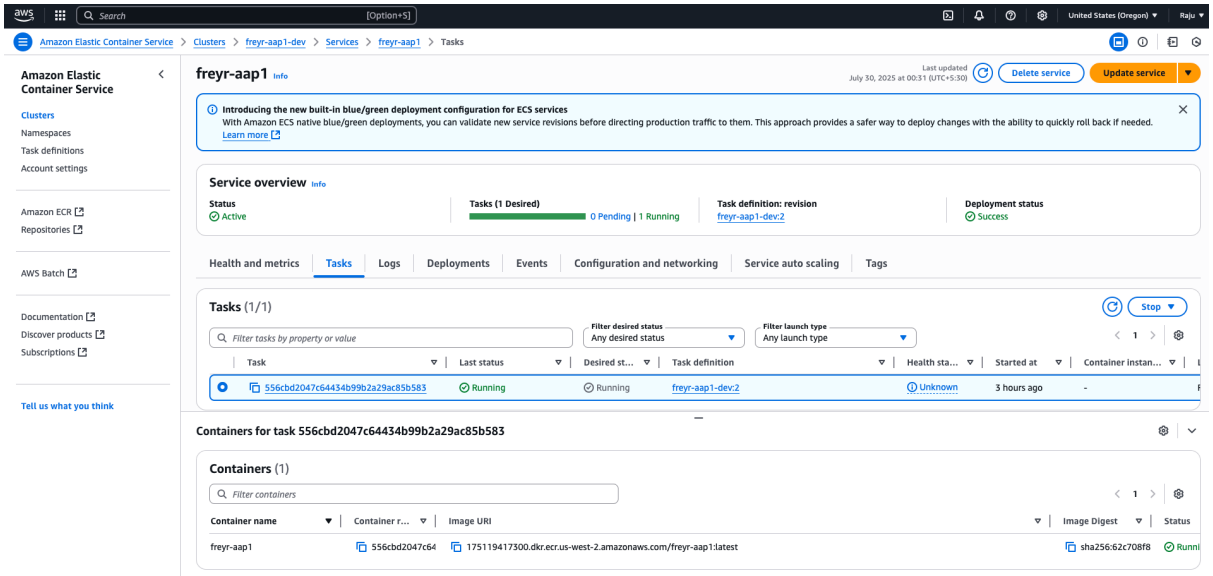
- Java sample hello world program configured with actuator health, compiled with maven generating app.jar
- Docker file with alpine java base image copying app.jar into docker image and building the image with ECR tag format and pushing it to ECR
- Deploying the docker image into ECS as Fargate service using fargate spot for optimized cost as this is for dev env and not recommended for production
- Exposing the container with internal NLB with target port 80 forwarded to application port 8080
- Integrating Private REST API with NLB through VPC Link with vpc endpoint resource policy
- To access Private REST API through secured channel connectivity which can be achieved with VPC endpoint

APP1 is deployed in VPC1

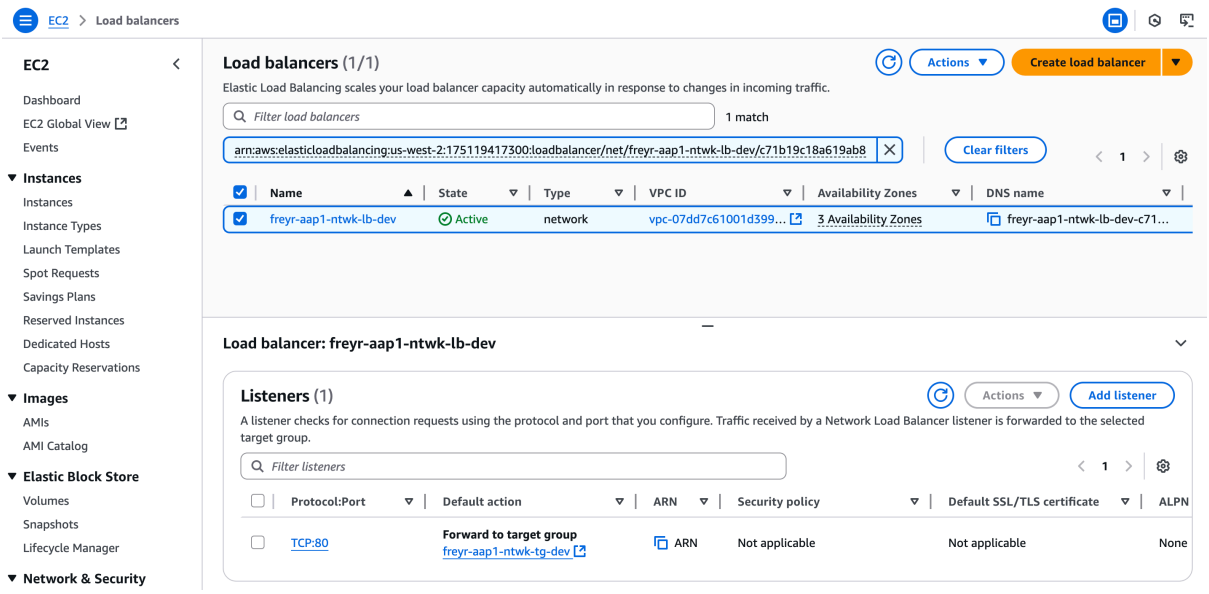
Infra Provision -> https://github.com/venkat-raju0492/freyr/tree/main/1/app1

