ASSIGNMENT 4.5

on

Cloud Deployment

Submitted by:

Haseebullah Shaikh (2303.KHI.DEG.015)

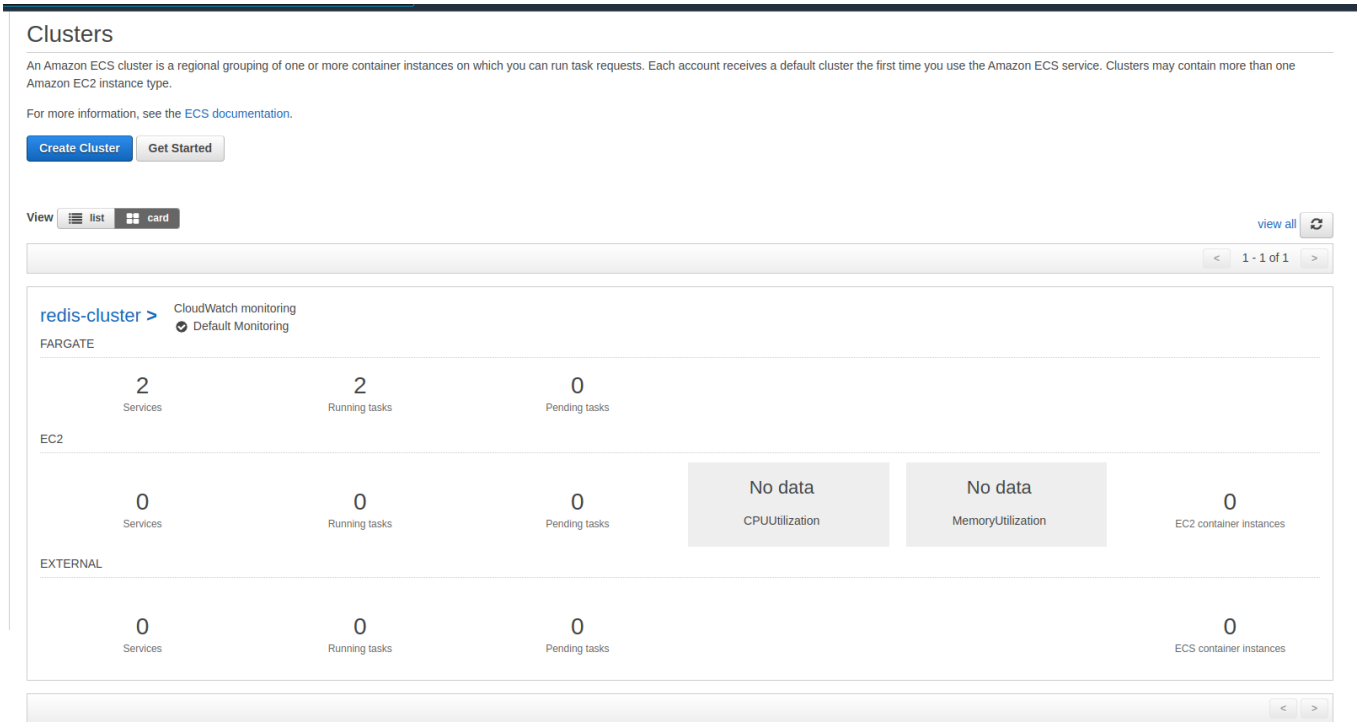
and

Faiza Gulzar Ahmed (2303.khi.deg.001)

