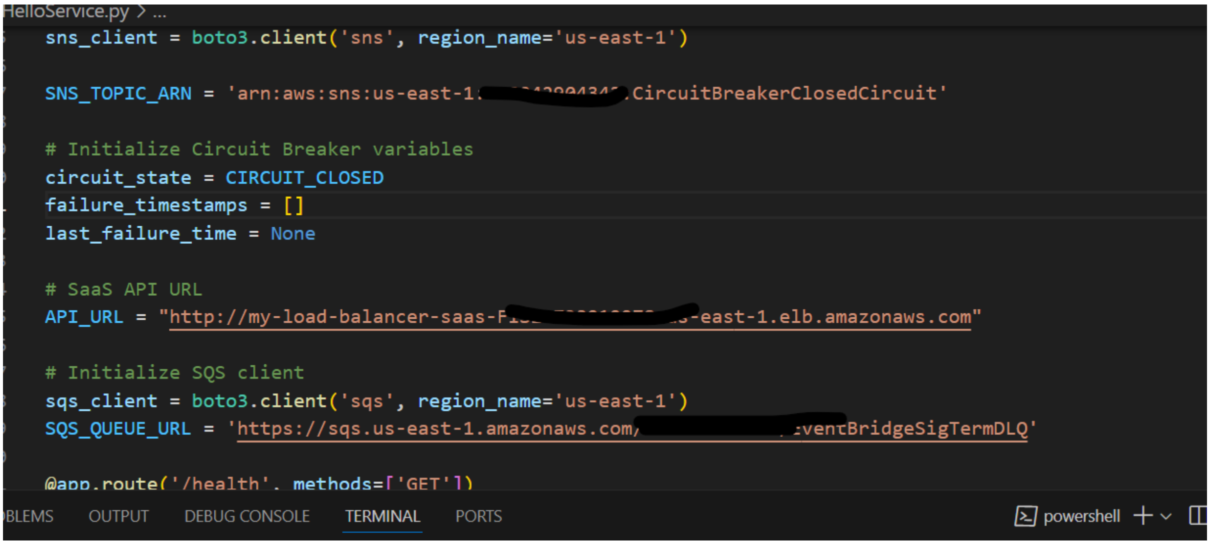
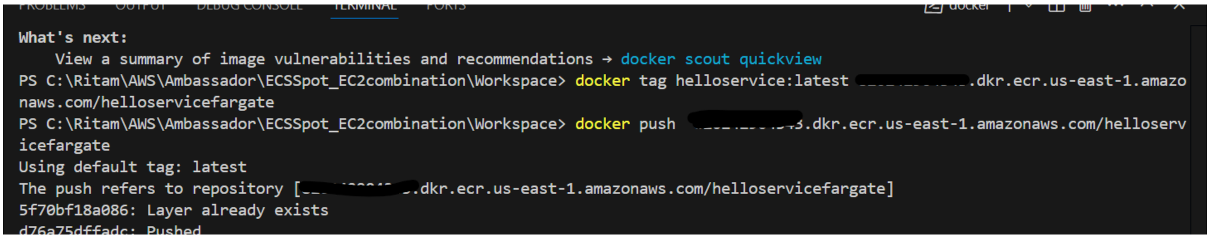
1. Main application build. Configure the SQS, SNS and ELB endpoint. This can be put in environment variables however here for POC the endpoints are hardcoded.



```
HelloService.py > ...
    sns_client = boto3.client('sns', region_name='us-east-1')

    SNS_TOPIC_ARN = 'arn:aws:sns:us-east-1:[redacted]:CircuitBreakerClosedCircuit'

    # Initialize Circuit Breaker variables
    circuit_state = CIRCUIT_CLOSED
    failure_timestamps = []
    last_failure_time = None

    # SaaS API URL
    API_URL = "http://my-load-balancer-saas-F[redacted]-east-1.elb.amazonaws.com"

    # Initialize SQS client
    sqs_client = boto3.client('sqs', region_name='us-east-1')
    SQS_QUEUE_URL = 'https://sqs.us-east-1.amazonaws.com/[redacted]EventBridgeSigTermDLQ'

    @app.route('/health', methods=['GET'])
```

