Fuel Quota

https://github.com/shashperera/FuelQuota

Name **:** W.S.S.Perera

Sri Lanka Institute of Information Technology

Sri Lanka

Table of Contents

List of Figures

1 Outline of the project

FuelQuota is a scalable QR solution to track fuel quotas and designed with 3 APIS following microservices architecture. Vehicle API will generate a QR code with vehicle number and GUID generated for the vehicle. Then Register vehicle details and update the Quota. Also there is a third-party API to get the registration information. Quota API can update quotas, reduce the quota, and add initial quota. Finally Authentication API to authenticate users.

A mocked Serverless Third Party API on AWS Lambda is used with the CSV file loaded onto DynamoDB. AWS Fargate is used to configure the services and Cloudformation stack can deploy the entire infrastructure on the cloud. JMeter load test can take multiple GET endpoints and test the load.

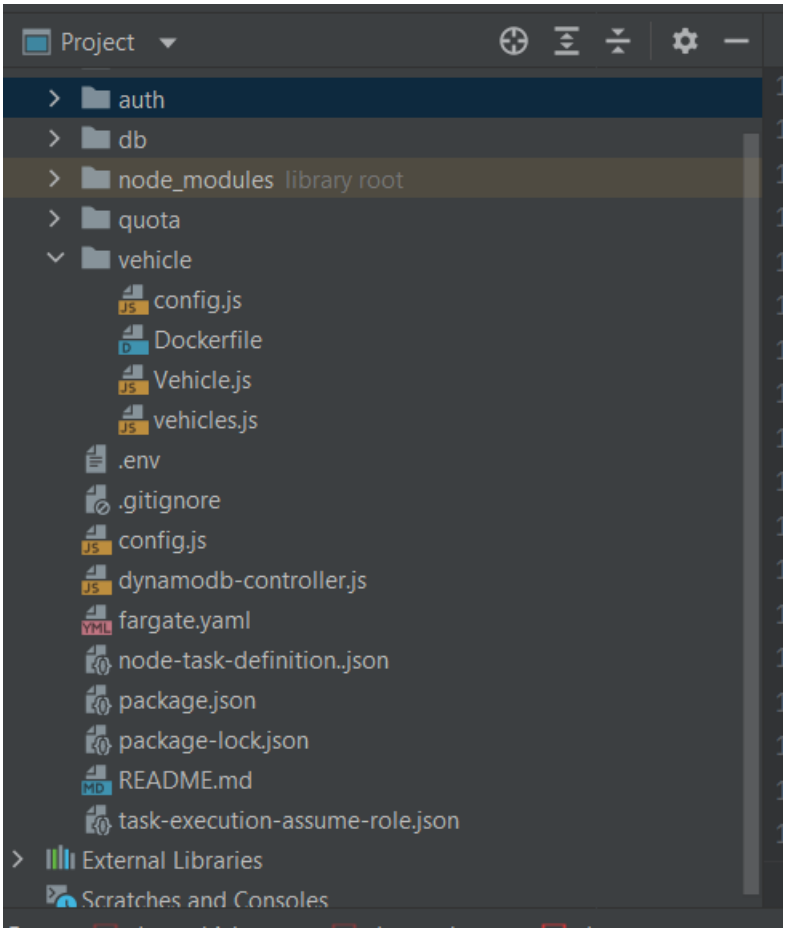


Figure 1.1: Project Structure

2 Architecture diagrams : Solution Design

https://drive.google.com/file/d/16izitGPRiiwpkXH77D7YhUx8FOgvJ2sD/view?usp=sharing

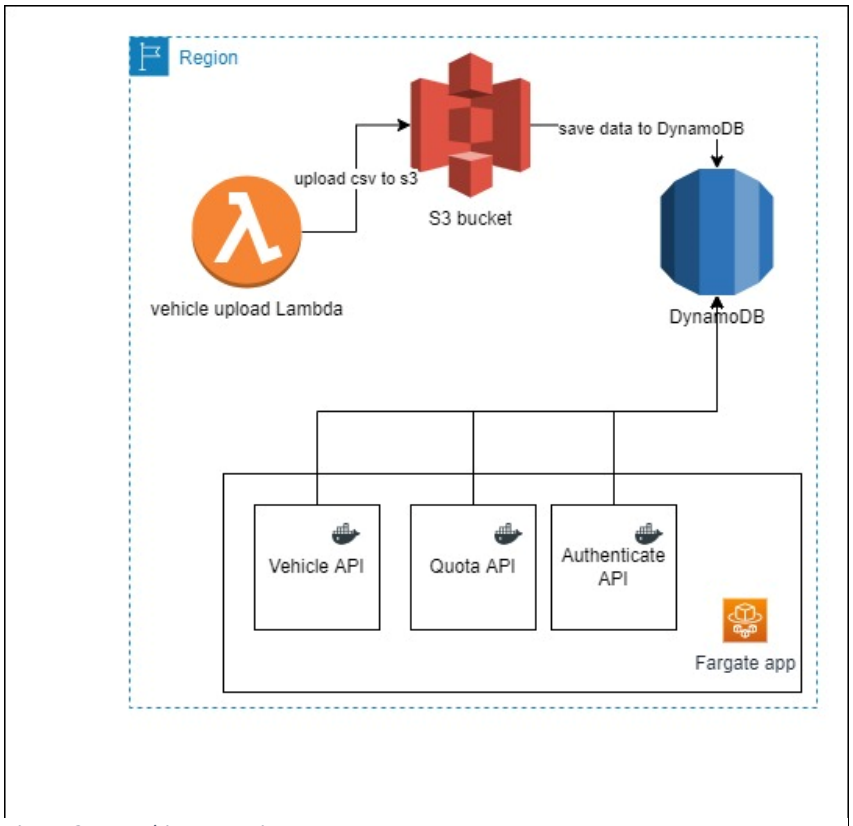


Figure 2.1: Architecture Diagram

Consider the regions in AWS cloud. The cost depends on the region we select.

3 Microservices (Nodejs)

1. Microservices running on port 3000,5000 and 9000.

```
dev-vehicle   dev-auth   dev-quota
> nodemon quota/quotas.js

[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node quota/quotas.js`
Up and Running on port 9000 - This is Quota service
Connection successfull!
```

Figure 3.1 : : Microservices

POST http://localhost:5000/user

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
{
    "admin":true,
    "first_name" : "Malika",
    "last_name" : "Perera",
    "email" : "sh@gmail.com",
    "password" : "1234"
}
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview HTML

```
New Customer created successfully!
```

Figure 3.2: User authentication port:5000

POST http://localhost:3000/vregister

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)


```
{
    "vehicleRegistrationNumber" : "GP8952",
    "registeredDate" : "26/08/2016",
    "chassisNumber" : 78542
}
```


Body Cookies Headers (7) Test Results

Pretty Raw Preview JSON


```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
        .eyJ2ZWhpY2xlX2lkIjoiNjMzZmY3NDU4Y2FiZWMyMzhjOWMzM2U1IiwidmVoaWNsZVJlZ2lzdHJhdGlvbk51bWJlciI6IkdQODk1MiIsI
        NjUzfQ.sVDdRRhCRM-M3Sj2WfyUQcc40w_5DTA9i78SvK2GGY4"
}
```


Figure 3.4: Vehicle API port:3000

POST http://localhost:9000/quotaV/generate Params Send Save

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)


```
{
    "vehicleID":"633e6e56fec5503ed0ff4f11",
    "maxQuota":50,
    "remainingQuota":30
}
```


Body Cookies Headers (7) Test Results Status: 200 OK Time: 122 m

Pretty Raw Preview JSON Save Respons


```
{
    "dataImage": "
        /M8TJXDxBU5ekvZIT9Yq31Chdrrde4WGu9xsVa6zUu1lqvcbHWeo2LtdZrXKy1XuNirfUaF2ut17hYa73GxVrrNS7WWq9xsdZ6jYu11mtcrLVe44eHVP5SxaQyVZyonFScqEwVk8pU8YTKE
        xWTyhMVk8pUcaJyUnGiMlVMKicVd6j8pYonLtZar3Gx1nqNi7XWa
        /zwYRWfpPJNFXdUTCpPqJxUTCp3qEwVJyonKlPFpPKEylQxVUwqd6hMFXdUfJLKJ12stV7jYq31Ghdrrdf44ctU7qi4o
        +JE5UT1pGJSmSqeqLijY1KZKu5QmSpOVCaVqeJEZaqYVCaVqeKk4kTlk1TuqPimi7XWa1ystV7jYq31Gj/8x61MFVPficpU8YTKVDGpTBV3qEwVJyonFZPKHRVPqEwVk8pJxaRyUjGpTBX
        /ZRdrrde4WGu9xsVa6zV++I+rmFSmijtUpopPqphUpoo7VJ5QmSpOVCaVk4qp4kT1DpUnKt7kYq31Ghdrrde4WGu9xg9fVvGXKiaVOyoml5dUpoqpY1KZKqaKSWWquEP1RGWquEP1pOKkY1
        L511T8Sy7WWq9xsdZ6jYu11mv88GEqf011qjipmFTuqJhUpopJ5UR1qphUpoo7VKaKT1KZKk4qJpWpY1KZKiaVqWJSOVGZKk5U
        /mUXa63XuFhrvcbFWus1fnio419WcVLxSSqfVPFExR0qU8WkM1WcVJxUPFFxh8odFf81F2ut17hYa73GxVrrNX54SGWqmFROKiaVOyqeUJkqpoonKiaVqWJSeULlmypOVKaKSWWquEP1iYp
        J5UR1qjhRmSom1ZOKJy7WWq9xsdZ6jYu11mvYL75I5aTiDpWTiknlpGJSeaLiRGWqeEJlqphUpopJZaqYVE4qPk1lqrhD5aTim1ROKiaVqeKJi7XWa1ystV7jYq31Gj98mMpUMa1MK1PFpD
        JVTCqTy1RxonJScaLyhMpUMa1MFScqd1TcUTGpTBV3qJyo
```


