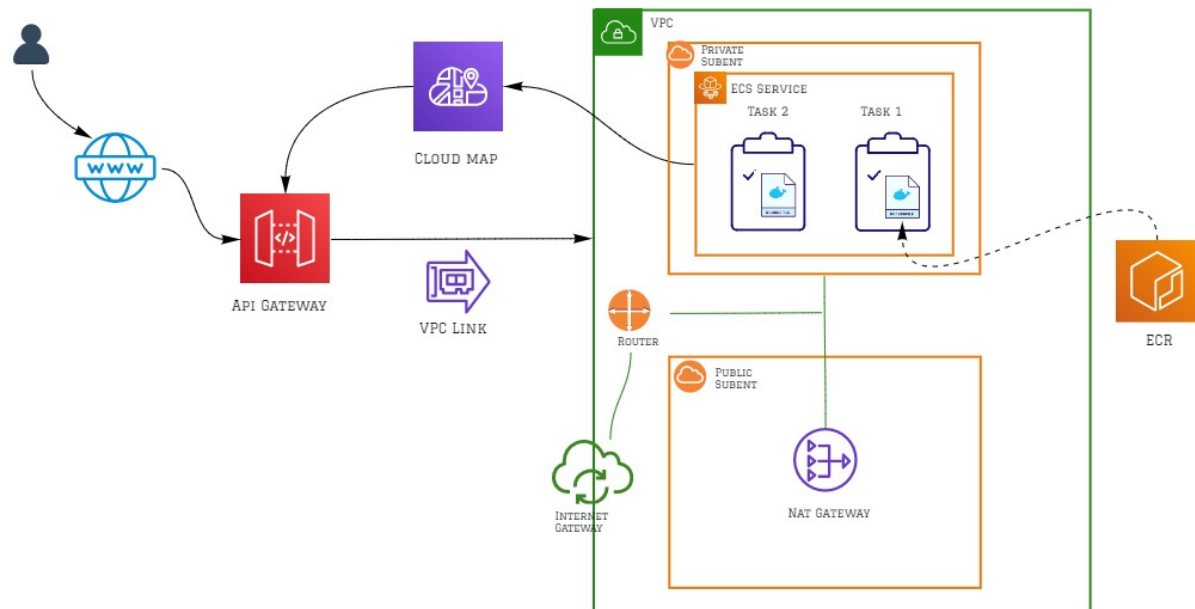


## Innablr Technical Challenge

## Table of Contents

Architecture Overview	3
Folder Structure	3
Deployment	5
Terraform Backend Configuration	5
Deployment Pipeline	5
Variables Passing	6
Testing	7
Enhancements	8

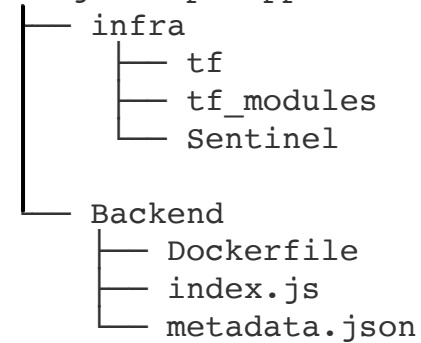
## Architecture Overview



miro

## Folder Structure

fargate-api-app




- Backend:
  - A simple dockerfile that has only one stage based on slim node image. It just copies the application source and runs npm install.
  - Index.js just displays hello world message with a random number:

```
< > ↻ 0u4gtdvy42.execute-api.ap-southeast-2.amazonaws.com
{"Hello World! you are visotry number:":32}
```

- The Image created by the dockerfile is pushed to GitHub ghcr. The Image URI and Image Name are provided to fargatecluster to create a service in the cluster.

## fargate-api-app Private

 Install from the command line

[Learn more about packages](#)

```
$ docker pull ghcr.io/rna87/fargate-api-app:main
```

### Details

 rna87

 fargate-api-app

 Readme

[Clusters](#) > [innablr-production-cluster](#) > Task: 7309ef5d45e846d3bdc94847eb8cb587

### Task : 7309ef5d45e846d3bdc94847eb8cb587

Details
Tags
Logs

Filter logs

⌵

All

30s

5m

1h

6h

1d

1w

Timestamp (UTC+00:00)	Message
2022-11-07 07:40:46	This hello-world-app is listening at https://0u4gtdvy42.execute-api.ap-southeast-2.amazonaws.com/
This hello-world-app is listening at https://0u4gtdvy42.execute-api.ap-southeast-2.amazonaws.com/	
2022-11-07 07:40:46	> hello-world-app@1.0.0 start /usr/src/app
> hello-world-app@1.0.0 start /usr/src/app	
2022-11-07 07:40:46	> node index.js
> node index.js	

