Platform Engineering Service

18.09.2023

—

Nagarajan Govindarajan

Senior Cloud Engineer

BLK56 Toa Payoh

Singapore, 310056

Overview

There are two parts to this take home exercise which is meant to challenge you on the aspects of the cloud infrastructure engineer role that you might encounter day to day.

The first part of the exercise is a programming exercise to write a small bespoke tool to perform some tasks.

The field of cloud computing is still very nascent and new. The ecosystem is still developing and there are many new tools that are not mature yet. Frequently, we might encounter some weird behaviour or bugs in the tools that we use. It is also common to encounter situations where there is no tool that performs exactly what we need, or the tool that we are using is missing certain features. Sometimes, it is possible to cobble together a myriad of tools to perform our tasks, but sometimes it's not.

It is therefore useful to be able to read and write code so that we can help to figure out where bugs or problems mighthave arisen from in the tools. We can then write detailed and useful bug reports for the maintainers to work on. If wehave the time, we might even be able write code ourselves and contribute to the open source community.

The second part of the exercise is to deploy the tool you have written in the first part to a production cloud environment.

The core part of the role is to run and maintain the cloud infrastructure and services and applications running on top of the infrastructure. It is important that the applications and services take advantage of the cloud environment as much as possible to provide the necessary performance and availability to fulfil business needs.

Goals

1. Please perform the tasks described in the next section.
2. Document instructions on how to execute and use your solution.
3. Think through the design and take note of the various decisions and tradeoffs that you make.
4. You will be askedquestions during your interview for discussion.
5. Archive your solution and email them to the hiring manager.

Programming Task

Using a programming language of your choice, implement a long-running process that:

Accepts a csv file containing a list of up to 1000 urls with names at startup.

The process should pull all these urls every 10 minutes to check their HTTP status.

The process should also bind a local port, to provide a summary of monitoring status in the past hour in anysuitable format.

Containerise the application

A sample of the CSV file might include:

name, url

google, https://www.google.com

...

sph, https://www.sph.com.sg

Deployment Task

Provide a solution in the form of code using

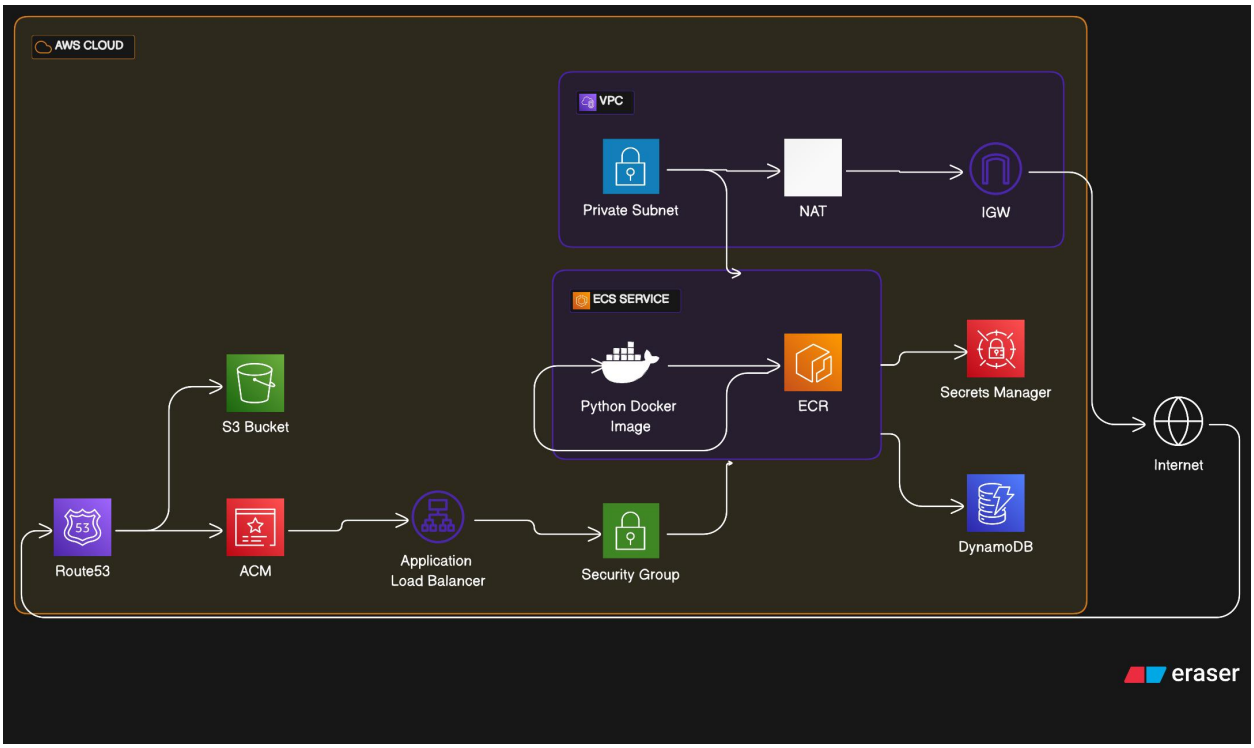
Terraform

to deploy the containerised application you have written above to acloud provider of your choice. Your solution should include setting up the cloud provider account so that the applicationcan be deployed.