Figure 3.3 Quota API port:9000

Figure 3.5: Generate QR using vehicle Reg no and GUID

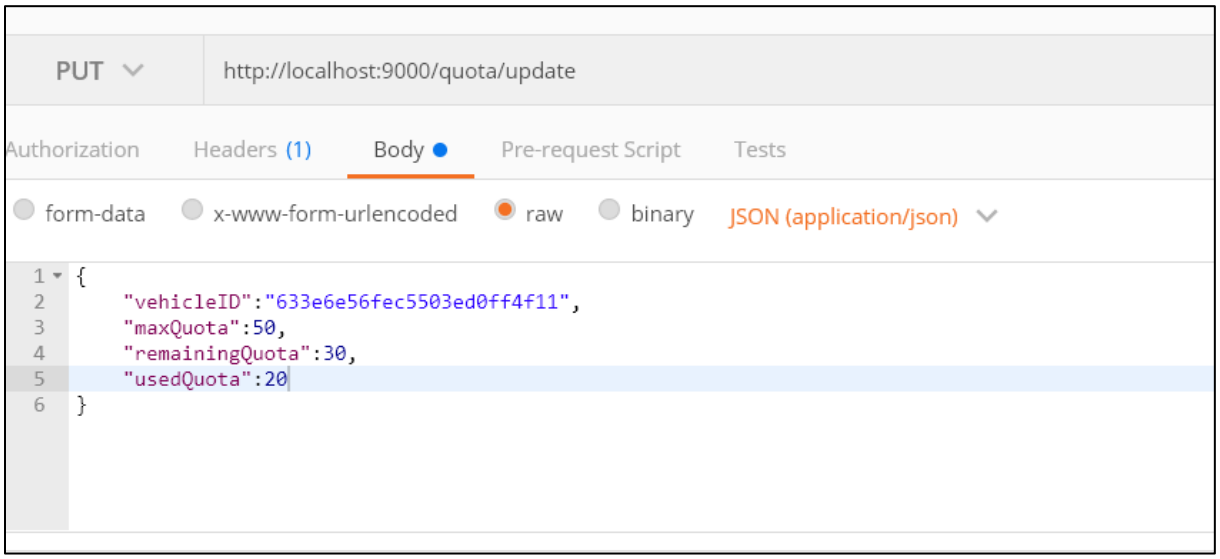


Figure 3.6: PUT Quota

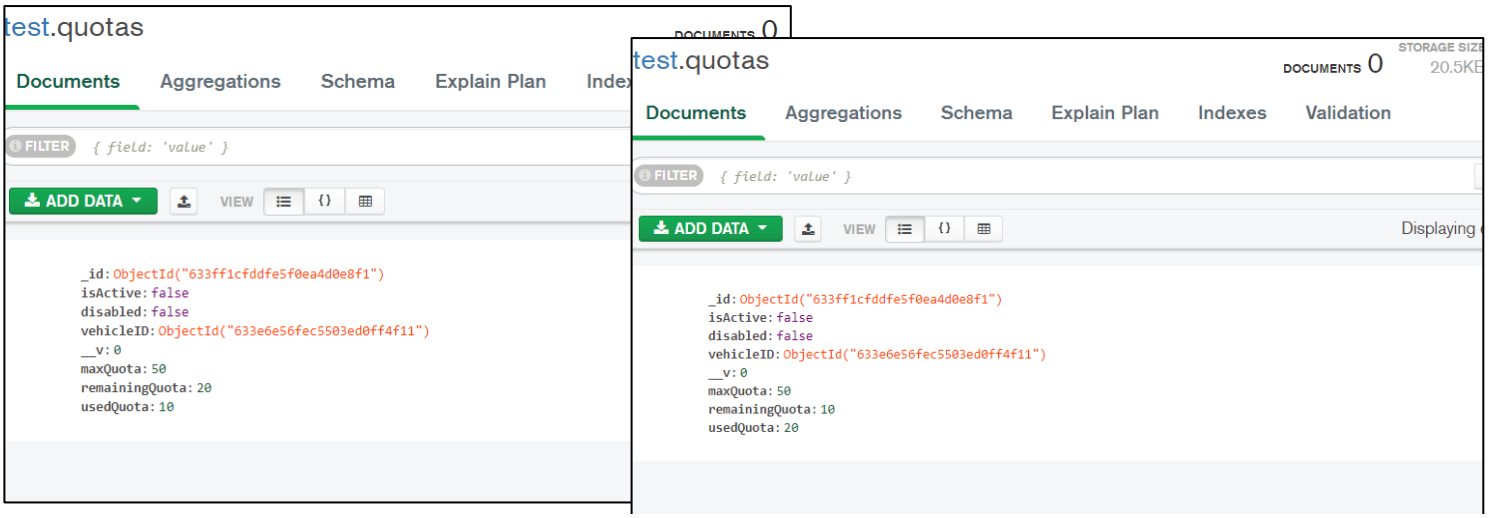


Figure 3.7: Quotas updated

3.1 Containerizing microservices

- A container is a standardized software component that wraps up code and all of its dependencies to ensure that an application will run swiftly and consistently in different computing environments.
- An application's code, runtime, system tools, system libraries, and settings are all included in a lightweight, independent, executable package known as a Docker container image.

```
PS C:\Users\DeLL\Downloads\microservices-demo-main\microservices-demo-main\auth>  docker build -t auths .
[+] Building 4.3s (9/9) FINISHED
 => [3/4] RUN npm install
 => [4/4] COPY . .
 => exporting to image
 => => exporting layers
 => => writing image sha256:88ef294a0460b5aff608720f9adf62c93171ffd994ca3176119e629654509916
 => => naming to docker.io/library/auths
PS C:\Users\DeLL\Downloads\microservices-demo-main\microservices-demo-main\auth> docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
auths        latest    88ef294a0460   5 seconds ago    77.8MB
quotas       latest    f4bce3e67775   27 seconds ago   77.8MB
vehciles     latest    09545c155e3d   3 minutes ago    77.8MB
```

Figure 3.8: Docker build

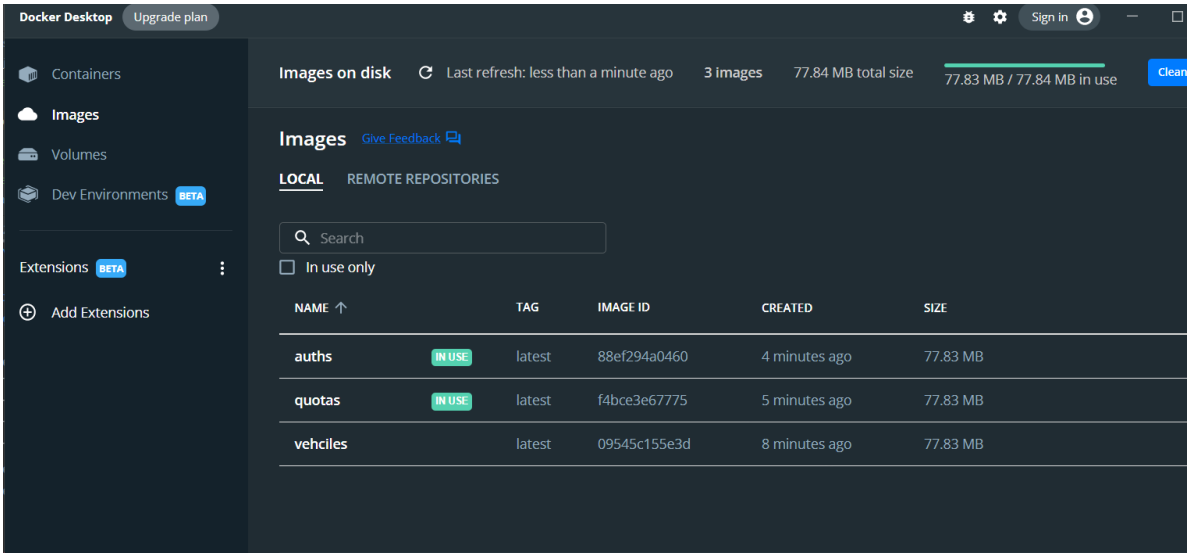


Figure 3.9 : Docker images

3.2 Docker on Amazon ECS (Create tasks definitions and add the images to AWS ECR)

- Push the image created to Amazon ECR, Elastic Container Registry, create an ECS cluster and download the image from ECR onto the ECS cluster.

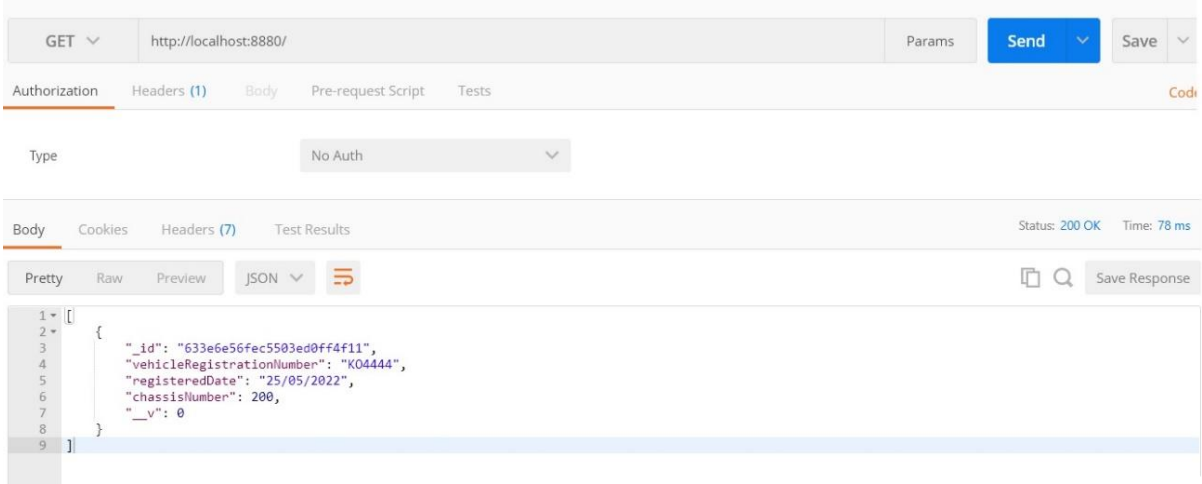


Figure 3.10: Docker Image running

- Before that create an IAM user and setup our AWS CLI

3.2.1 Configuring the AWS CLI

- ECS-User
- Access key –
- Secret Key -

3.2.2 Creating ECS cluster

- aws ecs create-cluster --cluster-name docker-on-aws
- Validate : aws ecs list-clusters
- To delete : aws ecs delete-cluster --cluster docker-on-aws