Dated: 16th May 2023

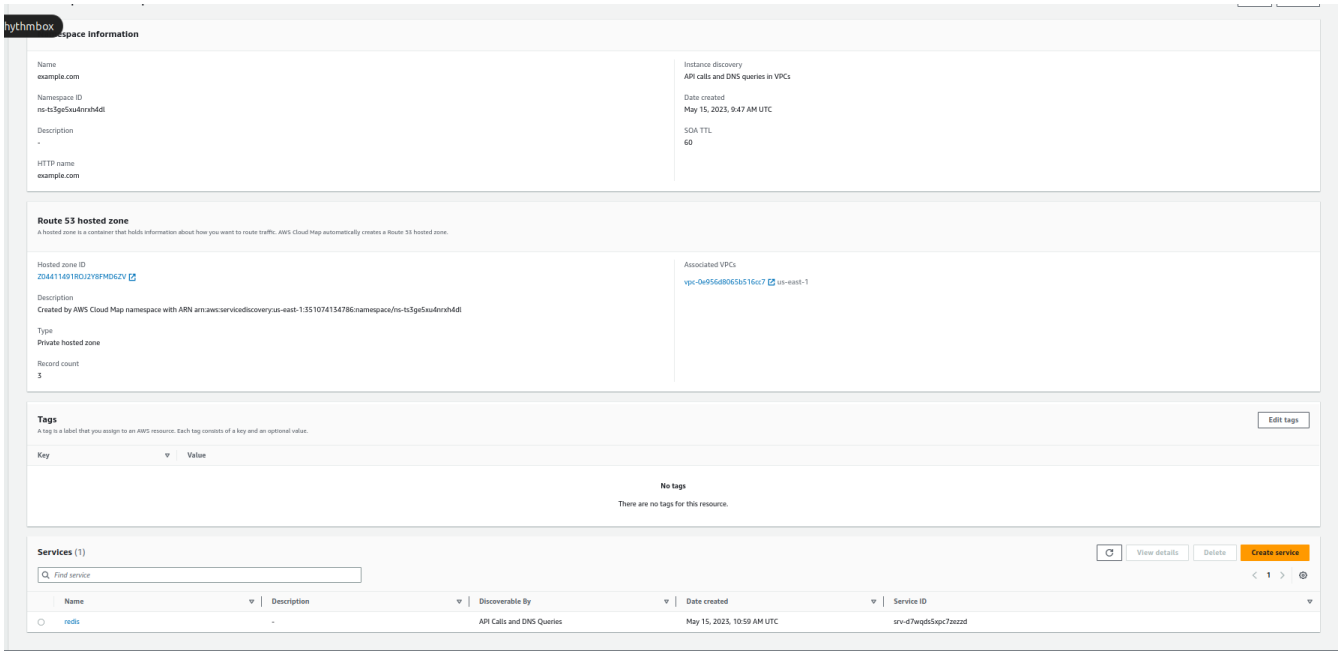
Solution:

1- Created AWS Cluster to get necessary resources such as VPC and subnets.

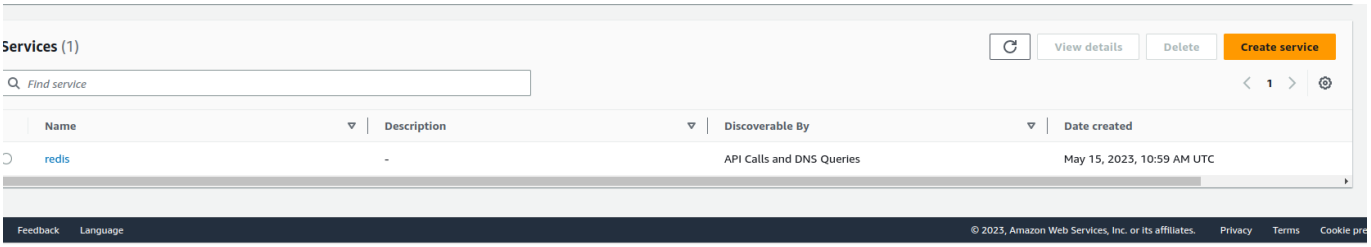


2- Setting up service recovery to enable commination between redis and counter app using same vpc.

