

# Assignment

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

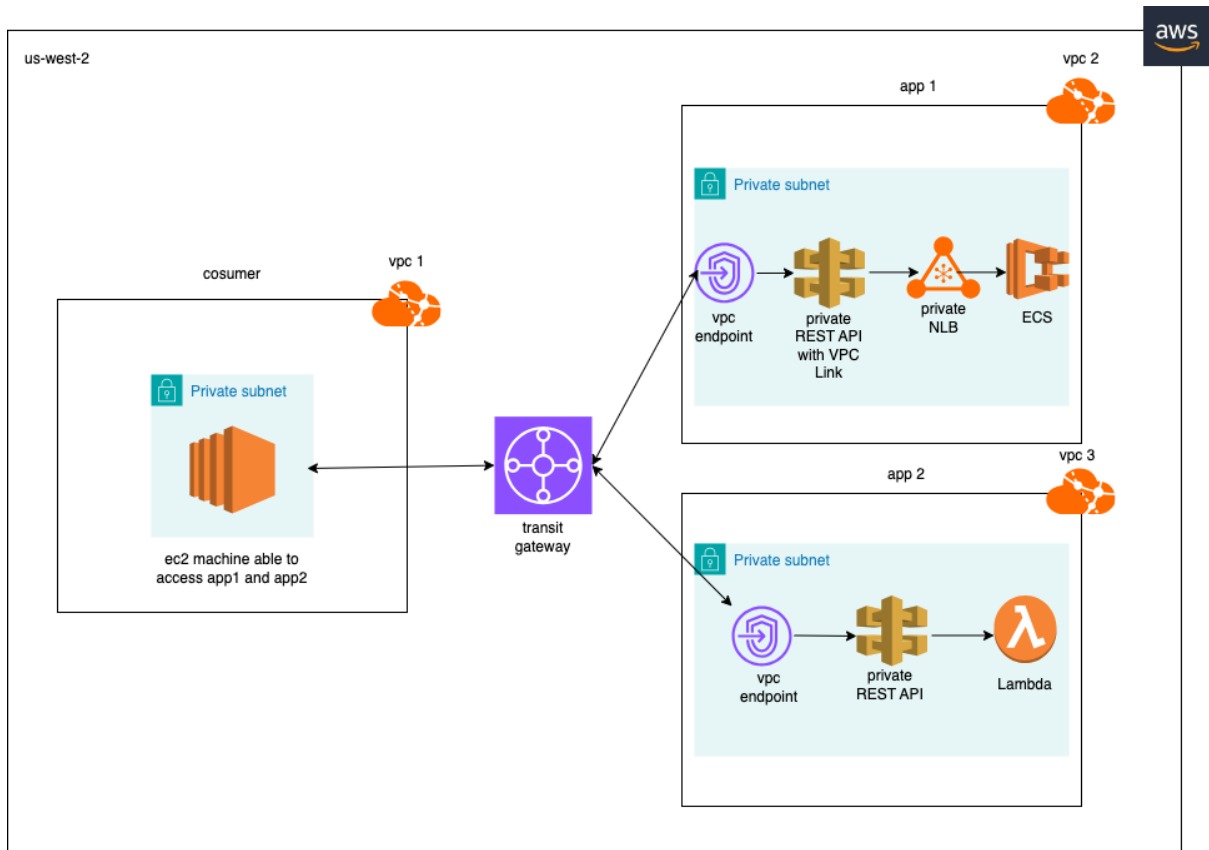
Solution

## 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"]
```

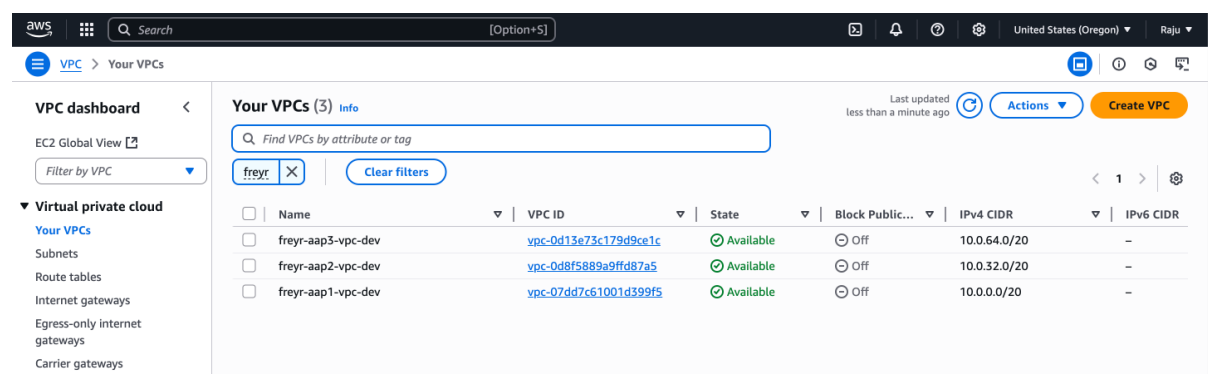
### VPC 3

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

The screenshot shows the AWS Management Console for the Amazon Elastic Container Service (ECS) console. The main navigation bar at the top includes the AWS logo, a search bar, and the region 'United States (Oregon)'. The breadcrumb trail indicates the path: Amazon Elastic Container Service > Clusters > freyr-aap1-dev > Services > freyr-aap1 > Tasks.

The left sidebar contains the 'Amazon Elastic Container Service' section with links to Clusters, Namespaces, Task definitions, and Account settings. Below this are links for Amazon ECR, Repositories, AWS Batch, Documentation, Discover products, and Subscriptions.

The main content area displays the 'freyr-aap1' service overview. It includes a notification about the new blue/green deployment configuration for ECS services. The service overview shows the status as 'Active', with 1 task desired (0 Pending, 1 Running). The task definition is 'revision freyr-aap1-dev-2'. The deployment status is 'Success'.