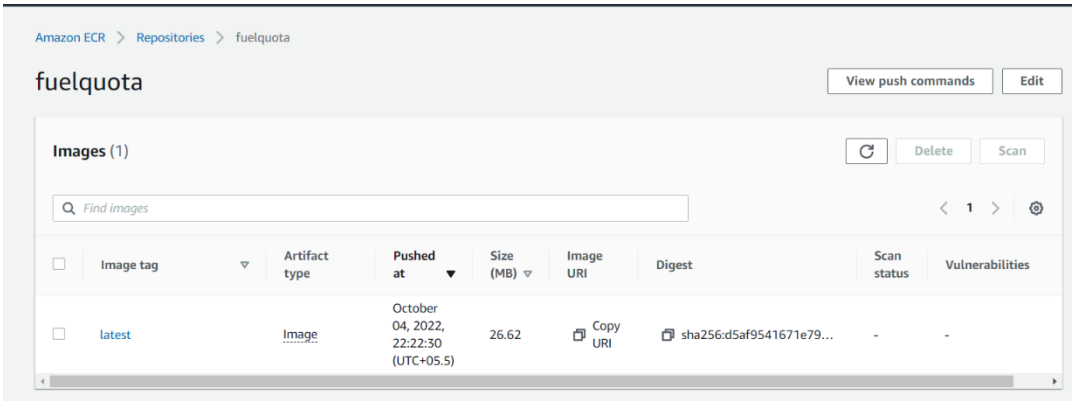
```
C:\Users\DeLL>aws configure
AWS Access Key ID [None]: AKIAX3W627NSCYQ4ZWUC
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]:

C:\Users\DeLL>aws ecs create-cluster --cluster-name docker-on-aws
{
    "cluster": {
        "clusterArn": "arn:aws:ecs:us-east-1:540559473508:cluster/docker-on-aws",
        "clusterName": "docker-on-aws",
        "status": "ACTIVE",
        "registeredContainerInstancesCount": 0,
        "runningTasksCount": 0,
        "pendingTasksCount": 0,
        "activeServicesCount": 0,
        "statistics": [],
        "tags": [],
        "settings": [
            {
                "name": "containerInsights",
                "value": "disabled"
            }
        ],
        "capacityProviders": [],
        "defaultCapacityProviderStrategy": []
    }
}


C:\Users\DeLL>aws ecs list-clusters
{
    "clusterArns": [
        "arn:aws:ecs:us-east-1:540559473508:cluster/docker-on-aws"
    ]
}
```

3.2.3 Pushing an Image to Amazon ECR

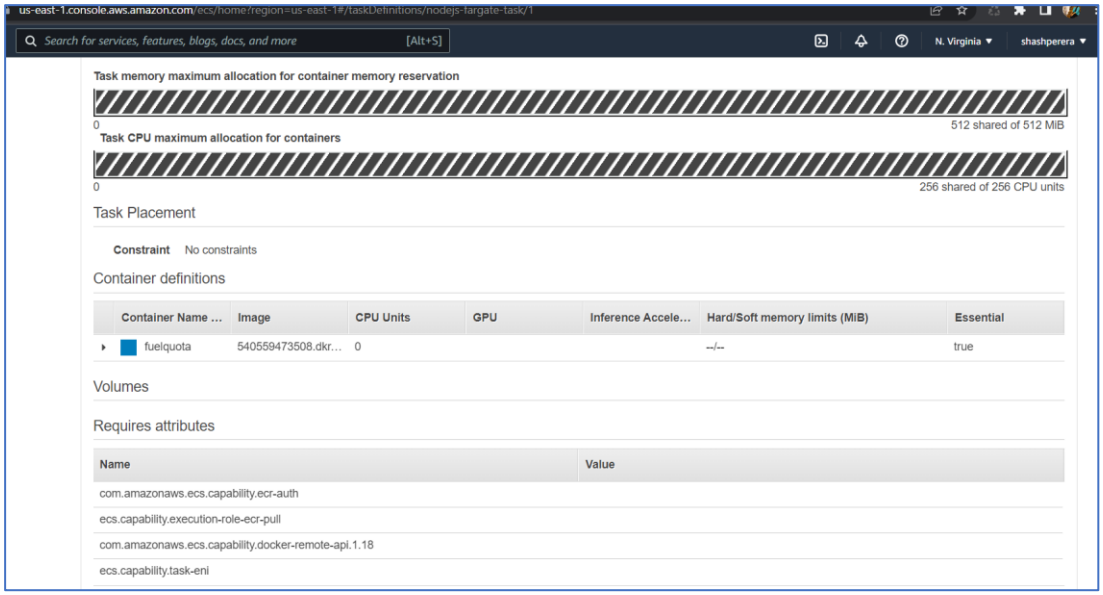
- Create repo - aws ecr create-repository --repository-name fuelquota
- Login - aws ecr get-login-password | docker login --username AWS --password-stdin 540559473508.dkr.ecr.us-east-1.amazonaws.com

```
C:\Users\DeLL>aws ecr create-repository --repository-name fuelquota
{
    "repository": {
        "repositoryArn": "arn:aws:ecr:us-east-1:540559473508:repository/fuelquota",
        "registryId": "540559473508",
        "repositoryName": "fuelquota",
        "repositoryUri": "540559473508.dkr.ecr.us-east-1.amazonaws.com/fuelquota",
        "createdAt": "2022-10-04T22:18:40+05:30",
        "imageTagMutability": "MUTABLE",
        "imageScanningConfiguration": {
            "scanOnPush": false
        },
        "encryptionConfiguration": {
            "encryptionType": "AES256"
        }
    }
}
```

- Auth image : docker tag auths:latest 540559473508.dkr.ecr.us-east-1.amazonaws.com/fuelquota
- docker push 540559473508.dkr.ecr.us-east-1.amazonaws.com/fuelquota
- pull : docker pull 540559473508.dkr.ecr.region.amazonaws.com/fuelquota
- Verify image : aws ecr list-images --repository-name fuelquota



3.2.4 Registering ECS Task Definition



3.2.5 Create ECS Service

- Create the security group

```
C:\Users\DeLL>aws ec2 create-security-group --group-name ecs-security-group --description "Security Group us-east-1 for ECS"
{
    "GroupId": "sg-041c27d9e92266e3d"
}


C:\Users\DeLL>aws ec2 describe-security-groups --group-id sg-041c27d9e92266e3d
{
    "SecurityGroups": [
        {
            "Description": "Security Group us-east-1 for ECS",
            "GroupName": "ecs-security-group",
            "IpPermissions": [],
            "OwnerId": "540559473508",
            "GroupId": "sg-041c27d9e92266e3d",
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "Ipv6Ranges": [],
                    "PrefixListIds": [],
                    "UserIdGroupPairs": []
                }
            ],
            "VpcId": "vpc-afbf8dd5"
        }
    ]
}
```

Sg - sg-041c27d9e92266e3d

- Add IP permissions

- Get a list of our public subnets (5827ff56, 6dba564c, 224aa97d, ab310495, cf7096a9 03a0114e)
    - us-east-1 : should have 6 subnets

```
C:\Users\DeLL>aws ec2 authorize-security-group-ingress --group-id sg-041c27d9e92266e3d --protocol tcp --port 80 --cidr 0.0.0.0/0
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-01fe461ffe25d35c2",
            "GroupId": "sg-041c27d9e92266e3d",
            "GroupOwnerId": "540559473508",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "CidrIpv4": "0.0.0.0/0"
        }
    ]
}


C:\Users\DeLL>aws ec2 describe-subnets
{
    "Subnets": [
        {
            "AvailabilityZone": "us-east-1f",
            "AvailabilityZoneId": "use1-az5",
            "AvailableIpAddressCount": 4091,
            "CidrBlock": "172.31.64.0/20",
            "DefaultForAz": true,
            "MapPublicIpOnLaunch": true,
            "MapCustomerOwnedIpOnLaunch": false,
            "State": "available",
            "SubnetId": "subnet-5827ff56",
```

Commands :

- aws ecs create-service --cluster docker-on-aws --service-name fuelquota-service --task-definition nodejs-fargate-task:1 --desired-count 1 --network-configuration "awsvpcConfiguration={subnets=[subnet-5827ff56, subnet-6dba564c, subnet-224aa97d , subnet-ab310495, subnet-cf7096a9 ,subnet-03a0114e],securityGroups=[sg-041c27d9e92266e3d],assignPublicIp=ENABLED}" --launch-type "FARGATE"

```
C:\Users\DeLL>aws ecs create-service --cluster docker-on-aws --service-name fuelquota-service --task-definition nodejs-fargate-task:1 --desired-count 1 --ne
twork-configuration "awsvpcConfiguration={subnets=[ subnet-5827ff56, subnet-6dba564c, subnet-224aa97d , subnet-ab310495, subnet-cf7096a9 ,subnet-03a0114e],s
ecurityGroups=[sg-041c27d9e92266e3d],assignPublicIp=ENABLED}" --launch-type "FARGATE"
{
    "service": {
        "serviceArn": "arn:aws:ecs:us-east-1:540559473508:service/docker-on-aws/fuelquota-service",
        "serviceName": "fuelquota-service",
        "clusterArn": "arn:aws:ecs:us-east-1:540559473508:cluster/docker-on-aws",
        "loadBalancers": [],
        "serviceRegistries": [],
        "status": "ACTIVE",
        "desiredCount": 1,
        "runningCount": 0,
        "pendingCount": 0,
        "launchType": "FARGATE",
        "platformVersion": "LATEST",
        "platformFamily": "Linux",
        "taskDefinition": "arn:aws:ecs:us-east-1:540559473508:task-definition/nodejs-fargate-task:1",
        "deploymentConfiguration": {
            "deploymentCircuitBreaker": {
                "enable": false,
                "rollback": false
            },
            "maximumPercent": 200,
            "minimumHealthyPercent": 100
        },
        "deployments": [
            {
                "id": "ecs-svc/6077805582529378494",
                "status": "PRIMARY",
                "taskDefinition": "arn:aws:ecs:us-east-1:540559473508:task-definition/nodejs-fargate-task:1",
                "desiredCount": 1,
                "pendingCount": 0,
                "runningCount": 0,
                "failedTasks": 0,
                "createdAt": "2022-10-04T22:59:10.087000+05:30",
                "updatedAt": "2022-10-04T22:59:10.087000+05:30",
                "launchType": "FARGATE",
```

```
                            "subnet-03a0114e"
                        ],
                        "securityGroups": [
                            "sg-041c27d9e92266e3d"
                        ],
                        "assignPublicIp": "ENABLED"
                    }
                },
                "rolloutState": "IN_PROGRESS",
                "rolloutStateReason": "ECS deployment ecs-svc/6077805582529378494 in progress."
            }
        ],
        "roleArn": "arn:aws:iam::540559473508:role/aws-service-role/ecs.amazonaws.com/AWSServiceRoleForECS",
        "events": [],
        "createdAt": "2022-10-04T22:59:10.087000+05:30",
        "placementConstraints": [],
        "placementStrategy": [],
        "networkConfiguration": {
            "awsvpcConfiguration": {
                "subnets": [
                    "subnet-5827ff56",
                    "subnet-6dba564c",
                    "subnet-224aa97d",
                    "subnet-ab310495",
                    "subnet-cf7096a9",
                    "subnet-03a0114e"
                ],
                "securityGroups": [
                    "sg-041c27d9e92266e3d"
                ],
                "assignPublicIp": "ENABLED"
            }
        },
        "schedulingStrategy": "REPLICA",
        "createdBy": "arn:aws:iam::540559473508:user/ECS-User",
        "enableECSManagedTags": false,
        "propagateTags": "NONE",
        "enableExecuteCommand": false
    }
}
```