Technology Stack

- **Programming task - Python**
- **Infrastructure as code - Terraform(HCL)**
- **Cloud - AWS (ECS, ALB, Security Groups, Secrets, VPC, NAT, IGW, s3, DynamoDb, IAM, ACM, CloudWatch, Route53)**
- **Web app - Typescripts, Angular, Html5, CSS**

Target Architecture



How it works

ECS Service with FastAPI and Health Check

- **Environment**: AWS ECS with Fargate launch type, running within a private subnet.
- **Language**: The application is written in Python, using the FastAPI framework.
- **Task**: The ECS task is scheduled to run health check jobs on URLs every 10 minutes.
- **CSV File**: A CSV file containing the list of URLs is bundled within the Docker image.
- **Docker Image**: The Docker image containing the FastAPI application and the CSV file is hosted in a public Amazon ECR (Elastic Container Registry).

DynamoDB for Storing Summary

- **Database**: AWS DynamoDB is used to store the summary of health checks.
- **Data Model**: The table schema is designed to accommodate summary details like time, status, and status code count.

Summary APIs and ALB

- **APIs**: The ECS service exposes APIs that return health check summaries.
- **Load Balancer**: An AWS ALB distributes incoming API requests across ECS tasks.
- **Route 53**: Domain name resolution and routing are handled via AWS Route 53.
- **SSL**: SSL certificates are installed to secure the API requests and responses.

Angular App in S3

- **Static Site**: An Angular application is hosted in a public S3 bucket.
- **Functionality**: The app fetches health summary details via the secure APIs exposed by the ECS service.

Security Measures

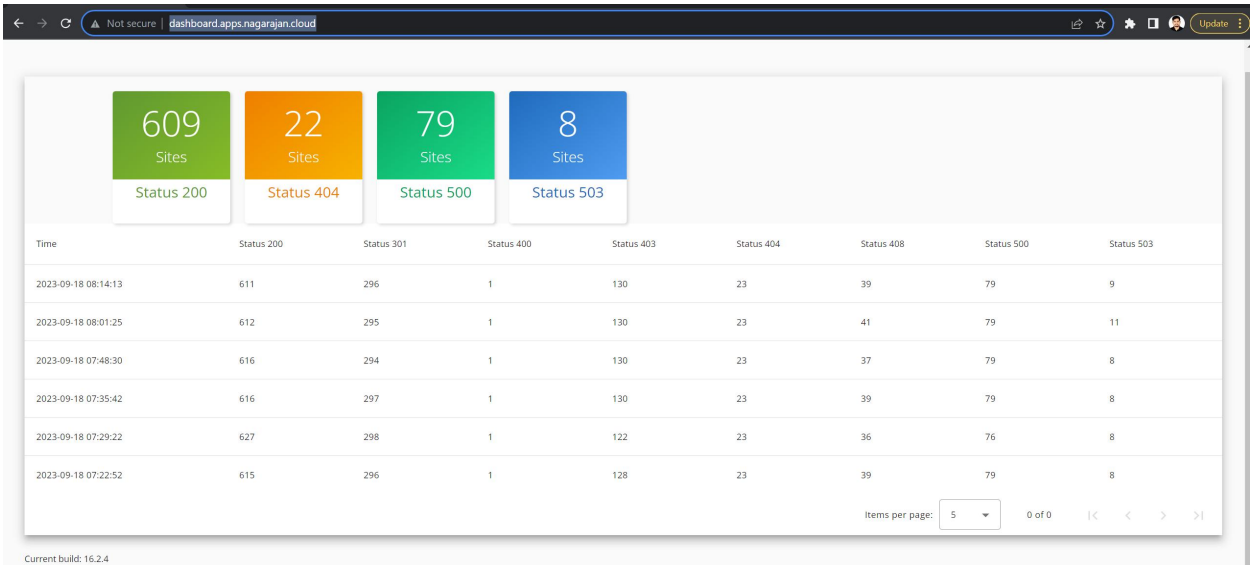
- **VPC and Private Subnet**: The ECS tasks are run within a VPC and a private subnet for added security.
- **SSL**: All traffic between the Angular app and the APIs are encrypted using SSL certificates.

In summary, this architecture offers a scalable, secure, and efficient solution for performing periodic health checks on URLs and exposing these summaries through an Angular app hosted on S3.