Last updated on November 7, 2022

- infra/tf\_modules

This folder contains two terraform modules:

#### 1. networking module

- It basically creates the networking infrastructure for the cluster.
- Creates 1 public subnets, and 1 private subnet, internet gateway and nat Gateway
- It creates a VPC Link: this aws service is managed by api gateway. So api gateway can talk privately to the assigned resources in the VPC (the ECS container-based app). [VPC Link](#)
- It creates an AWS Cloud Map: this is for service discovery and its enabled in the ecs service.
- It creates a security group for the VPC Link.

## 2. fargate\_cluster module

- It creates a ECS cluster, fargate ECS service, two tasks:

Clusters > innablr-production-cluster

Cluster : innablr-production-cluster [Update Cluster](#) [Delete Cluster](#)

Get a detailed view of the resources on your cluster.

Cluster ARN: `arn:aws:ecs:ap-southeast-2:478525466663:cluster/innablr-production-cluster`

Status: **ACTIVE**

Registered container instances: 0

Pending tasks count: 0 Fargate, 0 EC2, 0 External

Running tasks count: 2 Fargate, 0 EC2, 0 External

Active service count: 1 Fargate, 0 EC2, 0 External

Draining service count: 0 Fargate, 0 EC2, 0 External

Services | Tasks | ECS Instances | Metrics | Scheduled Tasks | Tags | Capacity Providers

Create Update Delete Actions

Filter in this page Launch type ALL Service type ALL < 1-1 >

Service Name	Status	Service type	Task Definition	Desired tasks	Running tasks	Launch type	Platform version
innablr-production	ACTIVE	REPLICA	innablr-production:12	2	2	FARGATE	LATEST(1.4.0)

These two modules are used in `infra/tf/main.tf`. The output of the networking modules is used in `fargate_clster` module.

```
rana-elwakil7, 2 weeks ago | 1 author (rana-elwakil7)
module "networking" {
  source = "../tf_modules/networking"
  aws_region = local.config["aws_region"]
  environment = local.config["environment"]
  project_name = local.config["project_name"]
  vpc_cidr_block = local.config["vpc_cidr_block"]
}

You, 15 hours ago | 2 authors (rana-elwakil7 and others)
module "fargate_cluster" {
  source = "../tf_modules/fargate_cluster"
  aws_region = local.config["aws_region"]
  environment = local.config["environment"]
  project_name = local.config["project_name"]
  vpc_cidr_block = local.config["vpc_cidr_block"]
  fargate_namespace_id = module.networking.fargate_namespace_id
  container_name = local.config["container_name"]
  app_port = local.config["app_port"]
  image_uri = local.config["image_uri"]
  image_secret = local.config["image_secret"]
  last_sha = local.config["last_sha"]

  vpc_id = module.networking.vpc_id
  private_subnet_ids = [ module.networking.private_subnet_id ]
  network_stack_vpclink_id = module.networking.vpc_link_id
  vpc_link_sg = module.networking.vpc_link_sg
}
```

## Deployment

### Terraform Backend Configuration

To use sentinel tests I have used terraform cloud.

### Deployment Pipeline

Github actions pipeline has two jobs that run sequentially:

Build and push image job (build docker image and publishes it to GitHub ghcr) and infra apply job for planning and applying terraform resources.

I have chose to create two different jobs as its easier to add conditions later on the project on how these jobs are triggered. We might want to only publish docker images on creating github releases.