Figure 3.11: Create Services

4 DynamoDB and Lambda

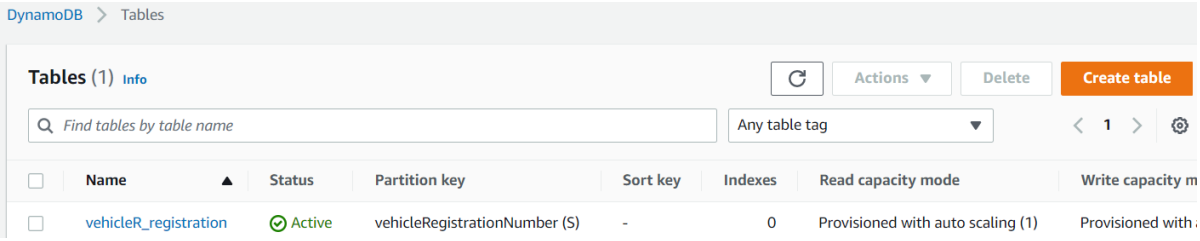


Figure 4.1: Table

1 Install aws-sdk NPM package

2 Store Access and Secret keys in config file

Notes to consider when using DynamoDB with Mongo (Refer appendix for dynamodb controller)

How to use the sdk and DocumentClient of sdk:

- Import and initialize aws-sdk in your JavaScript code
- Provide IAM User credentials (Access key ID and Secret access key) by calling config.update method of aws-sdk
- Initialize DynamoDB.DocumentClient of aws-sdk
- Call required methods of DocumentClient like (scan or put). DocumentClient methods require parameters like options and callbacks
- Options require properties like TableName and Item. Item is the record user wants to insert.

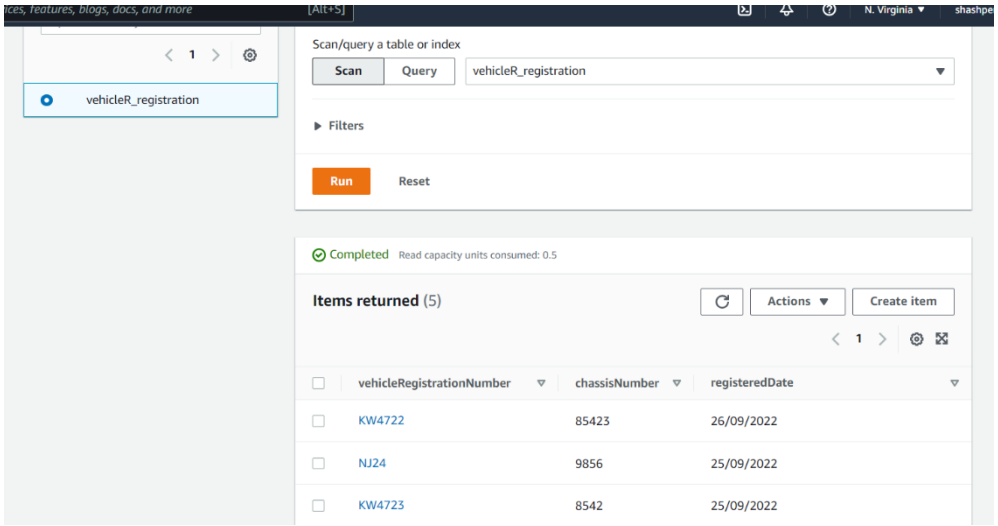
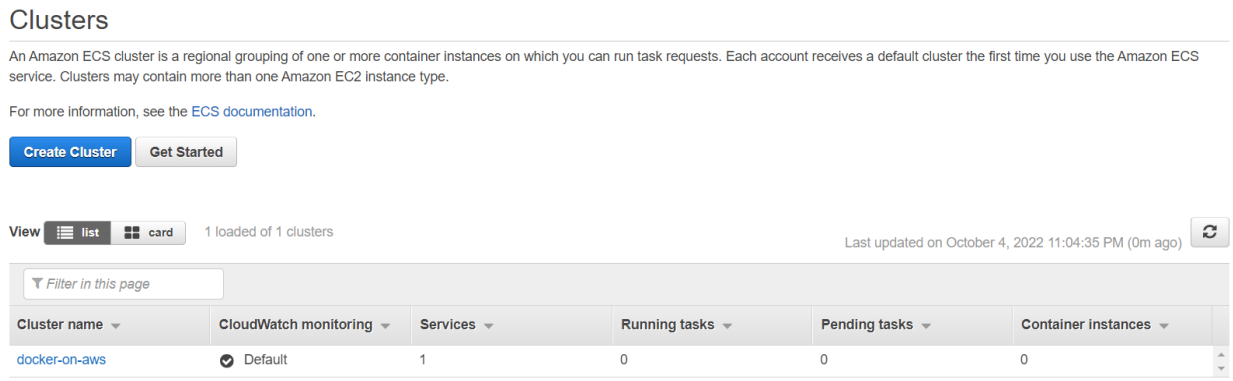
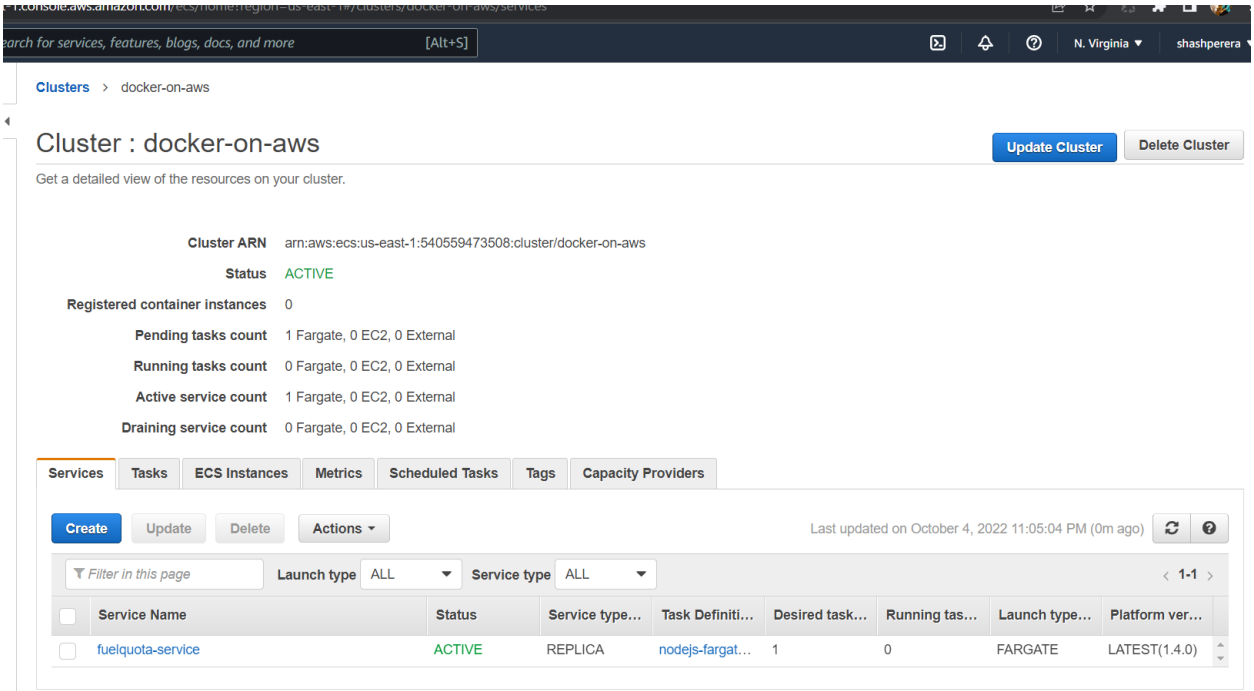


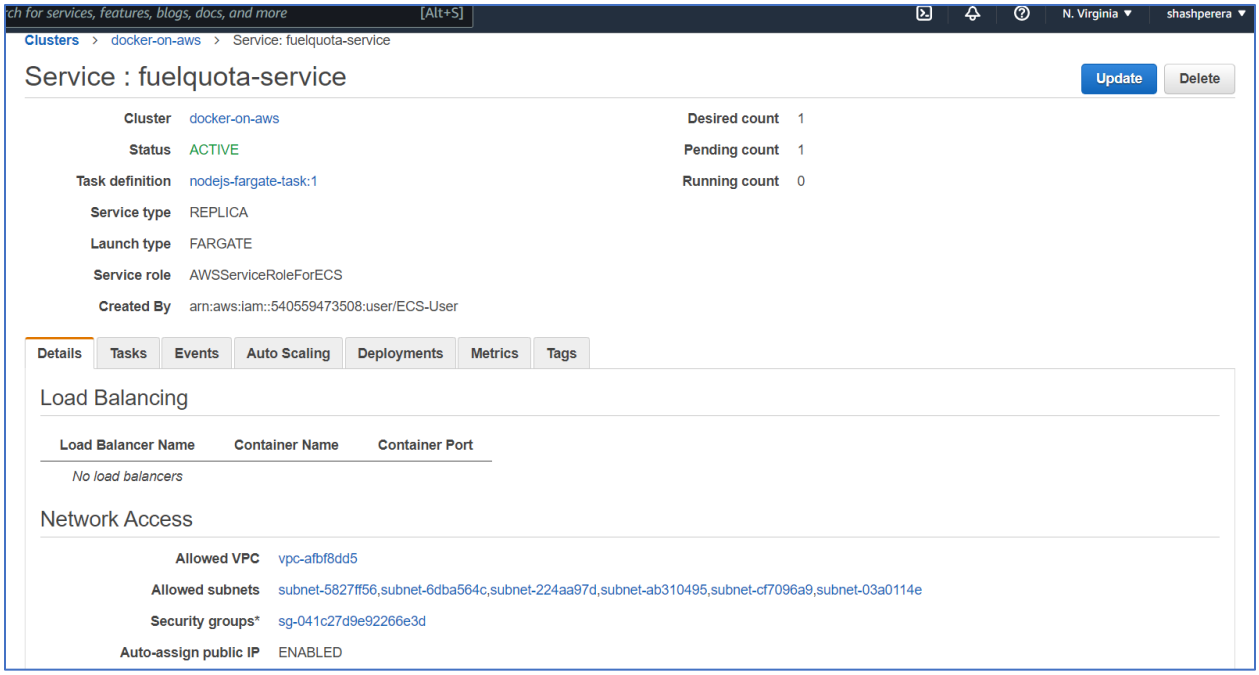
Figure 4.2: Table items added using lambda function