Sample application code -> https://github.com/venkat-raju0492/freyr/tree/main/1/app1/ecs-app

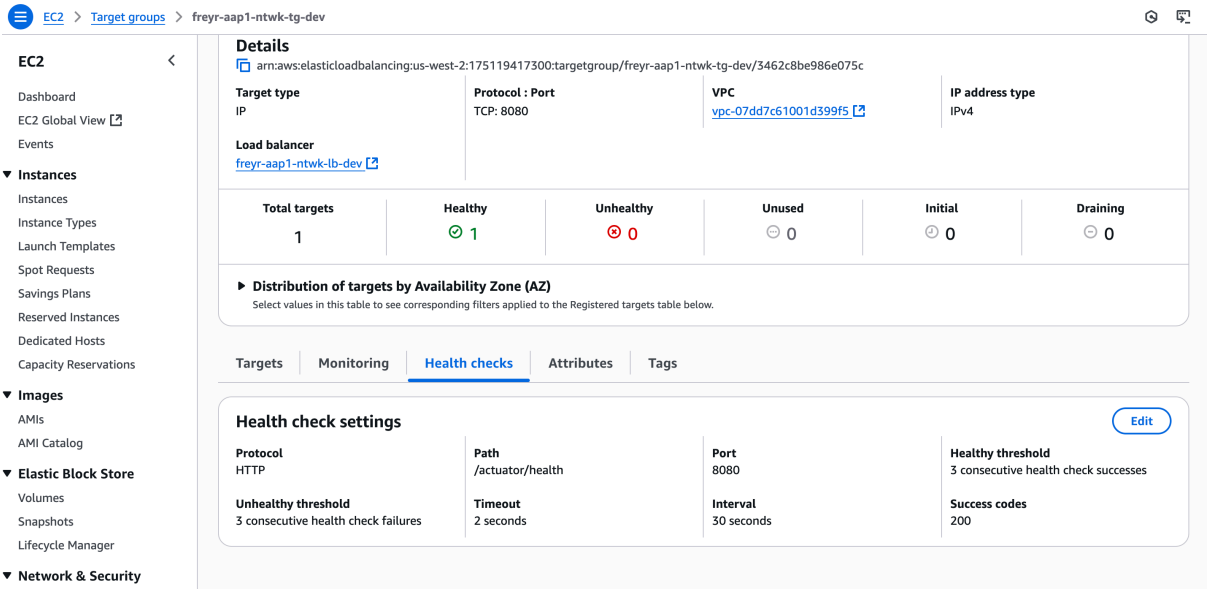
ECS Service



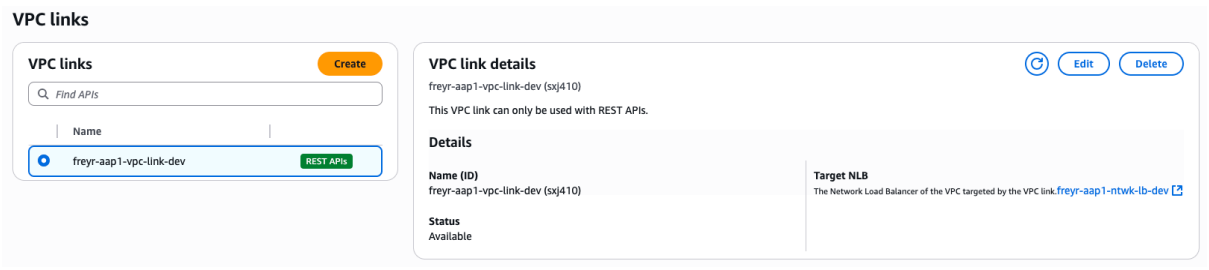
NLB with target group



Target group with healthy targets and its configs



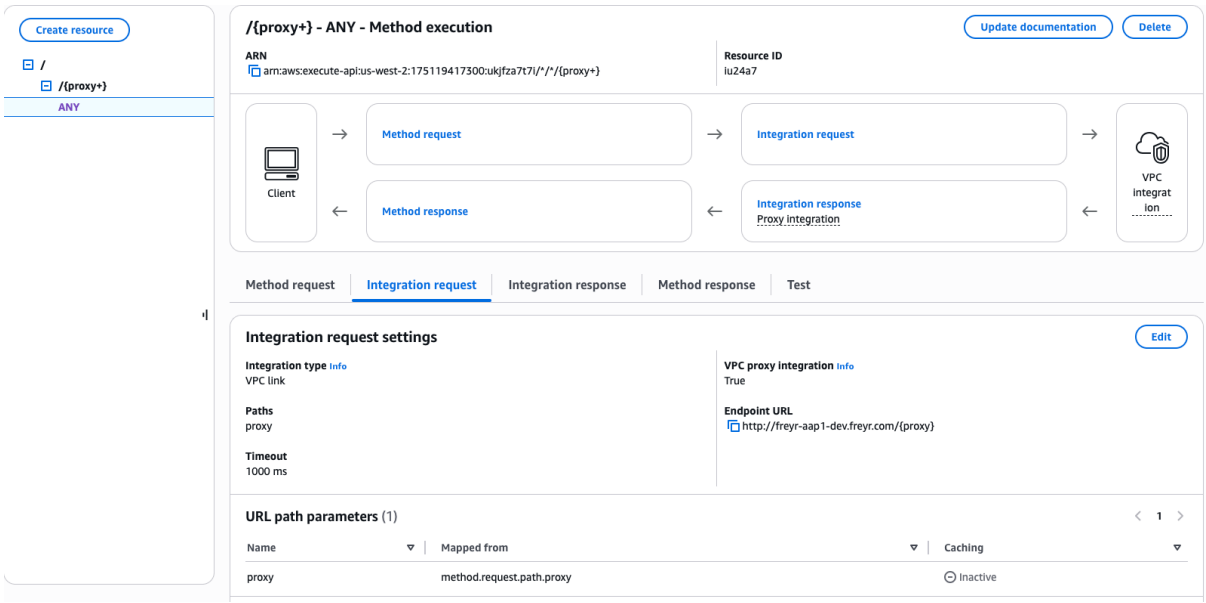