## Variables Passing

Variable are initialised in `infra/terraform/config.yaml`

```
You, 15 hours ago | 2 authors (rana-elwakil7 and others)
1  aws_region: "ap-southeast-2"
2  environment: "production"
3  project_name: "innablr"
4  vpc_cidr_block: "192.168.0.0/16"
5  container_name: "api"
6  app_port: "8080"
7  image_uri: "ghcr.io/rna87/fargate-api-app:main"
8  image_secret: "arn:aws:secretsmanager:ap-southeast-2:478525466663:secret:github-actions-qSj15"
```

As per requirements, last commit Sha is getting passed to terraform config:

```
- name: Getting Last Commit SHA
  run: echo "last_sha:" ${github.sha} >> config.yaml
```

That in turn passed as a environment variable to the container

← → ↺ 0u4gtdvy42.execute-api.ap-southeast-2.amazonaws.com/status

```
{ "hello-world-app": { "version": "1.0.0", "description": "Hello World Micro Service Challenge", "sha": "40b75505f232f420de840d0822310402b0c7003a" } }
```

ap-southeast-2.console.aws.amazon.com/ec2/home?region=ap-southeast-2#clusters/innablr-production-cluster/tasks/3bf0c6e2b4e84b63bd94080c0b71c0/details

Network mode: aws-vpc

ENI Id: eni-074895a3a3a9000e6

Subnet Id: subnet-0ea90551a636fd573

Private IP: 192.168.1.183

Public IP: --

Mac address: 02:34:3b:34:16:50

Containers

Last updated on November 7, 2022 9:12:43 AM (1m ago)

Name	Container Runtime I...	Status	Image	Image Digest	CPU Units	Hard/Soft memory ...	Essential	Resource ID
api	3bf0c6e2b4e84b63bd...	RUNNING	ghcr.io/rna87/fargate-api-app:main		--	--/--	true	1f89e057-391d-4c5...

Details

Network bindings - not configured

Environment Variables

Key	Value
ENV	production
LASTCOMMIT	40b75505f232f420de840d0822310402b0c7003a
PORT	8080
URL	https://0u4gtdvy42.execute-api.ap-southeast-2.amazonaws.com/

As for description and version variables, they are supplied through a metadata file: `backend/metadata.json`

## Testing

I have used sentinel and created 3 test cases:

- enforce-ssh-disabled

To check for any security group that have SSH open to CIDR "0.0.0.0/0" for ingress rules.

- limit-cost-and-percentage-increase

This policy restricts both the total monthly cost and the percentage increase in the monthly cost that would be incurred if the current plan were applied

- vpc-dns-support

This policy checks that the VPC supports DNS


I have downloaded the mock data from terraform cloud and test these policies locally:

Then created a policy set in terraform cloud to run these policies before terraform apply:

```
"""
"-----",
"""
"\u001b[1mOrganization Policy Check:",
"""
"===== Results for policy set: <empty policy set name> =====",
"""
"Sentinel Result: true",
"""
"This result means that all Sentinel policies passed and the protected",
"behavior is allowed.",
"""
"3 policies evaluated.",
"""
"## Policy 1: vpc-dns-support (soft-mandatory)",
"""
"Result: true",
"""
"./vpc-dns-support.sentinel:6:1 - Rule \"main\"",
"Value:",
"true",
"""
"## Policy 2: limit-cost-and-percentage-increase (soft-mandatory)",
"""
"Result: true",
"""
"./limit-cost-and-percentage-increase.sentinel:6:1 - Rule \"main\"",
"Value:",
"true",
"""
"## Policy 3: enforce-ssh-disabled (soft-mandatory)",
"""
"Result: true",
"""
"./enforce-ssh-disabled.sentinel:6:1 - Rule \"main\"",
"Value:",
"true"
```

Output from terraform UI:

✓ Planned and finished Triggered via CLI CURRENT

 RNA triggered a run from CLI 28 minutes ago Run Details

✓ Plan finished 27 minutes ago

✓ Cost estimation finished 27 minutes ago Resources: 0 of 9 estimated - \$0.00/mo +\$0.00

✓ Policy check passed 27 minutes ago Policies: 3 passed, 0 failed

Queued 27 minutes ago -> Passed 27 minutes ago

✓ passed vpc-dns-support

✓ passed limit-cost-and-percentage-increase

✓ passed enforce-ssh-disabled

View Logs View JSON Data

## Enhancements

- Design first. It's a lot cheaper to invest on spec first and iterate on it. Should we decide from the beginning to divide the backend and frontend component? It really depends on the teams that are working on the project. But it's important to distinguish these components from build and deploy point of view.
- Using git: I haven't really used git "effectively" in terms of branching. Also, I would like to make more smaller commits as it will make it easier to revert code to a previous state and write meaningful commits.
- For the IAM roles: use a resource-based security policies. And using a role for ansible deployments.
- For the docker file: We used the same file for test and prod. In a more complex application, its recommended to add another step to create a minimal Docker image that only consists of the application source, modules and Nodejs runtime.