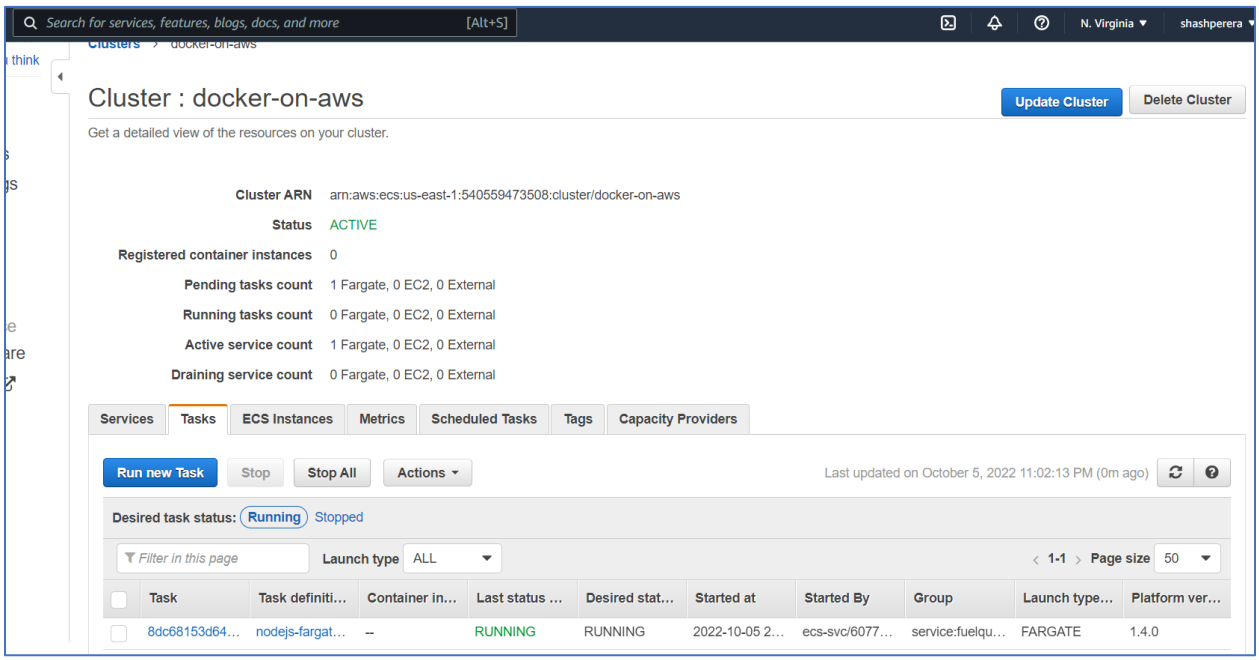
5 AWS Fargate

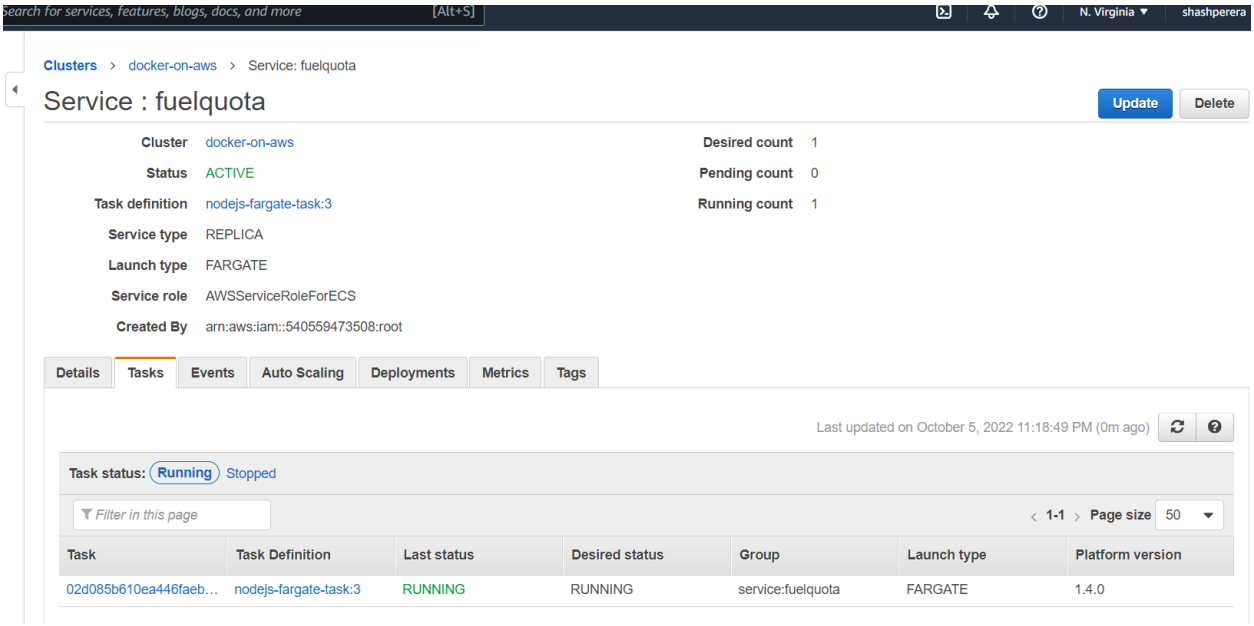
5.1 Configuring the services.

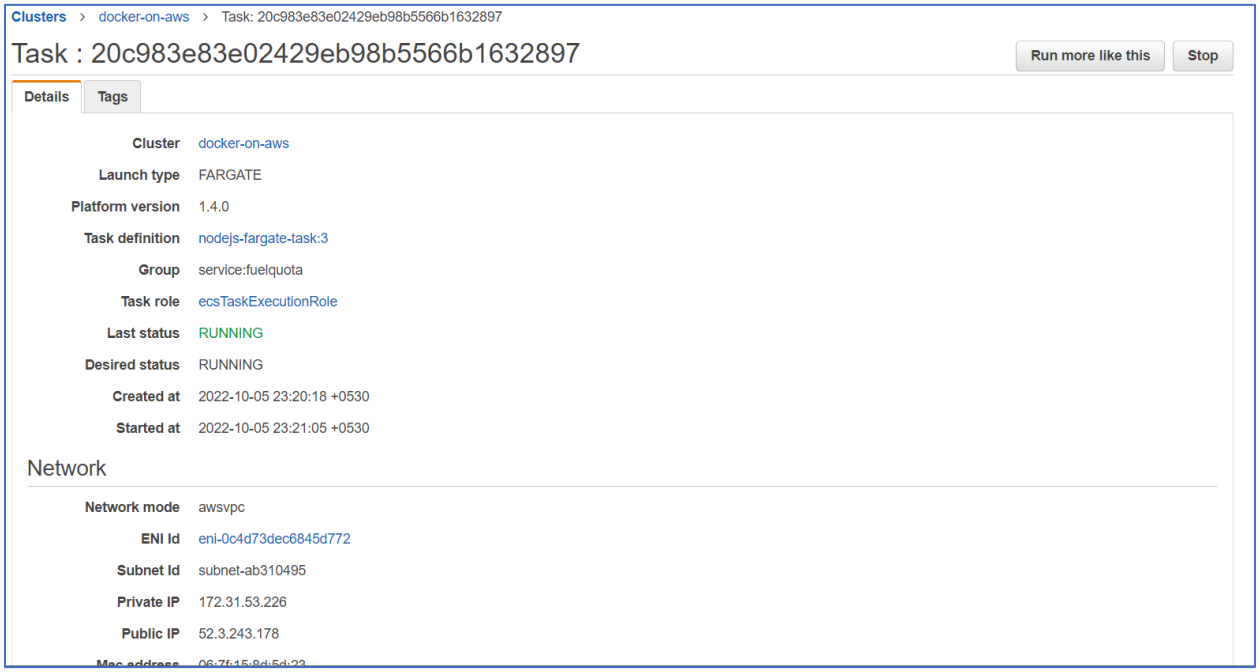





Clusters > docker-on-aws > Service: fuelquota-service
Service : fuelquota-service
Update
Delete
Cluster docker-on-aws
Status ACTIVE
Task definition nodejs-fargate-task:1
Service type REPLICA
Launch type FARGATE
Service role AWSServiceRoleForECS
Created By arn:aws:iam::540559473508:user/ECS-User
Desired count 1
Pending count 1
Running count 0
Details
Tasks
Events
Auto Scaling
Deployments
Metrics
Tags
Load Balancing
Load Balancer Name
Container Name
Container Port
No load balancers
Network Access
Allowed VPC vpc-afbf8dd5
Allowed subnets subnet-5827ff56,subnet-6dba564c,subnet-224aa97d,subnet-ab310495,subnet-cf7096a9,subnet-03a0114e
Security groups* sg-041c27d9e92266e3d
Auto-assign public IP ENABLED


Search for services, features, blogs, docs, and more [Alt+S]
N. Virginia
shashperera
Cluster : docker-on-aws
Update Cluster
Delete Cluster
Get a detailed view of the resources on your cluster.
Cluster ARN arn:aws:ecs:us-east-1:540559473508:cluster/docker-on-aws
Status ACTIVE
Registered container instances 0
Pending tasks count 1 Fargate, 0 EC2, 0 External
Running tasks count 0 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
Run new Task
Stop
Stop All
Actions
Last updated on October 5, 2022 11:02:13 PM (0m ago)
Desired task status: Running Stopped
Filter in this page
Launch type ALL
1-1
Page size 50
Task
Task definiti...
Container in...
Last status ...
Desired stat...
Started at
Started By
Group
Launch type...
Platform ver...
8dc68153d64...
nodejs-fargat...
--
RUNNING
RUNNING
2022-10-05 2...
ecs-svc/6077...
service:fuelqu...
FARGATE
1.4.0


Search for services, features, blogs, docs, and more
[Alt+S]
N. Virginia
shashperera
Clusters > docker-on-aws > Service: fuelquota
Service : fuelquota
Update
Delete
Cluster docker-on-aws
Status ACTIVE
Task definition nodejs-fargate-task:3
Service type REPLICA
Launch type FARGATE
Service role AWSServiceRoleForECS
Created By arn:aws:iam::540559473508:root
Desired count 1
Pending count 0
Running count 1
Details
Tasks
Events
Auto Scaling
Deployments
Metrics
Tags
Last updated on October 5, 2022 11:18:49 PM (0m ago)
Task status: Running Stopped
Filter in this page
1-1
Page size 50
Task
Task Definition
Last status
Desired status
Group
Launch type
Platform version
02d085b610ea446faeb…
nodejs-fargate-task:3
RUNNING
RUNNING
service:fuelquota
FARGATE
1.4.0