VPC Link integrated with NLB



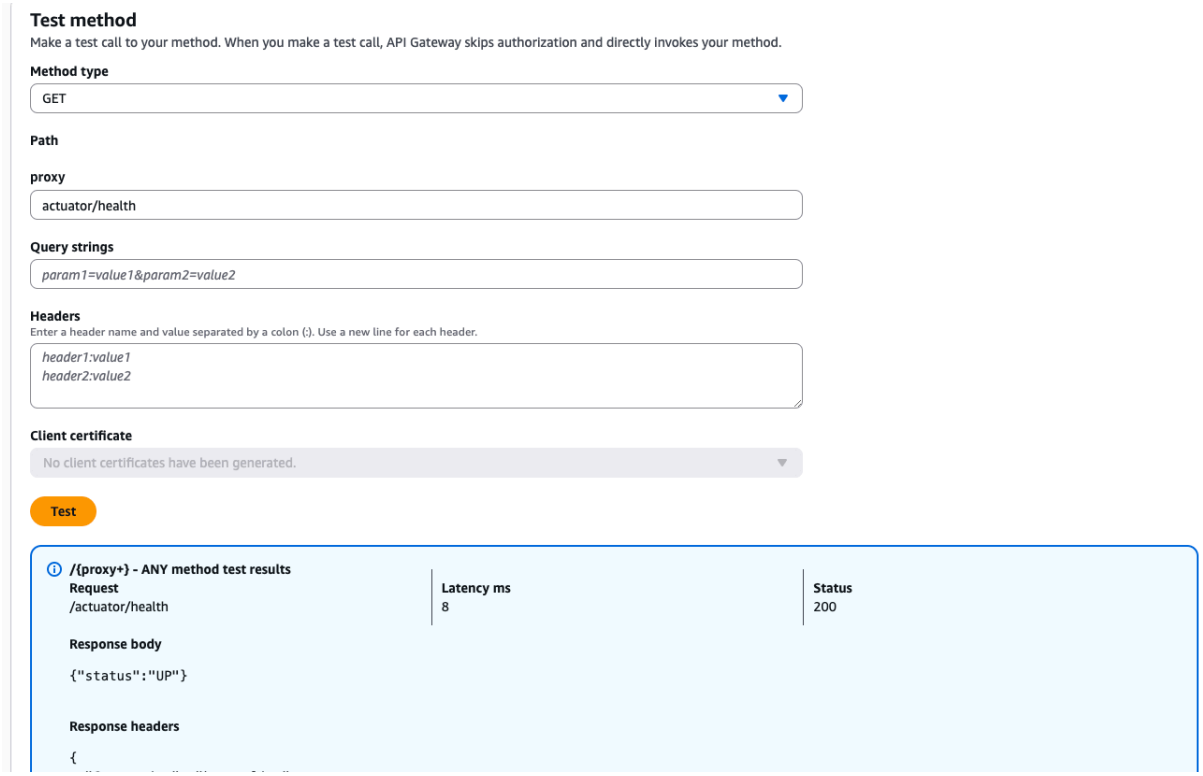
Private Rest API

○	freyr-aap1-api-dev	freyr-aap1-api-dev rest api	ukjfza7t7i	REST	Private	2025-07-29

Proxy resource with VPC link integration



Tested API with actuator path and empty path with 200 success resonse