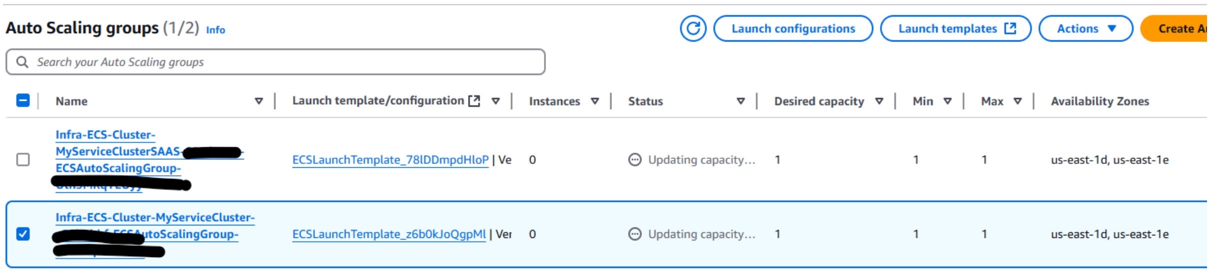
2. Now the endpoint will be build using the following commands and will insert the image in ECR.

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin <account>.dkr.ecr.us-east-1.amazonaws.com/helloservicefargate

docker build --load -t helloservice .

docker tag helloservice:latest <account>..dkr.ecr.us-east-1.amazonaws.com/helloservicefargate
docker push  <account>..dkr.ecr.us-east-1.amazonaws.com/helloservicefargate
```



```
What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Ritam\AWS\Ambassador\ECSSpot_EC2combination\Workspace> docker tag helloservice:latest [redacted].dkr.ecr.us-east-1.amazo
naws.com/helloservicefargate
PS C:\Ritam\AWS\Ambassador\ECSSpot_EC2combination\Workspace> docker push [redacted].dkr.ecr.us-east-1.amazonaws.com/helloserv
icefargate
Using default tag: latest
The push refers to repository [[redacted].dkr.ecr.us-east-1.amazonaws.com/helloservicefargate]
5f70bf18a086: Layer already exists
d76a75dffadc: Pushed
```

3. ASG is already created with launch template where the EC2 count is marked as 1 which will give EC2 infrastructure to ECS where the main application is running.



Auto Scaling groups (1/2) Info

Search your Auto Scaling groups

	Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones
☐	Infra-ECS-Cluster-MyServiceClusterSAAS-[redacted] ECSAutoScalingGroup-[redacted]	ECSLaunchTemplate_78lDDmpdHloP | Ve	0	Updating capacity…	1	1	1	us-east-1d, us-east-1e
☑	Infra-ECS-Cluster-MyServiceCluster-[redacted]utoScalingGroup-[redacted]	ECSLaunchTemplate_z6b0kJoQgpMl | Ver	0	Updating capacity…	1	1	1	us-east-1d, us-east-1e

4. Create load balancer first

```
aws elbv2 create-load-balancer \
  --name my-load-balancer-instance \
  --subnets subnet-<> subnet-<> \
  --security-groups sg-<> \
  --type application
```

5. Create target group