The 'Tasks' tab is selected, showing a table of tasks. The table has columns for Task, Last status, Desired status, Task definition, Health status, Started at, and Container instances. One task is listed with ID '556cbd2047c64434b99b2a29ac85b583', status 'Running', and task definition 'freyr-aap1-dev-2'.

The 'Containers' tab is also visible, showing a table of containers for the task '556cbd2047c64434b99b2a29ac85b583'. The table has columns for Container name, Container ID, Image URI, Image Digest, and Status. One container is listed with name 'freyr-aap1', ID '556cbd2047c64', and image URI '175119417300.dkr.ecr.us-west-2.amazonaws.com/freyr-aap1:latest'.

# NLB with target group

EC2

Dashboard

EC2 Global View

Events

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

Images

AMIs

AMI Catalog

Elastic Block Store

Volumes

Snapshots

Lifecycle Manager

Network & Security

Load balancers (1/1)

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Filter load balancers

1 match

arn:aws:elasticloadbalancing:us-west-2:175119417300:loadbalancer/net/freyr-aap1-ntwk-lb-dev/c71b19c18a619ab8

Clear filters

1

Name	State	Type	VPC ID	Availability Zones	DNS name
freyr-aap1-ntwk-lb-dev	Active	network	vpc-07dd7c61001d399...	3 Availability Zones	freyr-aap1-ntwk-lb-dev-c71...

Load balancer: freyr-aap1-ntwk-lb-dev

Listeners (1)

A listener checks for connection requests using the protocol and port that you configure. Traffic received by a Network Load Balancer listener is forwarded to the selected target group.