To access APP1 url -> https://ukjfza7t7i-vpce-05a417755deb94613.execute-api.us-west-2.amazonaws.com/dev/actuator/health

App2

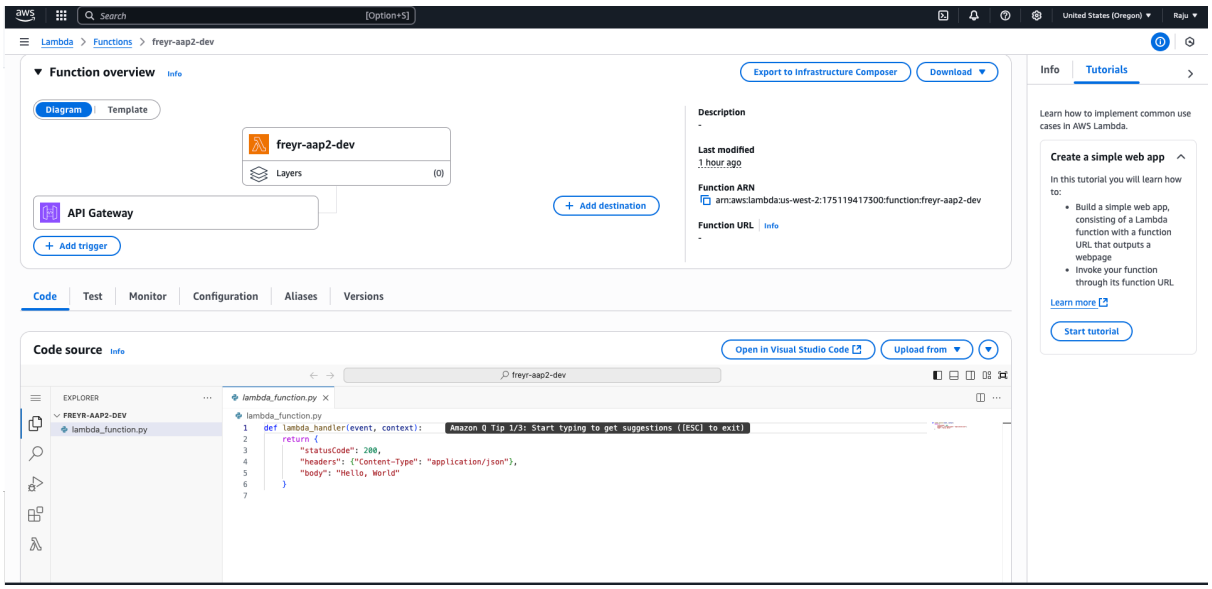
- Sample python lambda code zipped and uploaded to s3
- Deployed the zip file to lambda function
- Added private api gateway with lambda integration and configured lambda triggered