```
aws elbv2 create-target-group \
  --name my-target-group-instance \
  --protocol HTTP \
  --port 80 \
  --vpc-id vpc-<> \
  --target-type instance
```

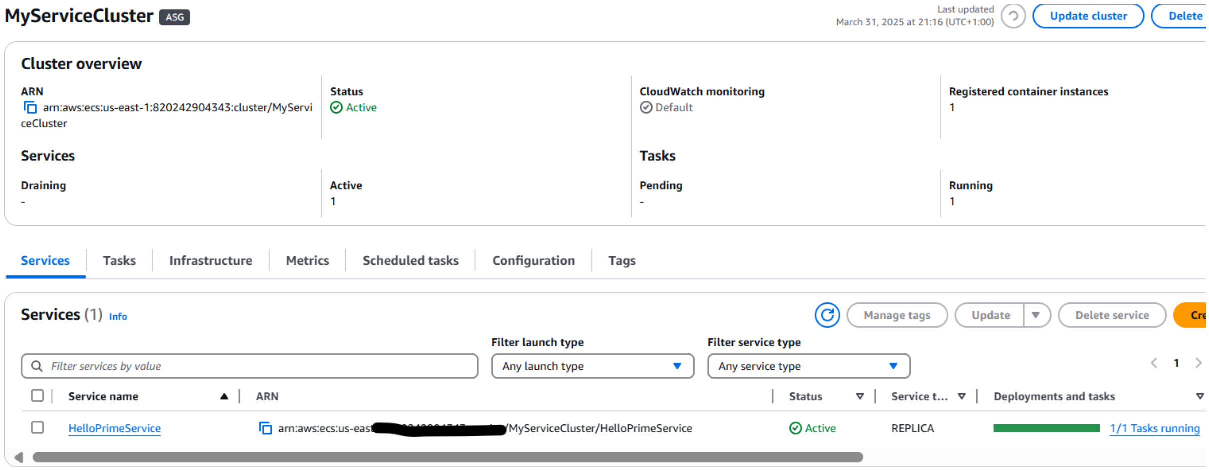
6. Create listener for target group

```
aws elbv2 create-listener \
  --load-balancer-arn arn:aws:elasticloadbalancing:us-east-1:<>:loadbalancer/app/my-load-balancer-instance/<> \
  --protocol HTTP \
  --port 80 \
  --default-actions Type=forward,TargetGroupArn=arn:aws:elasticloadbalancing:us-east-1:<>:targetgroup/my-target-group-instance/<>
```

7. Create the service

```
aws ecs create-service \
  --cluster MyServiceCluster \
  --service-name HelloPrimeService \
  --task-definition HelloWorldPrimeService \
  --desired-count 1 \
  --launch-type EC2 \
  --load-balancers "targetGroupArn=arn:aws:elasticloadbalancing:us-east-1:<>:targetgroup/my-target-group-instance/<>,containerName=HelloWorldPrime,containerPort=80"
```

The service is created which will call the SAAS application and circuit breaker enabled



MyServiceCluster ASG

Last updated March 31, 2025 at 21:16 (UTC+1:00)

Cluster overview

ARN	Status	CloudWatch monitoring	Registered container instances
arn:aws:ecs:us-east-1:820242904343:cluster/MyServiceCluster	Active	Default	1

Services

Draining	Active
-	1

Tasks

Pending	Running
-	1

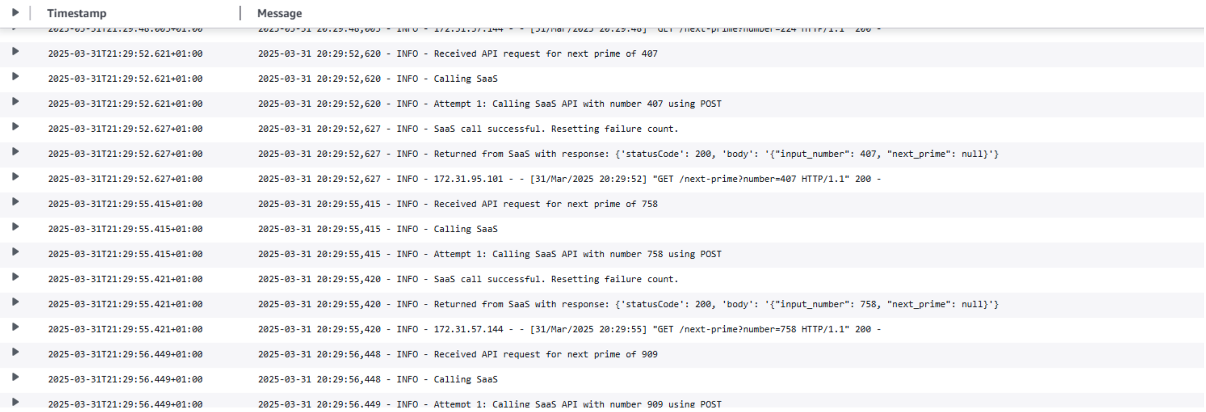
Services (1) Info

Filter launch type: Any launch type

Filter service type: Any service type

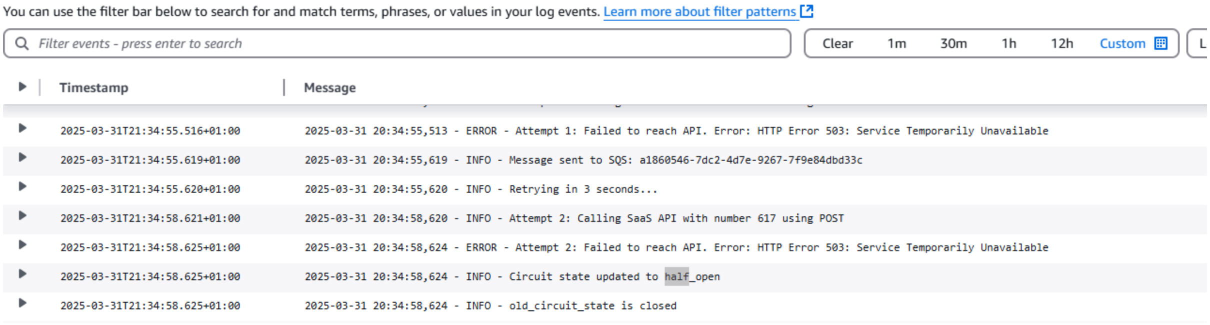
Service name	ARN	Status	Service t...	Deployments and tasks
HelloPrimeService	arn:aws:ecs:us-eas[redacted]/MyServiceCluster/HelloPrimeService	Active	REPLICA	1/1 Tasks running

8. Successful call to saas api. Circuit is closed.