Filter listeners

1

Protocol:Port	Default action	ARN	Security policy	Default SSL/TLS certificate	ALPN
TCP:80	Forward to target group freyr-aap1-ntwk-tg-dev	ARN	Not applicable	Not applicable	None

# Target group with healthy targets and its configs

EC2

Dashboard

EC2 Global View

Events

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

Images

AMIs

AMI Catalog

Elastic Block Store

Volumes

Snapshots

Lifecycle Manager

Network & Security

Target groups

freyr-aap1-ntwk-tg-dev

Details

arn:aws:elasticloadbalancing:us-west-2:175119417300:targetgroup/freyr-aap1-ntwk-tg-dev/3462c8be986e075c

Target type

IP

Protocol : Port

TCP: 8080

VPC

vpc-07dd7c61001d399f5

IP address type

IPv4

Load balancer

freyr-aap1-ntwk-lb-dev

Total targets	Healthy	Unhealthy	Unused	Initial	Draining
1	1	0	0	0	0

Distribution of targets by Availability Zone (AZ)

Select values in this table to see corresponding filters applied to the Registered targets table below.

Targets

Monitoring

Health checks

Attributes

Tags

Health check settings

Protocol

HTTP

Path

/actuator/health

Port

8080

Healthy threshold

3 consecutive health check successes

Unhealthy threshold

3 consecutive health check failures

Timeout

2 seconds

Interval

30 seconds

Success codes

200

# VPC Link integrated with NLB

VPC links

Create

Find APIs

Name
freyr-aap1-vpc-link-dev

REST APIs

VPC link details

freyr-aap1-vpc-link-dev (sxj410)

This VPC link can only be used with REST APIs.

Details

Name (ID)

freyr-aap1-vpc-link-dev (sxj410)

Status

Available

Target NLB

The Network Load Balancer of the VPC targeted by the VPC link.freyr-aap1-ntwk-lb-dev

# Private Rest API

<input type="radio"/>	<a href="#">freyr-aap1-api-dev</a>	freyr-aap1-api-dev rest api	ukjfza7t7i	REST	Private	2025-07-29
-----------------------	------------------------------------	--------------------------------	------------	------	---------	------------

## Proxy resource with VPC link integration

Create resource

/

/(proxy+)

ANY

/{proxy+} - ANY - Method execution

ARN

arn:aws:execute-api:us-west-2:175119417300:ukjfza7t7i:/{proxy+}

Resource ID

lu24a7

Update documentation

Delete

Client

Method request

Integration request

Method response

Integration response

Proxy integration

Method request

Integration request

Integration response

Method response

Test

Integration request settings

Integration type info

VPC link

Paths

proxy

Timeout

1000 ms

VPC proxy integration info

True

Endpoint URL

[http://freyr-aap1-dev.freyr.com/\(proxy\)](http://freyr-aap1-dev.freyr.com/(proxy))

URL path parameters (1)

Name

Mapped from

Caching

proxy

method.request.path.proxy

Inactive

## Tested API with actuator path and empty path with 200 success response

Test method

Make a test call to your method. When you make a test call, API Gateway skips authorization and directly invokes your method.

Method type

GET

Path

proxy

actuator/health

Query strings

param1=value1&param2=value2

Headers

Enter a header name and value separated by a colon (:). Use a new line for each header.

header1:value1  
header2:value2

Client certificate

No client certificates have been generated.