Live Demo

Please visit below url for live Demo (Angular app fetching health summary from live apis)

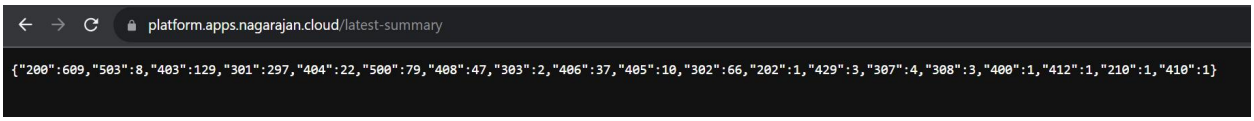
http://dashboard.apps.nagarajan.cloud



Api calls to ECS via LB

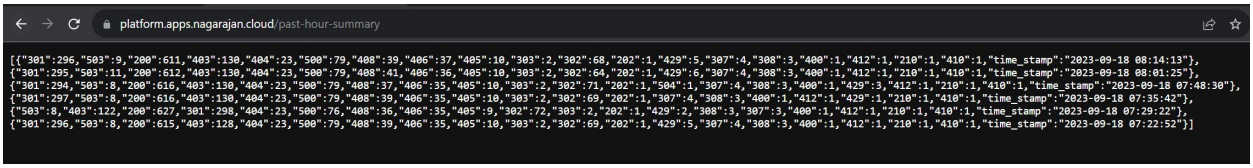
1.Latest websites health summary

https://platform.apps.nagarajan.cloud/latest-summary



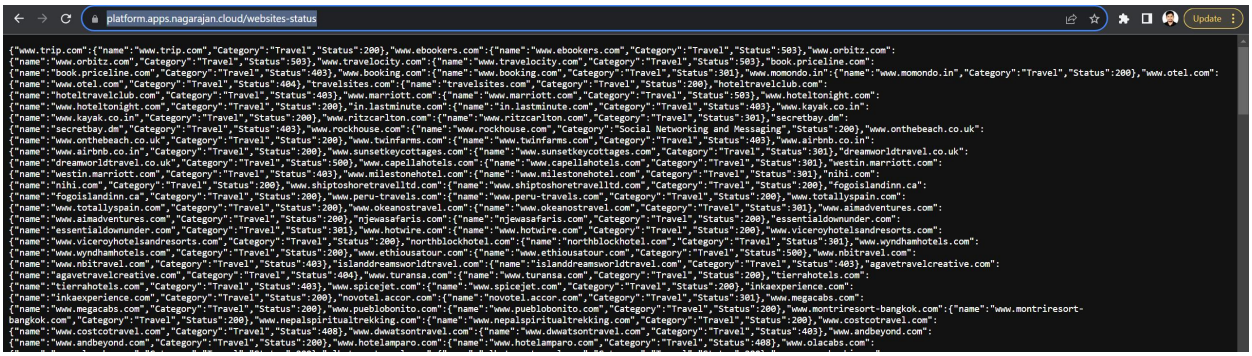
2. Last one hour health summary

https://platform.apps.nagarajan.cloud/past-hour-summary



3. All websites latest health

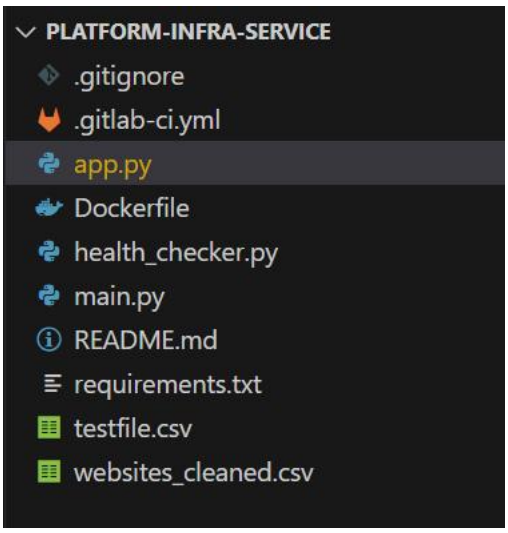
https://platform.apps.nagarajan.cloud/websites-status



Source Code Structure

1.Health check task service python code

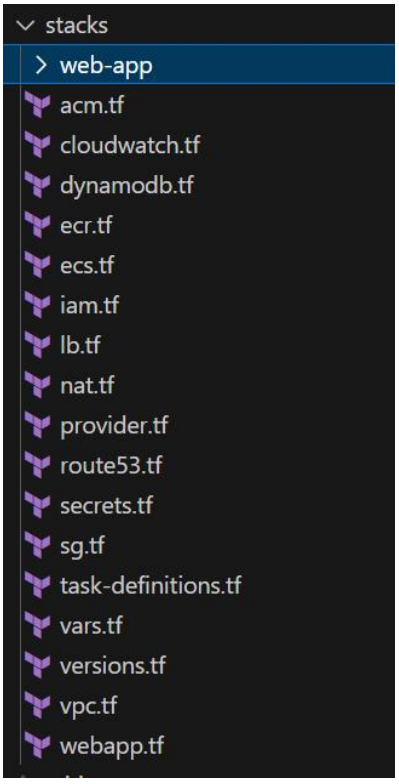
https://github.com/nagarajyadava/platform-infra-service



- Use main.py for testing in local
- Gitlab ci for making docker image using gitlab
- app.py for fastApi controller

2. Infrastructre As Code

https://github.com/nagarajyadava/platform-infra-iac



How to deploy it?