Clusters > docker-on-aws > Task: 20c983e83e02429eb98b5566b1632897
Task : 20c983e83e02429eb98b5566b1632897
Run more like this
Stop
Details
Tags
Cluster docker-on-aws
Launch type FARGATE
Platform version 1.4.0
Task definition nodejs-fargate-task:3
Group service:fuelquota
Task role ecsTaskExecutionRole
Last status RUNNING
Desired status RUNNING
Created at 2022-10-05 23:20:18 +0530
Started at 2022-10-05 23:21:05 +0530
Network
Network mode awsvpc
ENI Id eni-0c4d73dec6845d772
Subnet Id subnet-ab310495
Private IP 172.31.53.226
Public IP 52.3.243.178

Clusters > docker-on-aws > Task: 20c983e83e02429eb98b5566b1632897

Task : 20c983e83e02429eb98b5566b1632897

Run more like this | Stop

Details | Tags

Cluster	docker-on-aws
Launch type	FARGATE
Platform version	1.4.0
Task definition	nodejs-fargate-task:3
Group	service:fuelquota
Task role	ecsTaskExecutionRole
Last status	RUNNING
Desired status	RUNNING
Created at	2022-10-05 23:20:18 +0530
Started at	2022-10-05 23:21:05 +0530

Network

Network mode	awsvpc
ENI Id	eni-0c4d73dec6845d772
Subnet Id	subnet-ab310495
Private IP	172.31.53.226
Public IP	52.3.243.178
Mac address	06:7f:15:8d:5d:23

Clusters > docker-on-aws > Service: fuelquota

Service : fuelquota

Update | Delete

Cluster	docker-on-aws	Desired count	1
Status	ACTIVE	Pending count	1
Task definition	nodejs-fargate-task:3	Running count	0
Service type	REPLICA		
Launch type	FARGATE		
Service role	AWSServiceRoleForECS		
Created By	arn:aws:iam::540559473508:root		

Details | Tasks | Events | Auto Scaling | Deployments | Metrics | Tags

Task Placement

Strategy	No strategies
Constraint	No constraints

Service Deployment Options

Minimum healthy percent	100
Maximum percent	200
Deployment circuit breaker	Disabled

create pipeline | view pipelines

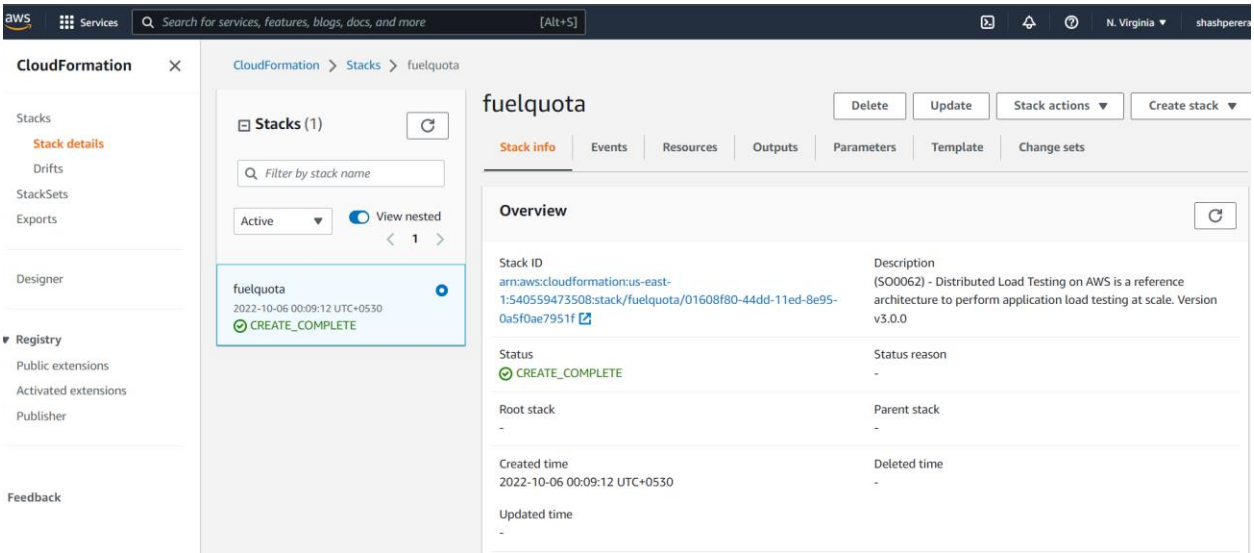
Last updated on October 5, 2022 11:23:09 PM (0m ago)

Filter in this page

1-1

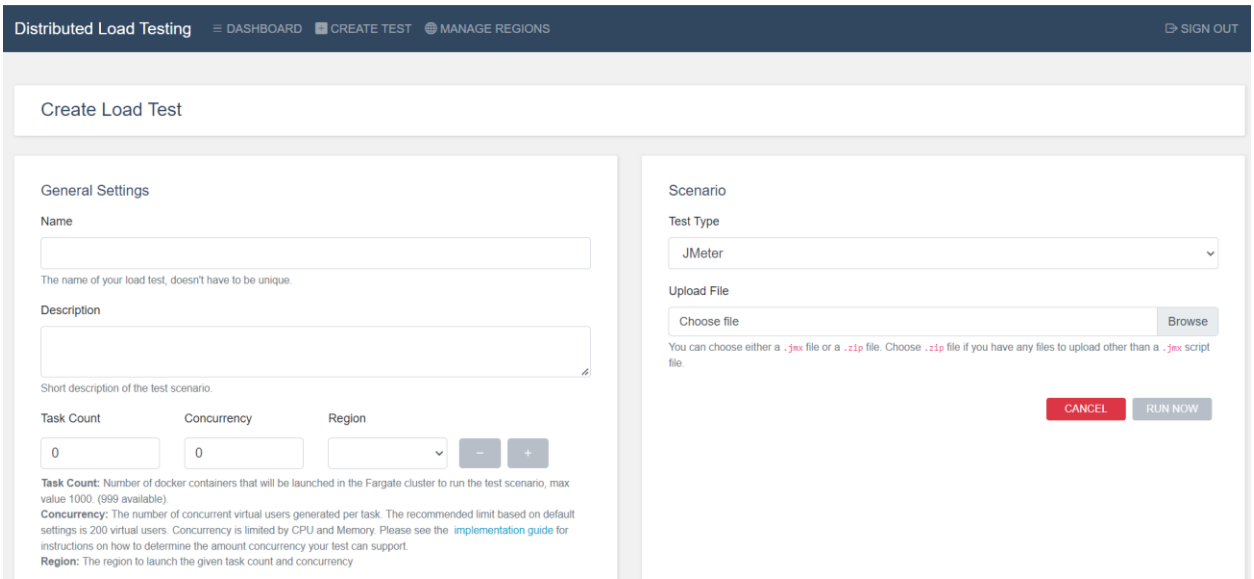
Deployment Id ...	Status	Desired count ...	Pending count ...	Running count...	Created time	Updated time ...	Rollout state	Rollout state re...
ecs-svc/142222...	PRIMARY	1	1	0	2022-10-05 23:1...	2022-10-05 23:2...	Completed	ECS deploymen...

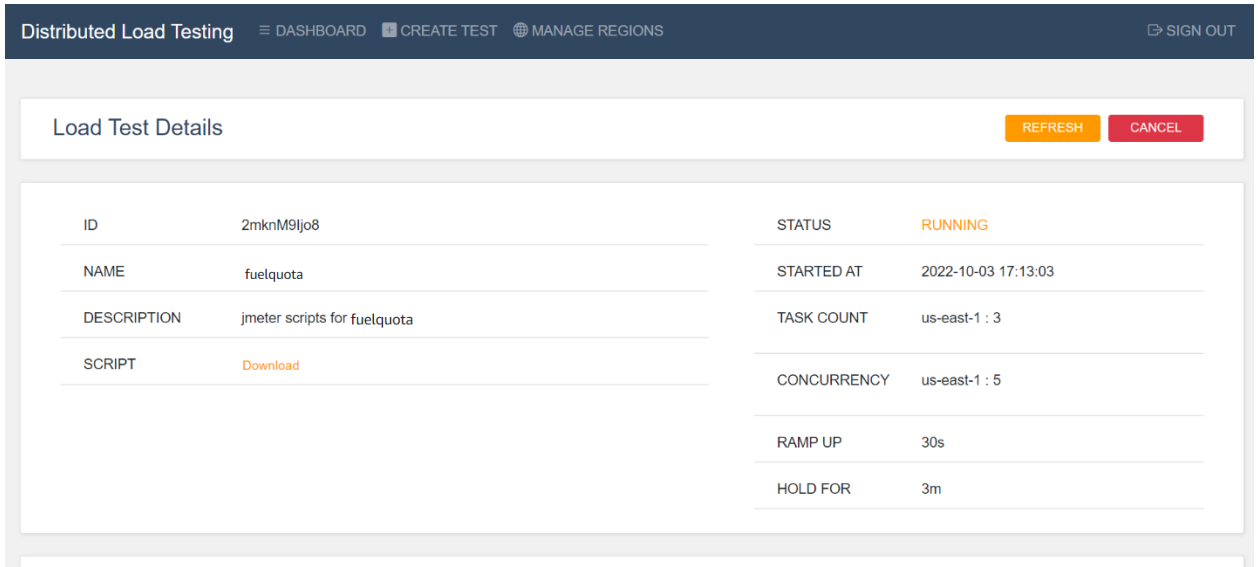
6 Cloudformation stack to deploy the entire infrastructure on the cloud

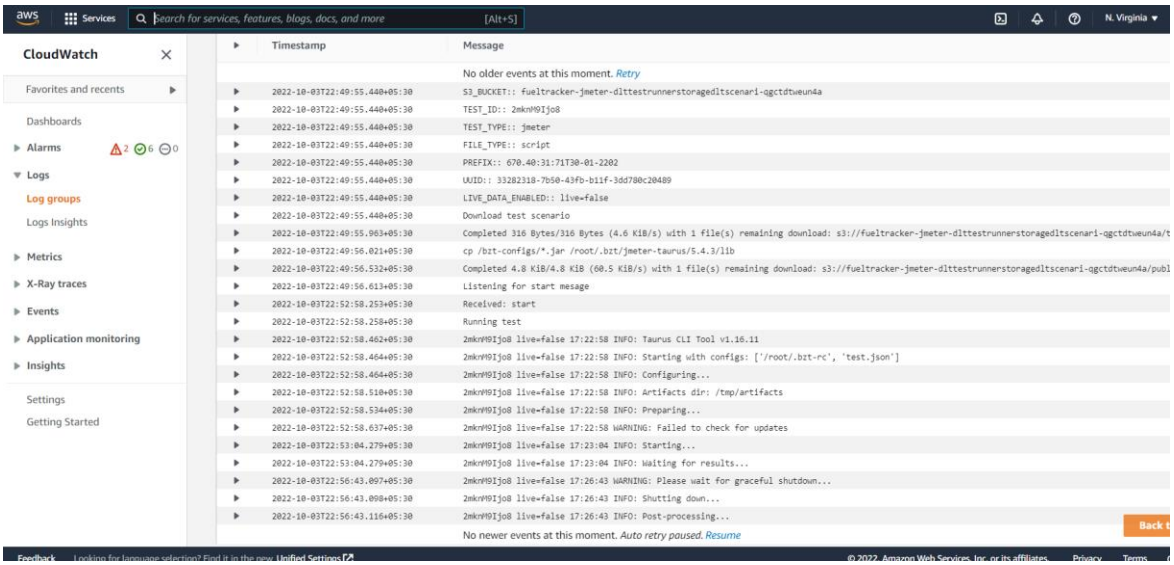


Find the scripts in the appendix or github repo.