```
Timestamp                       Message
2025-03-31T21:29:52.621+01:00   2025-03-31 20:29:52,620 - INFO - Received API request for next prime of 407
2025-03-31T21:29:52.621+01:00   2025-03-31 20:29:52,620 - INFO - Calling SaaS
2025-03-31T21:29:52.621+01:00   2025-03-31 20:29:52,620 - INFO - Attempt 1: Calling SaaS API with number 407 using POST
2025-03-31T21:29:52.627+01:00   2025-03-31 20:29:52,627 - INFO - SaaS call successful. Resetting failure count.
2025-03-31T21:29:52.627+01:00   2025-03-31 20:29:52,627 - INFO - Returned from SaaS with response: {'statusCode': 200, 'body': '{"input_number": 407, "next_prime": null}'}
2025-03-31T21:29:52.627+01:00   2025-03-31 20:29:52,627 - INFO - 172.31.95.101 - - [31/Mar/2025 20:29:52] "GET /next-prime?number=407 HTTP/1.1" 200 -
2025-03-31T21:29:55.415+01:00   2025-03-31 20:29:55,415 - INFO - Received API request for next prime of 758
2025-03-31T21:29:55.415+01:00   2025-03-31 20:29:55,415 - INFO - Calling SaaS
2025-03-31T21:29:55.415+01:00   2025-03-31 20:29:55,415 - INFO - Attempt 1: Calling SaaS API with number 758 using POST
2025-03-31T21:29:55.421+01:00   2025-03-31 20:29:55,420 - INFO - SaaS call successful. Resetting failure count.
2025-03-31T21:29:55.421+01:00   2025-03-31 20:29:55,420 - INFO - Returned from SaaS with response: {'statusCode': 200, 'body': '{"input_number": 758, "next_prime": null}'}
2025-03-31T21:29:55.421+01:00   2025-03-31 20:29:55,420 - INFO - 172.31.57.144 - - [31/Mar/2025 20:29:55] "GET /next-prime?number=758 HTTP/1.1" 200 -
2025-03-31T21:29:56.449+01:00   2025-03-31 20:29:56,448 - INFO - Received API request for next prime of 909
2025-03-31T21:29:56.449+01:00   2025-03-31 20:29:56,448 - INFO - Calling SaaS
2025-03-31T21:29:56.449+01:00   2025-03-31 20:29:56,449 - INFO - Attempt 1: Calling SaaS API with number 909 using POST
```

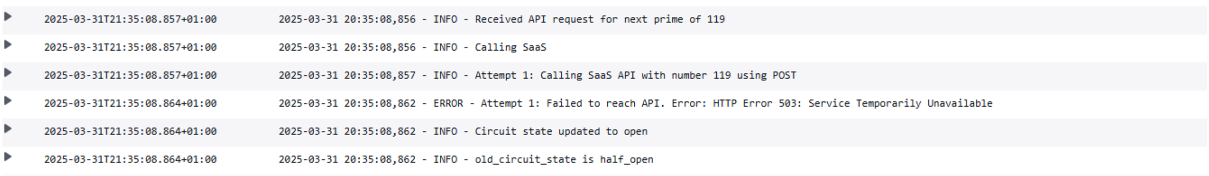
9. Unsuccessful calls. Circuit is updated to half open



You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