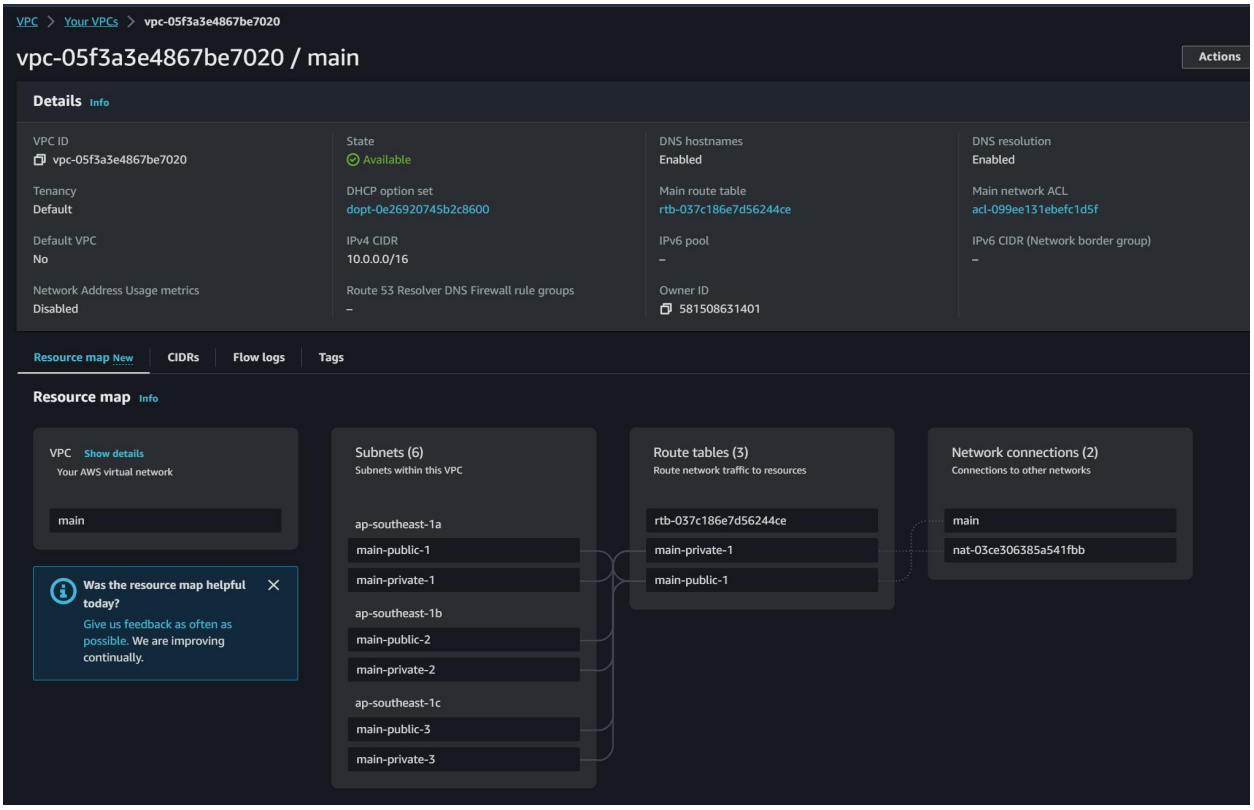
- Create a IAM user with below access

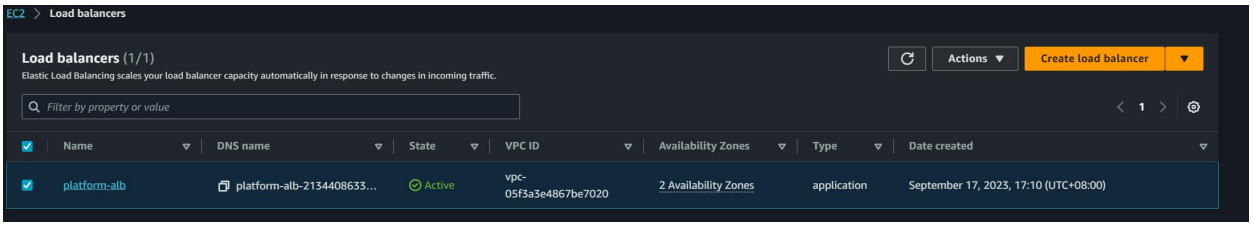
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PlatformEngAccess",
      "Effect": "Allow",
      "Action": [
        "s3:*", "cloudwatch:*", "elasticloadbalancing:*", "acm:*", "ec2:*", "route53:*", "route53domains:*", "dynamodb:*",
"ecr-public:*", "secretsmanager:*", "ecs:*", "ecr:*"
      ],
      "Resource": "*"
    }
  ]
}
```

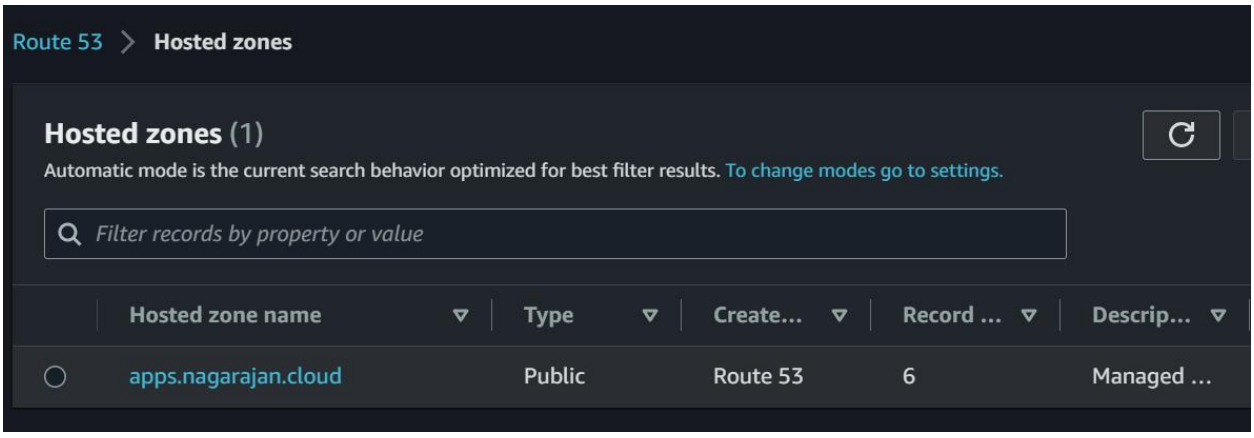
- 'aws configure' with above user's access key and id
- cd stacks
- Before deploying this iac please change below domain names to your domain names in stacks/var.tf