Test

/{proxy+} - ANY method test results

Request

/actuator/health

Latency ms

8

Status

200

Response body

{"status": "UP"}

Response headers

{

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

## App2

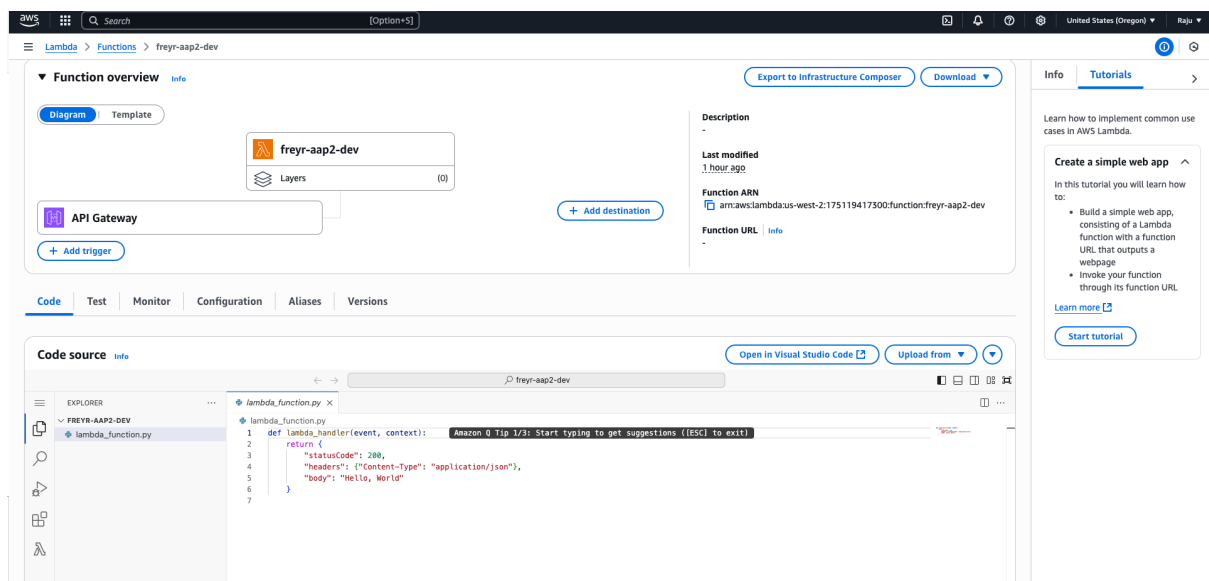
- Sample python lambda code zipped and uploaded to s3
- Deployed the zip file to lambda function
- Added private api gateway with lambda integration and configured lambda triggered

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



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

```
1  {
2    "Version": "2012-10-17",
3    "Statement": [
4      {
5        "Effect": "Allow",
6        "Principal": "*",
7        "Action": "execute-api:Invoke",
8        "Resource": "arn:aws:execute-api:us-west-2:175119417300:jhb0g7w8f4/*",
9        "Condition": {
10         "StringNotEquals": {
11           "aws:aws:SourceVpce": "vpce-08f35f9cde369685a"
12         }
13       }
14     },
15     {
16       "Effect": "Allow",
17       "Principal": "*",
18       "Action": "execute-api:Invoke",
19       "Resource": "arn:aws:execute-api:us-west-2:175119417300:jhb0g7w8f4/*"
20     }
21   ]
22 }
```

## Lambda integration

The screenshot displays the AWS API Gateway console interface. On the left, a sidebar shows the navigation menu with options like API Gateway, Resources, Stages, Authorizers, Gateway responses, Models, Resource policy, Documentation, Dashboard, and API settings. The main content area is titled 'Resources' and shows a list of resources. The selected resource is '/ - ANY - Method execution'. The 'Integration request settings' section is visible, showing the integration type as 'Lambda', the Lambda function as 'freyr-aap2-dev', the region as 'us-west-2', and the timeout as '1000 ms'. The 'URL path parameters' section shows 'No URL path parameters'.

## Invoking lambda by testing API

**Test method**  
Make a test call to your method. When you make a test call, API Gateway skips authorization and directly invokes your method.

**Method type**  
GET

**Query strings**  
param1=value1&param2=value2

**Headers**  
Enter a header name and value separated by a colon (:). Use a new line for each header.  
header1:value1  
header2:value2

**Client certificate**  
No client certificates have been generated.