Creating name space

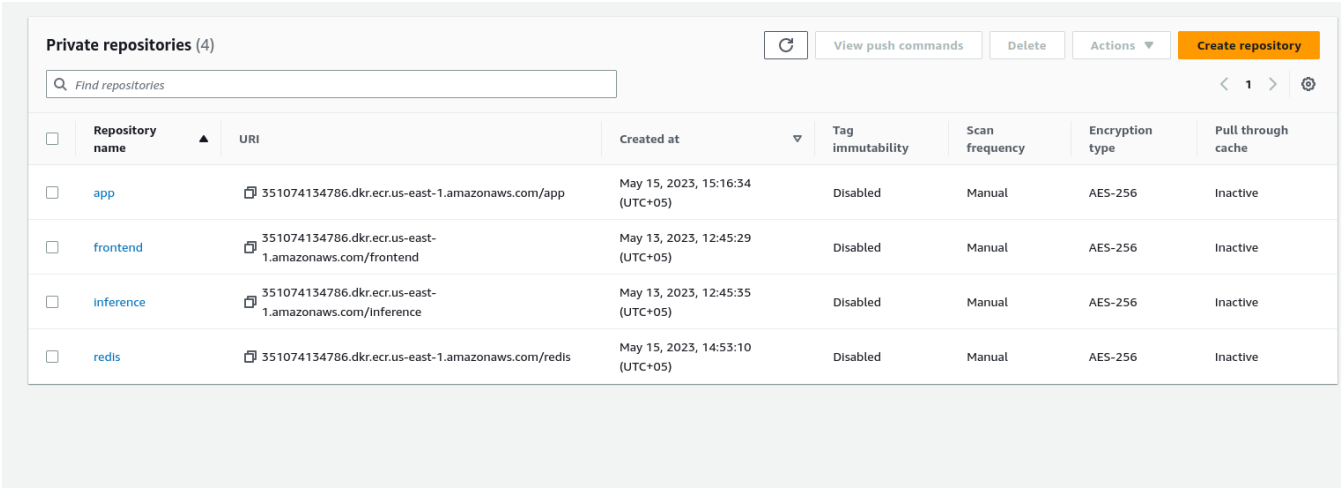


Creating Service in the namespace



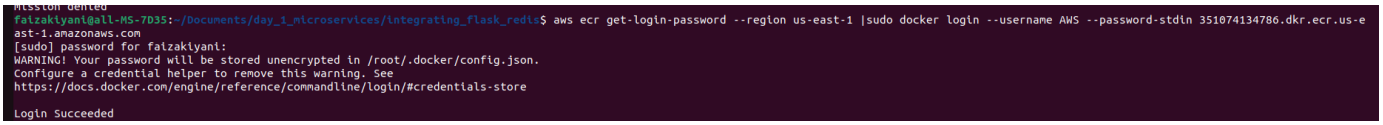
3. Built and pushed docker images to Amazon Container Registry

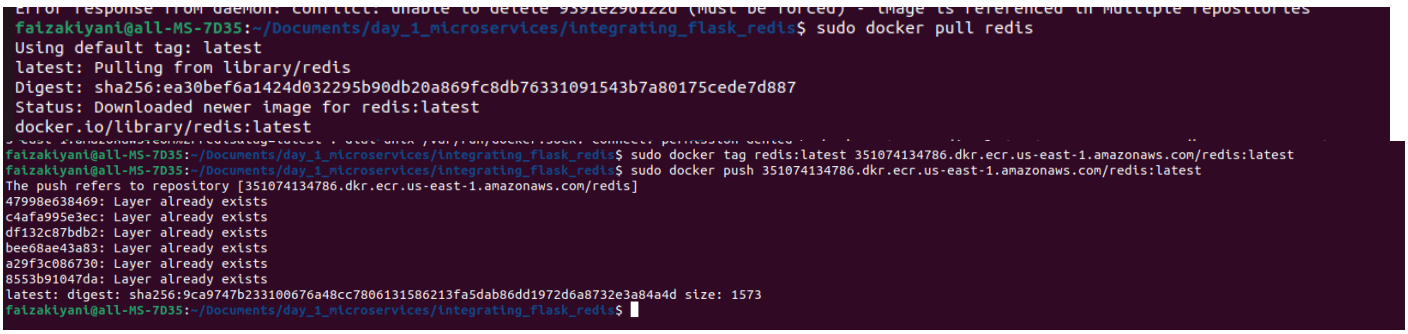
Created two private repositories redis and app.



Pushed the images in their respective repository using commands in view push commands.