```
variable "AWS_REGION" {
  default = "ap-southeast-1"
}

variable "MAIN_HOST" {
  default = "apps.nagarajan.cloud"
}

variable "ACM_DOMAIN" {
  default = "*.apps.nagarajan.cloud"
}

variable "ALB_DOMAIN" {
  default = "platform.apps.nagarajan.cloud"
}

variable "WEB_APP_DOMAIN" {
  default = "dashboard.apps.nagarajan.cloud"
}
```

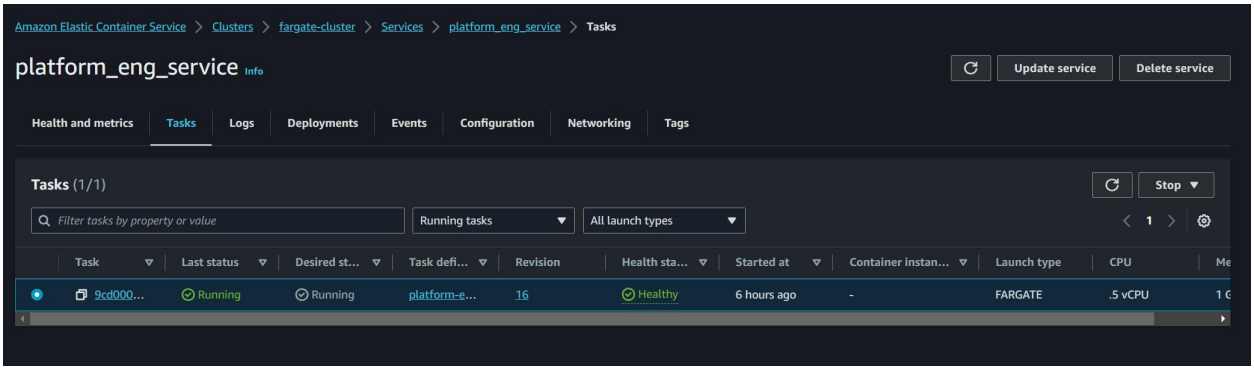
- terraform init
- terraform plan
- terraform apply
- Now all stacks will be created in above aws account
- Note:Web-app folder have angular app dist files.

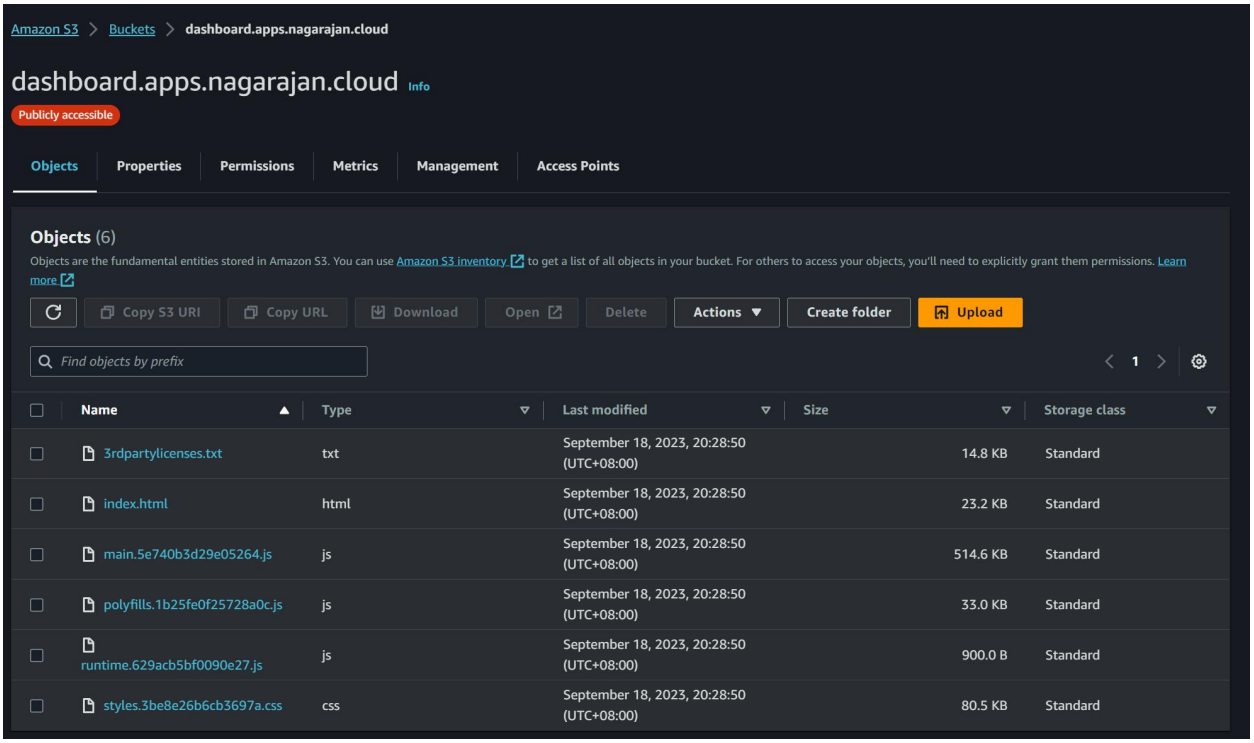

VPC > Your VPCs > vpc-05f3a3e4867be7020
vpc-05f3a3e4867be7020 / main
Actions
Details Info
VPC ID
vpc-05f3a3e4867be7020
State
Available
DNS hostnames
Enabled
DNS resolution
Enabled
Tenancy
Default
DHCP option set
dopt-0e26920745b2c8600
Main route table
rtb-037c186e7d56244ce
Main network ACL
acl-099ee131ebefc1d5f
Default VPC
No
IPv4 CIDR
10.0.0.0/16
IPv6 pool
–
IPv6 CIDR (Network border group)
–
Network Address Usage metrics
Disabled
Route 53 Resolver DNS Firewall rule groups
–
Owner ID
581508631401
Resource map New
CIDRs
Flow logs
Tags
Resource map Info
VPC Show details
Your AWS virtual network
main
Was the resource map helpful today?