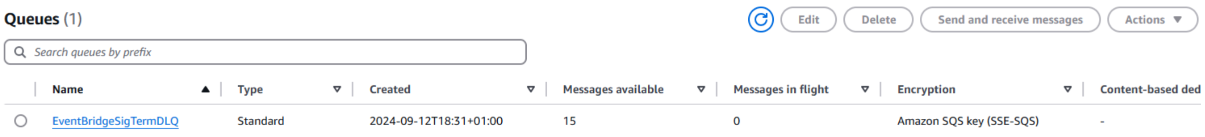
```
Timestamp                       Message
2025-03-31T21:34:55.516+01:00   2025-03-31 20:34:55,513 - ERROR - Attempt 1: Failed to reach API. Error: HTTP Error 503: Service Temporarily Unavailable
2025-03-31T21:34:55.619+01:00   2025-03-31 20:34:55,619 - INFO - Message sent to SQS: a1860546-7dc2-4d7e-9267-7f9e84dbd33c
2025-03-31T21:34:55.620+01:00   2025-03-31 20:34:55,620 - INFO - Retrying in 3 seconds...
2025-03-31T21:34:58.621+01:00   2025-03-31 20:34:58,620 - INFO - Attempt 2: Calling SaaS API with number 617 using POST
2025-03-31T21:34:58.625+01:00   2025-03-31 20:34:58,624 - ERROR - Attempt 2: Failed to reach API. Error: HTTP Error 503: Service Temporarily Unavailable
2025-03-31T21:34:58.625+01:00   2025-03-31 20:34:58,624 - INFO - Circuit state updated to half_open
2025-03-31T21:34:58.625+01:00   2025-03-31 20:34:58,624 - INFO - old_circuit_state is closed
```

10. Consecutive unsuccessful calls. Circuit is updated to closed.



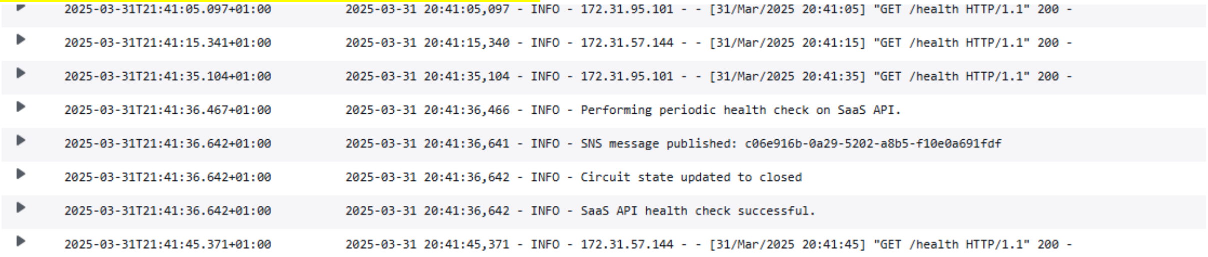
```
2025-03-31T21:35:08.857+01:00   2025-03-31 20:35:08,856 - INFO - Received API request for next prime of 119
2025-03-31T21:35:08.857+01:00   2025-03-31 20:35:08,856 - INFO - Calling SaaS
2025-03-31T21:35:08.857+01:00   2025-03-31 20:35:08,857 - INFO - Attempt 1: Calling SaaS API with number 119 using POST
2025-03-31T21:35:08.864+01:00   2025-03-31 20:35:08,862 - ERROR - Attempt 1: Failed to reach API. Error: HTTP Error 503: Service Temporarily Unavailable
2025-03-31T21:35:08.864+01:00   2025-03-31 20:35:08,862 - INFO - Circuit state updated to open
2025-03-31T21:35:08.864+01:00   2025-03-31 20:35:08,862 - INFO - old_circuit_state is half_open
```

11. All failed messages are saved in SQS



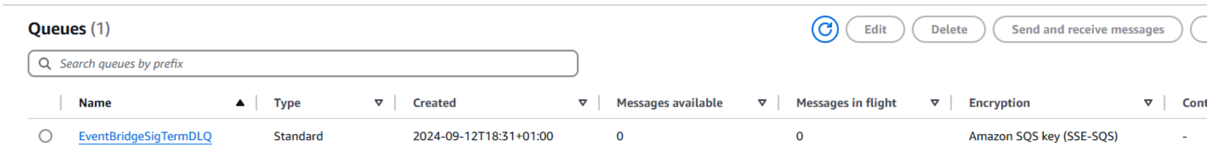
	Name	Type	Created	Messages available	Messages in flight	Encryption	Content-based ded
○	EventBridgeSigTermDLQ	Standard	2024-09-12T18:31+01:00	15	0	Amazon SQS key (SSE-SQS)	-

12. Auto health check is marking circuit to closed