APP2 is deployed in VPC2

Infra Provision -> https://github.com/venkat-raju0492/freyr/tree/main/1/app2

Sample python code -> https://github.com/venkat-raju0492/freyr/tree/main/1/app2/lambda-app

Lambda function with API triggers



Private API Gateway

○	freyr-aap2-api-dev	freyr-aap2-api-dev rest api	jhb0g7w8f4	REST	Private	2025-07-29

Private api gateway has resource policy with vpc endpoint

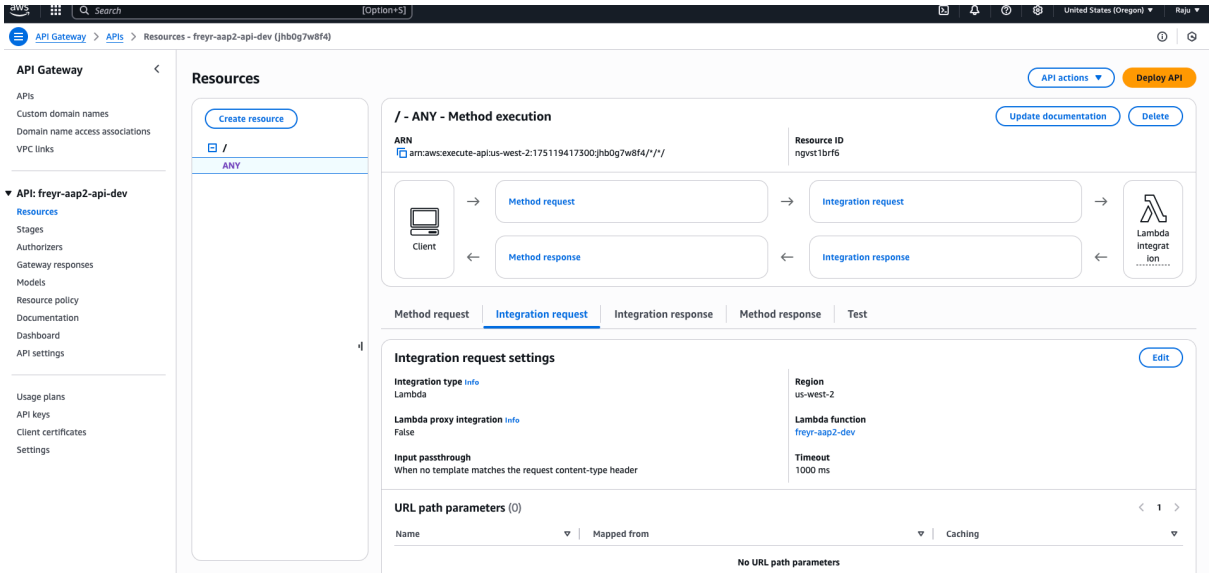
Resource policy Info

Use resource policies to configure access control to this API. You must redeploy your API for changes to this policy to take effect.