7 JMeter load tests







Automation scripts such as Cloudformation and the JMeter test results. (Check appendix)

8 References

[1] “Building Lambda functions with Node.js - AWS Lambda,” *docs.aws.amazon.com*. [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/lambda-nodejs.html. [Accessed: Oct. 07, 2022]
[2] “Tutorial: Build a CRUD API with Lambda and DynamoDB - Amazon API Gateway,” *docs.aws.amazon.com*. [Online]. Available: https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-dynamo-db.html
[3] M. Whittle, “Upload to AWS S3 Using a Node.js Script or AWS Lambda,” *The Startup*, Jul. 25, 2021. [Online]. Available: https://medium.com/swlh/upload-to-aws-s3-using-a-node-js-script-or-aws-lambda-e1877960bcea. [Accessed: Oct. 07, 2022]
[4] “Bucket policy examples - Amazon Simple Storage Service,” *docs.aws.amazon.com*. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/userguide/example-bucket-policies.html
[5] “Implementation Guide” [Online]. Available: https://docs.aws.amazon.com/solutions/latest/distributed-load-testing-on-aws/distributed-load-testing-on-aws.pdf
[6] “Build a REST API with Node.js, Express, and MySQL,” *LogRocket Blog*, Jan. 18, 2022. [Online]. Available: https://blog.logrocket.com/build-rest-api-node-express-mysql/

9 Appendix

9.1 Fargate yaml

```
Description: (SO0062) - Distributed Load Testing on AWS is a reference
architecture to perform application load testing at scale. Version v3.0.0
AWSTemplateFormatVersion: "2010-09-09"
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      - Label:
          default: Console access
        Parameters:
          - AdminName
          - AdminEmail
      - Label:
          default: Enter values here to use your own existing VPC
        Parameters:
          - ExistingVPCId
          - ExistingSubnetA
          - ExistingSubnetB
      - Label:
          default: Or have the solution create a new AWS Fargate VPC
```

```
        Parameters:
          - VpcCidrBlock
          - SubnetACidrBlock
          - SubnetBCidrBlock
          - EgressCidr
    ParameterLabels:
      AdminName:
        default: "* Console Administrator Name"
      AdminEmail:
        default: "* Console Administrator Email"
      ExistingVPCId:
        default: "The ID of an existing VPC in this region. Ex: `vpc-
1a2b3c4d5e6f`"
      ExistingSubnetA:
        default: "The ID of a subnet within the existing VPC. Ex: `subnet-
7h8i9j0k`"
      ExistingSubnetB:
        default: "The ID of a subnet within the existing VPC. Ex: `subnet-
1x2y3z`"
      VpcCidrBlock:
        default: AWS Fargate VPC CIDR Block
      SubnetACidrBlock:
        default: AWS Fargate Subnet A CIDR Block
      SubnetBCidrBlock:
        default: AWS Fargate Subnet A CIDR Block
      EgressCidr:
        default: AWS Fargate SecurityGroup CIDR Block
Parameters:
  AdminName:
    Type: String
    AllowedPattern: "[a-zA-Z0-9-]+"
    ConstraintDescription: Admin username must be a minimum of 4 characters
and cannot include spaces
    Description: Admin user name to access the Distributed Load Testing
console
    MaxLength: 20
    MinLength: 4
  AdminEmail:
    Type: String
    AllowedPattern: ^[_A-Za-z0-9-\+]+(\.[_A-Za-z0-9-]+)*@[A-Za-z0-9-]+(\.[A-
Za-z0-9]+)*(\.[A-Za-z]{2,})$
              - Config
    Type: AWS::S3::BucketPolicy
    Properties:
      Bucket:
        Ref: DLTCommonResourcesLogsBucket48A2774D
      PolicyDocument:
        Statement:
          - Action: s3:*
            Condition:
              Bool:
                aws:SecureTransport: "false"
            Effect: Deny
            Principal:
              AWS: "*"
            Resource:
              - Fn::GetAtt:
```

```
                - DLTCommonResourcesLogsBucket48A2774D
                - Arn
            - Fn::Join:
                - ""
                - - Fn::GetAtt:
                      - DLTCommonResourcesLogsBucket48A2774D
                      - Arn
                  - /*
      Version: "2012-10-17"

  Type: AWS::DynamoDB::Table
  Properties:
    KeySchema:
      - AttributeName: testId
        KeyType: HASH
      - AttributeName: testRunId
        KeyType: RANGE
    AttributeDefinitions:
      - AttributeName: testId
        AttributeType: S
      - AttributeName: testRunId
        AttributeType: S
    BillingMode: PAY_PER_REQUEST
    PointInTimeRecoverySpecification:
      PointInTimeRecoveryEnabled: true
    SSESpecification:
      SSEEnabled: true
    Tags:
      - Key: SolutionId
        Value:
          Fn::FindInMap:
            - Solution
            - Config
            - SolutionId
  UpdateReplacePolicy: Retain
  DeletionPolicy: Retain
DLTTestRunnerStorageHistoryDynamoDbPolicyA439CB46:
  Type: AWS::IAM::Policy
  Properties:
    PolicyDocument:
      Statement:

    existingVPC:
      Fn::If:
        - BoolExistingVPC
        - true
        - false
    Region:
      Ref: AWS::Region
    Resource: AnonymousMetric
    SolutionId:
      Fn::FindInMap:
        - Solution
        - Config
        - SolutionId
    UUID:
      Fn::GetAtt:
```

```
          - DLTCustomResourcesCustomResourceUuidD1C03F15
          - UUID
      VERSION:
        Fn::FindInMap:
          - Solution
          - Config
          - CodeVersion

    Type: AWS::Logs::SubscriptionFilter
    Properties:
      DestinationArn:
        Fn::GetAtt:
          - RealTimeDataRealTimeDataPublisher7E8F8F6C
          - Arn
      FilterPattern: '"INFO: Current:" "live=true"'
      LogGroupName:
        Ref: DLTEcsDLTCloudWatchLogsGroupFE9EC144

  DLTLambdaFunctionDLTTestLambdaTaskRole1FDBCEDD:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service: lambda.amazonaws.com
        Version: "2012-10-17"
      Policies:
        - PolicyDocument:
            Statement:
              - Action: ecs:ListTasks
                Effect: Allow
                Resource: "*"
              - Action:
                  - ecs:RunTask
                  - ecs:DescribeTasks
                Effect: Allow
                Resource:
                  - Fn::Join:
                      - ""
                                  - "-"
            - - Fn::FindInMap:
                  - Solution
                  - Config
                  - S3Bucket
              - Ref: AWS::Region
        S3Key:
          Fn::Join:
            - ""
            - - Fn::FindInMap:
                  - Solution
                  - Config
                  - KeyPrefix
              - /task-runner.zip
      Role:
        Fn::GetAtt:
```

```
          - DLTLambdaFunctionDLTTestLambdaTaskRole1FDBCEDD
          - Arn
                  - Solution
                  - Config
                  - S3Bucket
              - Ref: AWS::Region
        S3Key:
          Fn::Join:
            - ""
            - - Fn::FindInMap:
                  - Solution
                  - Config
                  - KeyPrefix
              - /task-canceler.zip
      Role:
        Fn::GetAtt:
          - DLTLambdaFunctionLambdaTaskCancelerRoleAE2C84CF
          - Arn
      Description: Stops ECS task
      Environment:
        Variables:
          METRIC_URL:
            Fn::FindInMap:
              - Solution
              - Config
              - URL
          SOLUTION_ID:
            Fn::FindInMap:
              - Solution
              - Config
              - SolutionId
          VERSION:
            Fn::FindInMap:
              - Solution
              - Config
              - CodeVersion
          SCENARIOS_TABLE:
            Ref: DLTTestRunnerStorageDLTScenariosTableAB6F5C2A
      Handler: index.handler
      Runtime: nodejs14.x
      Tags:
        - Key: SolutionId
          Value:
            Fn::FindInMap:
              - Solution
              - Config
              - SolutionId
      Timeout: 300
    DependsOn:
      - DLTLambdaFunctionLambdaTaskCancelerRoleAE2C84CF
    Metadata:
      cfn_nag:
        rules_to_suppress:
          - id: W58
            reason: CloudWatchLogsPolicy covers a permission to write
CloudWatch logs.
          - id: W89
```

```
            reason: This Lambda function does not require a VPC
          - id: W92
            reason: Does not run concurrent executions
  DLTLambdaFunctionTaskCancelerInvokePolicyA1C7562A:
    Type: AWS::IAM::Policy
    Properties:
      PolicyDocument:
        Statement:
          - Action: lambda:InvokeFunction
            Effect: Allow
            Resource:
              Fn::GetAtt:
                - DLTLambdaFunctionTaskCanceler4E12BDA6
                - Arn
        Version: "2012-10-17"
      PolicyName: DLTLambdaFunctionTaskCancelerInvokePolicyA1C7562A
      Roles:
        - Ref: DLTLambdaFunctionTaskStatusRole9288E645
        - Ref: DLTApiDLTAPIServicesLambdaRole4465EAA4
      Metadata:
      cfn_nag:
        rules_to_suppress:
          - id: W12
            reason: CloudWatch logs actions do not support resource level
permissions
          - id: W76
            reason: The IAM policy is written for least-privilege access.
  DLTStepFunctionTaskRunnerStepFunctionsC295A535:
    Type: AWS::StepFunctions::StateMachine
    Properties:
      RoleArn:
        Fn::GetAtt:
          - DLTStepFunctionTaskRunnerStepFunctionsRoleC2237F06
          - Arn
      DefinitionString:
        Fn::Join:
          - ""
          - - '{"StartAt":"Regions for testing","States":{"Regions for
testing":{"Type":"Map","ResultPath":null,"Next":"Parse
result","InputPath":"$","Parameters":{"testTaskConfig.$":"$$.Map.Item.Value",
"testId.$":"$.testId","testType.$":"$.testType","fileType.$":"$.fileType","sh
owLive.$":"$.showLive","prefix.$":"$.prefix"},"Iterator":{"StartAt":"Check
running tests","States":{"Check running tests":{"Next":"No running
tests","Retry":[{"ErrorEquals":["Lambda.ServiceException","Lambda.AWSLambdaEx
ception","Lambda.SdkClientException"],"IntervalSeconds":2,"MaxAttempts":6,"Ba
ckoffRate":2}],"Type":"Task","InputPath":"$","OutputPath":"$.Payload","Resour
ce":"arn:'
            - Ref: AWS::Partition
            - :states:::lambda:invoke","Parameters":{"FunctionName":"
            - Fn::GetAtt:
                - DLTLambdaFunctionTaskStatusChecker1AA63EC9
                - Arn
            - '","Payload.$":"$"}},"No running
tests":{"Type":"Choice","Choices":[{"Variable":"$.isRunning","BooleanEquals":
false,"Next":"Run workers"}],"Default":"Test is still running"},"Test is
still running":{"Type":"Fail","Error":"TestAlreadyRunning","Cause":"The same
test is already running."},"Run workers":{"Next":"Requires
```

```
leader?","Retry":[{"ErrorEquals":["Lambda.ServiceException","Lambda.AWSLambda
Exception","Lambda.SdkClientException"],"IntervalSeconds":2,"MaxAttempts":6,"
BackoffRate":2}],"Type":"Task","InputPath":"$","OutputPath":"$.Payload","Reso
urce":"arn:'
   Console:
    Description: Console URL
    Value:
      Fn::GetAtt:
        - DLTConsoleResourcesDLTCloudFrontToS3CloudFrontDistribution3EF384B4
        - DomainName
  SolutionUUID:
    Description: Solution UUID
    Value:
      Fn::GetAtt:
        - DLTCustomResourcesCustomResourceUuidD1C03F15
        - UUID
  RegionalCFTemplate:
    Description: S3 URL for regional CloudFormation template
    Value:
      Fn::Join:
        - ""
        - - https://s3.
          - Ref: AWS::Region
          - "."
          - Ref: AWS::URLSuffix
          - /
          - Ref: DLTTestRunnerStorageDLTScenariosBucketA9290D21
          - /regional-template/distributed-load-testing-on-aws-
regional.template
    Export:
      Name: RegionalCFTemplate
```