```
▶  2025-03-31T21:41:05.097+01:00   2025-03-31 20:41:05,097 - INFO - 172.31.95.101 - - [31/Mar/2025 20:41:05] "GET /health HTTP/1.1" 200 -
▶  2025-03-31T21:41:15.341+01:00   2025-03-31 20:41:15,340 - INFO - 172.31.57.144 - - [31/Mar/2025 20:41:15] "GET /health HTTP/1.1" 200 -
▶  2025-03-31T21:41:35.104+01:00   2025-03-31 20:41:35,104 - INFO - 172.31.95.101 - - [31/Mar/2025 20:41:35] "GET /health HTTP/1.1" 200 -
▶  2025-03-31T21:41:36.467+01:00   2025-03-31 20:41:36,466 - INFO - Performing periodic health check on SaaS API.
▶  2025-03-31T21:41:36.642+01:00   2025-03-31 20:41:36,641 - INFO - SNS message published: c06e916b-0a29-5202-a8b5-f10e0a691fdf
▶  2025-03-31T21:41:36.642+01:00   2025-03-31 20:41:36,642 - INFO - Circuit state updated to closed
▶  2025-03-31T21:41:36.642+01:00   2025-03-31 20:41:36,642 - INFO - SaaS API health check successful.
▶  2025-03-31T21:41:45.371+01:00   2025-03-31 20:41:45,371 - INFO - 172.31.57.144 - - [31/Mar/2025 20:41:45] "GET /health HTTP/1.1" 200 -
```

13. When the circuit is closed it invokes the SNS which invokes another Lambda to clear SQS



	Name	Type	Created	Messages available	Messages in flight	Encryption	Cont
○	EventBridgeSigTermDLQ	Standard	2024-09-12T18:31+01:00	0	0	Amazon SQS key (SSE-SQS)	-