Give us feedback as often as possible. We are improving continually.
Subnets (6)
Subnets within this VPC
ap-southeast-1a
main-public-1
main-private-1
ap-southeast-1b
main-public-2
main-private-2
ap-southeast-1c
main-public-3
main-private-3
Route tables (3)
Route network traffic to resources
rtb-037c186e7d56244ce
main-private-1
main-public-1
Network connections (2)
Connections to other networks
main
nat-03ce306385a541fbb

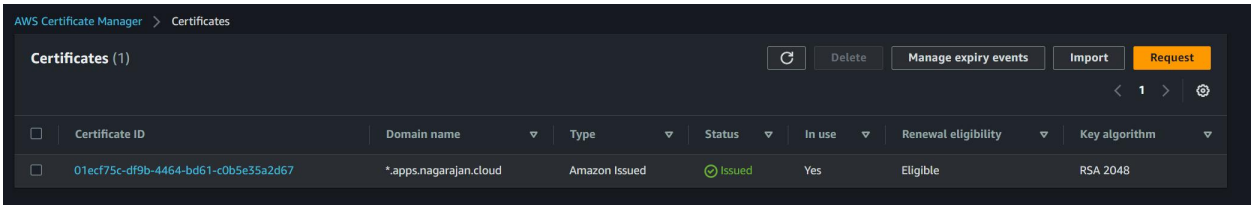

EC2 > Load balancers
Load balancers (1/1)
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.
Actions
Create load balancer
Filter by property or value
1
Name
DNS name
State
VPC ID
Availability Zones
Type
Date created
platform-alb
platform-alb-2134408633...
Active
vpc-05f3a3e4867be7020
2 Availability Zones
application
September 17, 2023, 17:10 (UTC+08:00)

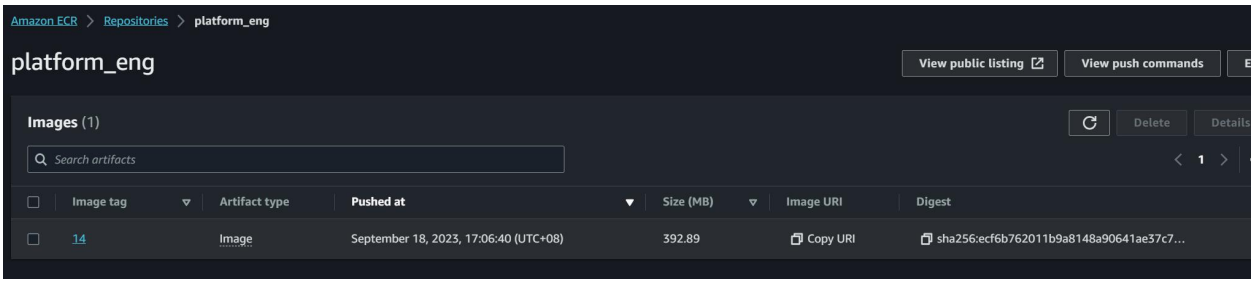

Route 53 > Hosted zones
Hosted zones (1)
Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.
Filter records by property or value
Hosted zone name
Type
Create...
Record ...
Descrip...
apps.nagarajan.cloud
Public
Route 53
6
Managed ...

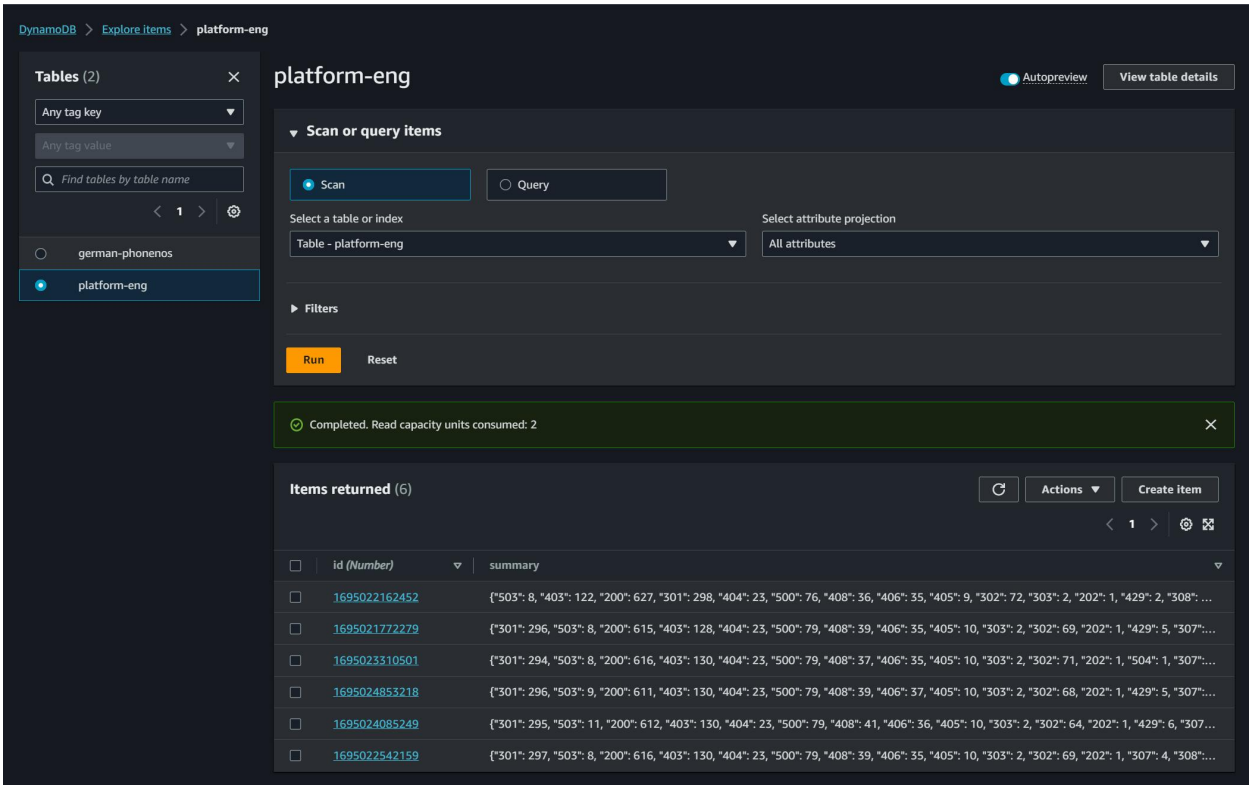

Security Groups (3) Info
Actions
Filter security groups
Name
Security group ID