9.2 Lambda function

```
import boto3
s3_boto3 = boto3.client("s3")
dynamodb= boto3.resource('dynamodb')
table=dynamodb.Table("vehicleR_registration")

def lambda_handler(event, context):
    bucket_name = event['Records'][0]['s3']['bucket']['name']
    s3_file_name= event['Records'][0]['s3']['object']['key']
    resp = s3_boto3.get_object(Bucket=bucket_name,Key=s3_file_name)
    data=resp['Body'].read().decode('utf-8')
    vehicles=data.split("\n")
    for veh in vehicles:
        veh=veh.split(',')
        table.put_item(
            Item={
                # "id":context.get_remaining_time_in_millis(),
                "vehicleRegistrationNumber":veh[0],
                "registeredDate":veh[1],
                "chassisNumber":veh[2]
```

```
            }
            )
        print(veh)
```

9.3 Task Definition

```
{
  "family": "nodejs-fargate-task",
  "networkMode": "awsvpc",
  "executionRoleArn": "arn:aws:iam::540559473508:role/ecsTaskExecutionRole",
  "containerDefinitions": [
    {
      "name": "nodejs-app",
      "image": "540559473508.dkr.ecr.us-east-
1.amazonaws.com/fuelquota:latest",
      "portMappings": [
        {
          "containerPort": 80,
          "hostPort": 80,
          "protocol": "tcp"
        }
      ],
      "essential": true
    }
  ],
  "requiresCompatibilities": [
    "FARGATE"
  ],
  "cpu": "256",
  "memory": "512"
}
```

9.4 Task execution role

```
"Version" : "2022-10-04",
"Statement": [
{
"Sid": "",
"Effect": "Allow",
"Principal": {
"Service": "ecs-tasks.amazonaws.com"
},
"Action": "sts:AssumeRole"
}
]
}
```

9.5 DynamoDB Controller

```
const AWS = require('aws-sdk');
const config = require('./../../config.js');
```

```
const uuidv1 = require('uuid/v1');

const getVehicles = function (req, res) {
    AWS.config.update(config.aws_remote_config);

    const docClient = new AWS.DynamoDB.DocumentClient();

    const params = {
        TableName: config.aws_table_name
    };

    docClient.scan(params, function (err, data) {

        if (err) {
            console.log(err)
            res.send({
                success: false,
                message: err
            });
        } else {
            const { Items } = data;
            res.send({
                success: true,
                vehicles: Items
            });
        }
    });
}

const vregister = function (req, res) {
    AWS.config.update(config.aws_remote_config);
    const docClient = new AWS.DynamoDB.DocumentClient();
    const Item = { ...req.body };
    Item.id = uuidv1();
    var params = {
        TableName: config.aws_table_name,
        Item: Item
    };

    // Call DynamoDB to add the item to the table
    docClient.put(params, function (err, data) {
        if (err) {
            res.send({
                success: false,
                message: err
            });
        } else {
            res.send({
                success: true,
                message: 'Added vehicle',
                vehicle: data
            });
        }
    });
}

module.exports = {
```

```
    getVehicles,
    vregister
}
```

9.6 Get vehicle Test script Jmeter

```
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.5">
  <hashTree>
    <TestPlan guiclass="TestPlanGui" testclass="TestPlan"
testname="fuelquota" enabled="true">
      <stringProp name="TestPlan.comments"></stringProp>
      <boolProp name="TestPlan.functional_mode">false</boolProp>
      <boolProp name="TestPlan.tearDown_on_shutdown">true</boolProp>
      <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
      <elementProp name="TestPlan.user_defined_variables"
elementType="Arguments" guiclass="ArgumentsPanel" testclass="Arguments"
testname="User Defined Variables" enabled="true">
        <collectionProp name="Arguments.arguments"/>
      </elementProp>
      <stringProp name="TestPlan.user_define_classpath"></stringProp>
    </TestPlan>
    <hashTree>
      <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
testname="fuelquota grp1" enabled="true">
        <stringProp name="ThreadGroup.on_sample_error">continue</stringProp>
        <elementProp name="ThreadGroup.main_controller"
elementType="LoopController" guiclass="LoopControlPanel"
testclass="LoopController" testname="Loop Controller" enabled="true">
          <boolProp name="LoopController.continue_forever">false</boolProp>
          <stringProp name="LoopController.loops">1</stringProp>
        </elementProp>
        <stringProp name="ThreadGroup.num_threads">1</stringProp>
        <stringProp name="ThreadGroup.ramp_time">1</stringProp>
        <boolProp name="ThreadGroup.scheduler">false</boolProp>
        <stringProp name="ThreadGroup.duration"></stringProp>
        <stringProp name="ThreadGroup.delay"></stringProp>
        <boolProp
name="ThreadGroup.same_user_on_next_iteration">true</boolProp>
      </ThreadGroup>
      <hashTree>
        <HTTPSamplerProxy guiclass="HttpTestSampleGui"
testclass="HTTPSamplerProxy" testname="HTTP Request" enabled="true">
          <elementProp name="HTTPsampler.Arguments" elementType="Arguments"
guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User Defined
Variables" enabled="true">
            <collectionProp name="Arguments.arguments"/>
          </elementProp>
          <stringProp
name="HTTPSampler.domain">54.89.226.130:3000/vehicles</stringProp>
          <stringProp name="HTTPSampler.port">></stringProp>
          <stringProp name="HTTPSampler.protocol">></stringProp>
          <stringProp name="HTTPSampler.contentEncoding">></stringProp>
          <stringProp name="HTTPSampler.path">/</stringProp>
          <stringProp name="HTTPSampler.method">GET</stringProp>
```

```
          <boolProp name="HTTPSampler.follow_redirects">true</boolProp>
          <boolProp name="HTTPSampler.auto_redirects">false</boolProp>
          <boolProp name="HTTPSampler.use_keepalive">true</boolProp>
          <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
          <stringProp name="HTTPSampler.embedded_url_re"></stringProp>
          <stringProp name="HTTPSampler.connect_timeout"></stringProp>
          <stringProp name="HTTPSampler.response_timeout"></stringProp>
        </HTTPSamplerProxy>
        <hashTree>
          <ResultCollector guiclass="TableVisualizer"
testclass="ResultCollector" testname="View Results in Table" enabled="true">
            <boolProp name="ResultCollector.error_logging">false</boolProp>
            <objProp>
              <name>saveConfig</name>
              <value class="SampleSaveConfiguration">
                <time>true</time>
                <latency>true</latency>
                <timestamp>true</timestamp>
                <success>true</success>
                <label>true</label>
                <code>true</code>
                <message>true</message>
                <threadName>true</threadName>
                <dataType>true</dataType>
                <encoding>false</encoding>
                <assertions>true</assertions>
                <subresults>true</subresults>
                <responseData>false</responseData>
                <samplerData>false</samplerData>
                <xml>false</xml>
                <fieldNames>true</fieldNames>
                <responseHeaders>false</responseHeaders>
                <requestHeaders>false</requestHeaders>
                <responseDataOnError>false</responseDataOnError>

<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMessage>
                <assertionsResultsToSave>0</assertionsResultsToSave>
                <bytes>true</bytes>
                <sentBytes>true</sentBytes>
                <url>true</url>
                <threadCounts>true</threadCounts>
                <idleTime>true</idleTime>
                <connectTime>true</connectTime>
              </value>
            </objProp>
            <stringProp name="filename">></stringProp>
          </ResultCollector>
          <hashTree/>
        </hashTree>
      </hashTree>
    </hashTree>
  </hashTree>
</jmeterTestPlan>
```