**Test**

**/ - ANY method test results**

Request	Latency ms	Status
/	287	200

**Response body**  
{ "statusCode": 200, "headers": { "Content-Type": "application/json", "body": "Hello, World!" }

**Response headers**  
{  
 "Content-Type": "application/json",  
 "X-Amzn-Trace-Id": "Root=1-68891ede-7928c7f5a3293c7f6c15ae8a;Parent=3c45acf15d36c532;Sampled=0;Lineage=1:b60f7706:0"  
}

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 gateways (1)** info

Find transit gateway by attribute or tag

Name	Transit gateway ID	State
freyr-dev-tgw	tgw-050933fc51f92a15d	Available

### Transit Gateway Attachments

**Transit gateway attachments (3)** info

Find transit gateway attachment by attribute or tag

Name	Transit gateway attachment ID	Transit gateway ID	State	Resource	Resource ID	Association route table
freyr-dev-vpc1-attachment	tgw-attach-016141e25c99384cf	tgw-050933fc51f92a15d	Available	VPC	ypc-07dd7c61001d399f5	tgw-rtb-0fa72ec2411j
freyr-dev-vpc2-attachment	tgw-attach-090949705b91b06e7	tgw-050933fc51f92a15d	Available	VPC	ypc-0d8f5889a9ffd87a5	tgw-rtb-0fa72ec2411j
freyr-dev-vpc3-attachment	tgw-attach-0bfcca9cda63f82e7	tgw-050933fc51f92a15d	Available	VPC	ypc-0d13e73c179d9ce1c	tgw-rtb-0fa72ec2411j

### Transit Gateway Route tables and its propagations



**Transit gateway route tables (1/1)** [Info](#)

Find transit gateway route table by attribute or tag

[Actions](#) [Create transit gateway route table](#)

<input checked="" type="checkbox"/>	Name	Transit gateway route table ID	Transit gateway ID	State	Default association route table	Default propagation route table
<input checked="" type="checkbox"/>	tgw-rtb-0fa72ec2411a7c098	tgw-rtb-0fa72ec2411a7c098	tgw-050933fc51f92a15d	Available	Yes	Yes

---

**Transit gateway route tables: tgw-rtb-0fa72ec2411a7c098**

Details | Associations | **Propagations** | Prefix list references | Routes | Tags

**Propagations (3)** [Info](#)

Find propagation by attribute or tag

[Delete propagation](#) [Create propagation](#)

<input type="checkbox"/>	Attachment ID	Resource type	Resource ID	State
<input type="checkbox"/>	tgw-attach-016141e25c99384cf	VPC	vpc-07dd7c61001d399f5	Enabled
<input type="checkbox"/>	tgw-attach-090949705b91b06e7	VPC	vpc-0d8f5889a9ff687a5	Enabled
<input type="checkbox"/>	tgw-attach-0bfcca9cda63f82e7	VPC	vpc-0d13e73c179d9ce1c	Enabled

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

Ex:

**Route tables (1/3)** [Info](#)

Find route tables by attribute or tag

[freyr-aap3](#) [Clear filters](#)

Last updated 3 minutes ago [Actions](#) [Create route table](#)

<input type="checkbox"/>	Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC	Owner ID
<input type="checkbox"/>	-	rtb-0097d494b3b846a0a	-	-	Yes	vpc-0d13e73c179d9ce1c   freyr...	175119417300
<input type="checkbox"/>	freyr-aap3-dev-rtb-public	rtb-063545956c98a3827	3 subnets	-	No	vpc-0d13e73c179d9ce1c   freyr...	175119417300
<input checked="" type="checkbox"/>	freyr-aap3-dev-rtb-private	rtb-017129c9b0bc08a75	3 subnets	-	No	vpc-0d13e73c179d9ce1c   freyr...	175119417300

---

**rtb-017129c9b0bc08a75 / freyr-aap3-dev-rtb-private**

Find routes

Destination	Target	Status	Propagated
0.0.0.0/0	nat-0338ac1860dc4b330	Active	No
10.0.0.0/20	tgw-050933fc51f92a15d	Active	No
10.0.32.0/20	tgw-050933fc51f92a15d	Active	No
10.0.64.0/20	local	Active	No

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://jh0g7w8f4-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 ~]# 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 securely connect internally we shall be using vpc endpoints by allowing certain cidr's/ip's in the security groups