Security group name
VPC ID
Description
lb-sg
sg-063b592fafa2faf11
lb-sg
vpc-05f3a3e4867be7020
security group for load...
–
sg-00e969dae628183af
default
vpc-05f3a3e4867be7020
default VPC security gr...
ecs-service-sg
sg-06d62824db91094bf
ecs-service-sg
vpc-05f3a3e4867be7020
security group for ecs


Amazon Elastic Container Service > Clusters > fargate-cluster > Services > platform_eng_service > Tasks
platform_eng_service Info
Update service
Delete service
Health and metrics
Tasks
Logs
Deployments
Events
Configuration
Networking
Tags
Tasks (1/1)
Stop
Filter tasks by property or value
Running tasks
All launch types
Task
Last status
Desired st...
Task defi...
Revision
Health sta...
Started at
Container instan...
Launch type
CPU
9cd000...
Running
Running
platform-e...
16
Healthy
6 hours ago
-
FARGATE
.5 vCPU


Amazon S3 > Buckets > dashboard.apps.nagarajan.cloud
dashboard.apps.nagarajan.cloud Info
Publicly accessible
Objects
Properties
Permissions
Metrics
Management
Access Points
Objects (6)
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Find objects by prefix
Name
Type
Last modified
Size
Storage class
3rdpartylicenses.txt
txt
September 18, 2023, 20:28:50 (UTC+08:00)
14.8 KB
Standard
index.html
html