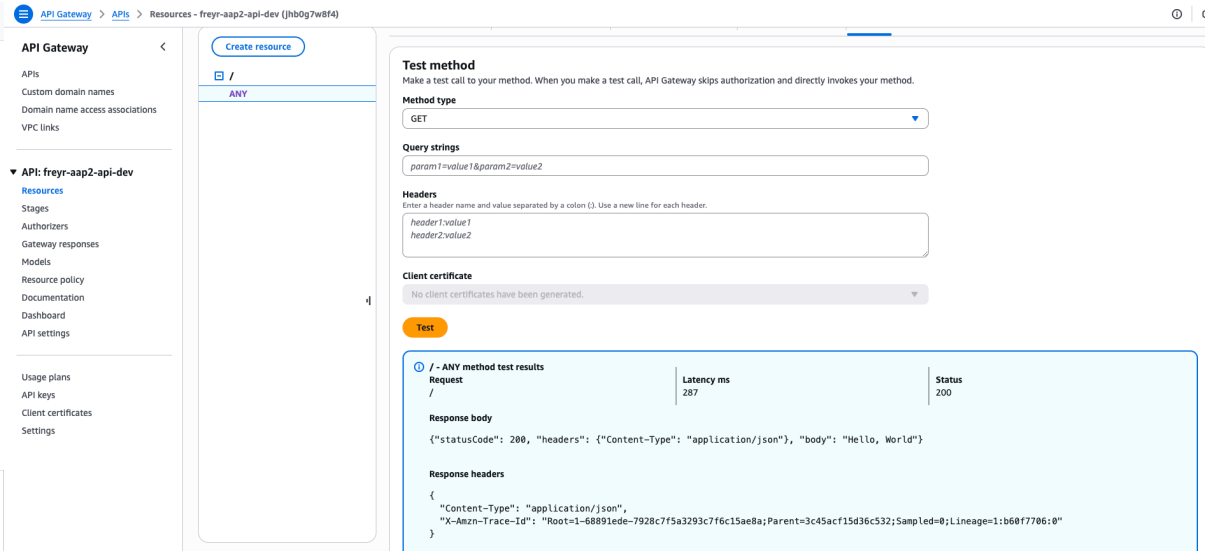
Policy details

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": "arn:aws:execute-api:us-west-2:175119417300:jhb0g7w8f4/*",
      "Condition": {
        "StringNotEquals": {
          "aws:aws:SourceVpce": "vpce-08f35f9cde369685a"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": "arn:aws:execute-api:us-west-2:175119417300:jhb0g7w8f4/*"
    }
  ]
}
```

Lambda integration



Invoking lambda by testing API



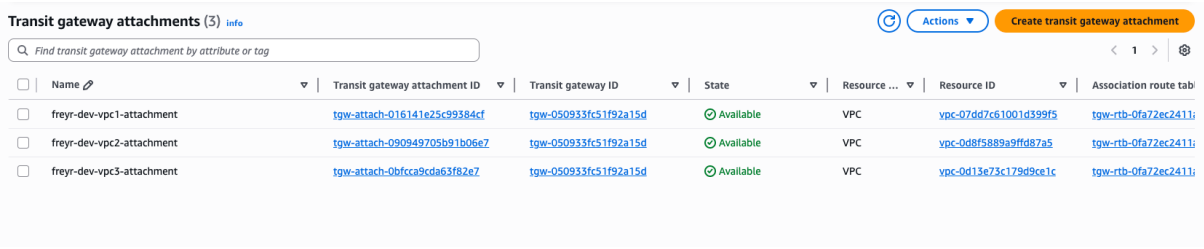
To access APP2 url -> https://jhb0g7w8f4-vpce-08f35f9cde369685a.execute-api.us-west-2.amazonaws.com/dev/

Setting up Transit Gateway and attachment for all three VPC to connect with each other

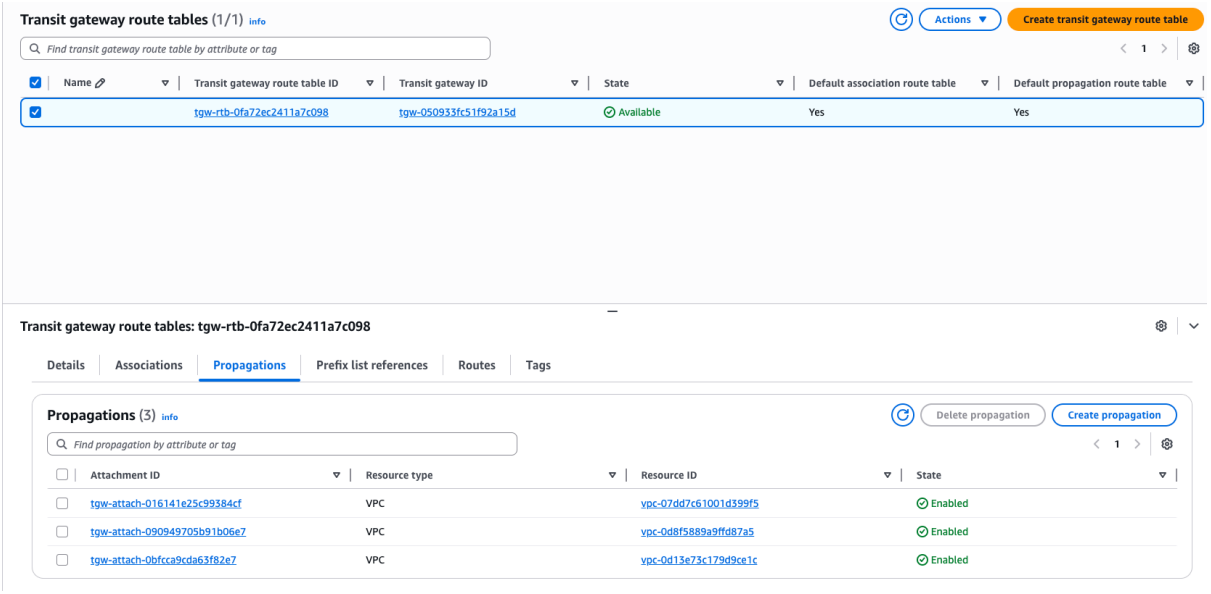
Terraform code -> https://github.com/venkat-raju0492/freyr/tree/main/1/transit-gateway

Transit Gateway

Transit Gateway Attachments

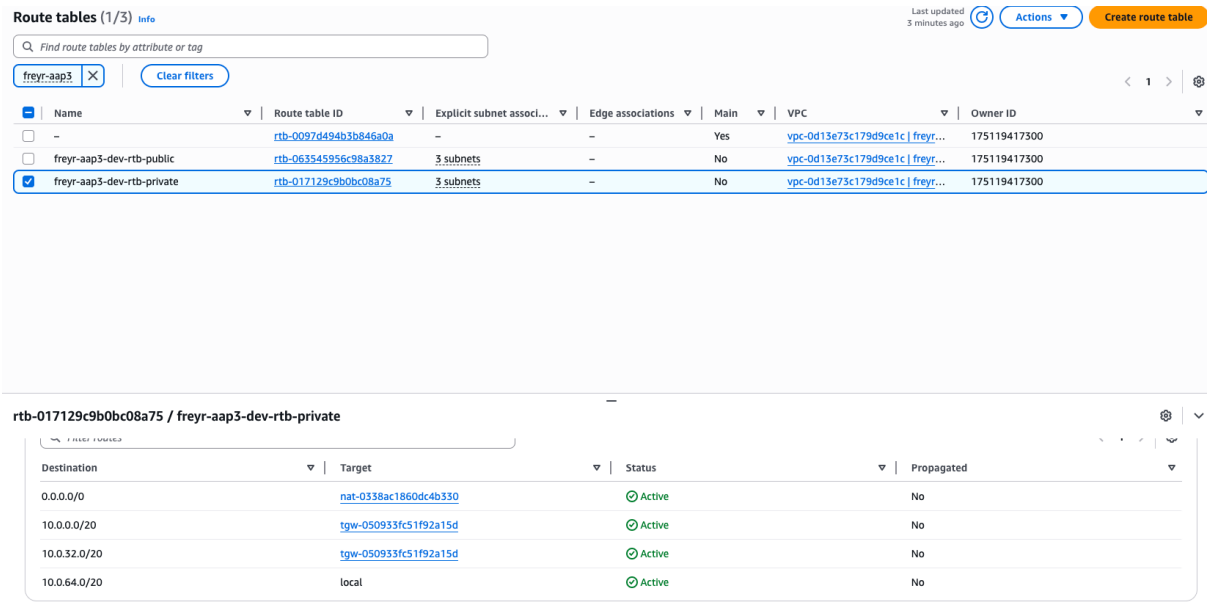


Transit Gateway Route tables and its propogations



Along with this whitelisted all CIDRS in private route tables of each VPC's

Ex:



After this setup APP1 and APP2 can communicate privately make sure CIDR/IP's are allowed in VPC endpoint security group

I have launched a private ec2 in VPC3, I have logged into ec2 and able to access both the applications

```
[[root@ip-10-0-71-176 ~]# curl https://jhb0g7w8f4-vpce-08f35f9cde369685a.execute-api.us-west-2.amazonaws.com/dev/
[{"statusCode": 200, "headers": {"Content-Type": "application/json"}, "body": "Hello, World"}[root@ip-10-0-71-176 ~]#
[[root@ip-10-0-71-176 ~]#
 [root@ip-10-0-71-176 ~]#
[[root@ip-10-0-71-176 ~]#  curl https://ukjfza7t7i-vpce-05a417755deb94613.execute-api.us-west-2.amazonaws.com/dev/actuator/health
 {"status":"UP"}[root@ip-10-0-71-176 ~]#
```

Conclusion

By using transit gateway, we can achieve cross network private connectivity and for API Gateway to securily connect internally we shall be using vpc endpoints by allowing certain cidr’s/ip’s in the security groups