September 18, 2023, 20:28:50 (UTC+08:00)
23.2 KB
Standard
main.5e740b3d29e05264.js
js
September 18, 2023, 20:28:50 (UTC+08:00)
514.6 KB
Standard
polyfills.1b25fe0f25728a0c.js
js
September 18, 2023, 20:28:50 (UTC+08:00)
33.0 KB
Standard
runtime.629acb5bf0090e27.js
js
September 18, 2023, 20:28:50 (UTC+08:00)
900.0 B
Standard
styles.3be8e26b6cb3697a.css
css
September 18, 2023, 20:28:50 (UTC+08:00)
80.5 KB
Standard

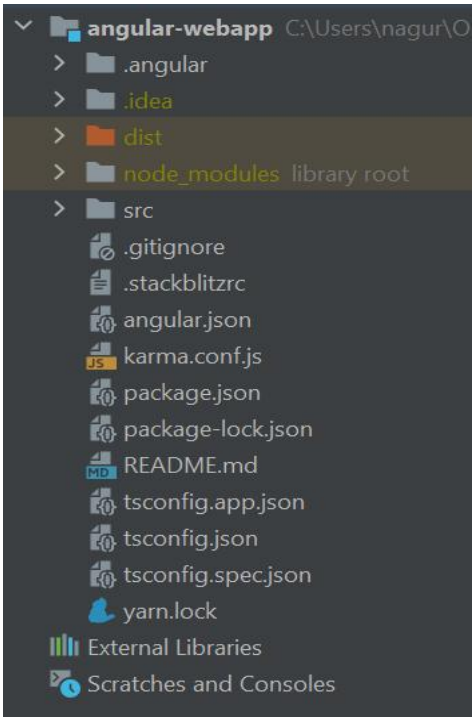

AWS Certificate Manager > Certificates
Certificates (1)
Delete
Manage expiry events
Import
Request
Certificate ID
Domain name
Type
Status
In use
Renewal eligibility
Key algorithm
01ecf75c-df9b-4464-bd61-c0b5e35a2d67
*.apps.nagarajan.cloud
Amazon Issued
Issued
Yes
Eligible
RSA 2048


Amazon ECR > Repositories > platform_eng
platform_eng
View public listing
View push commands
Images (1)
Delete
Search artifacts
Image tag
Artifact type
Pushed at
Size (MB)
Image URI
Digest
14
Image
September 18, 2023, 17:06:40 (UTC+08)
392.89
Copy URI
sha256:ecf6b762011b9a8148a90641ae37c7...



3. Angular web app

https://github.com/nagarajyadava/angular-webapp



- npm install
- npm update
- ‘ng build’ to generate dist files
- Copy dist files into iac-code/stacks/web-app to deploy