Pulled and pushed redis image



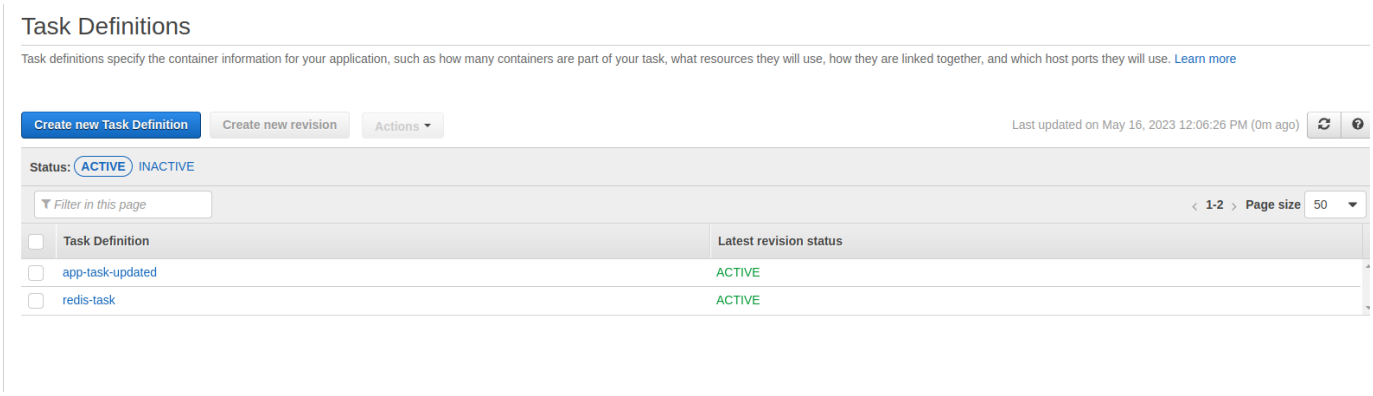


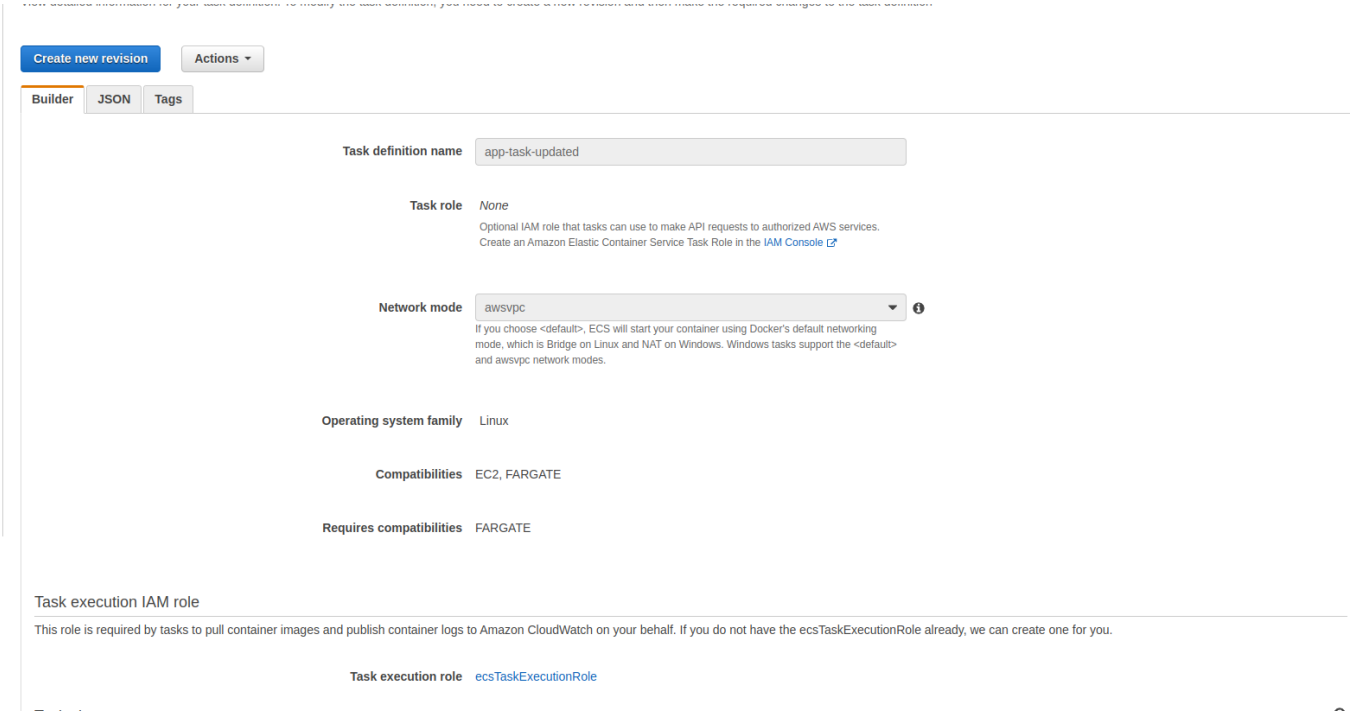
Built and pushed app image

```
faizakiyani@all-MS-7D35:~/Documents/day_1_microservices/integrating_flask_redis$ sudo docker build -t app .
Sending build context to Docker daemon   12.8kB
Step 1/10 : FROM python:3.7-alpine
 ---> e4fbc12a05a9
Step 2/10 : WORKDIR /code
 ---> Using cache
 ---> ea484d0f3a4a
Step 3/10 : ENV FLASK_APP=app.py
 ---> Using cache
 ---> f0c2e9a14817
Step 4/10 : ENV FLASK_RUN_HOST=0.0.0.0
 ---> Using cache
 ---> 643370ca2a73
Step 5/10 : RUN apk add --no-cache gcc musl-dev linux-headers
 ---> Using cache
 ---> e04ea13126c8
Step 6/10 : COPY requirements.txt requirements.txt
 ---> Using cache
 ---> f4815c5d01f8
Step 7/10 : RUN pip install -r requirements.txt
 ---> Using cache
 ---> 0312b1a3c0bd
Step 8/10 : EXPOSE 5000
 ---> Using cache
 ---> d81a0ac03ad2
Step 9/10 : COPY . .
 ---> Using cache
 ---> 58dd3cc4a426
Step 10/10 : CMD ["flask", "run"]
 ---> Using cache
 ---> 9391e296122d
Successfully built 9391e296122d
Successfully tagged app:latest
faizakiyani@all-MS-7D35:~/Documents/day_1_microservices/integrating_flask_redis$ docker tag app:latest 3510741
```

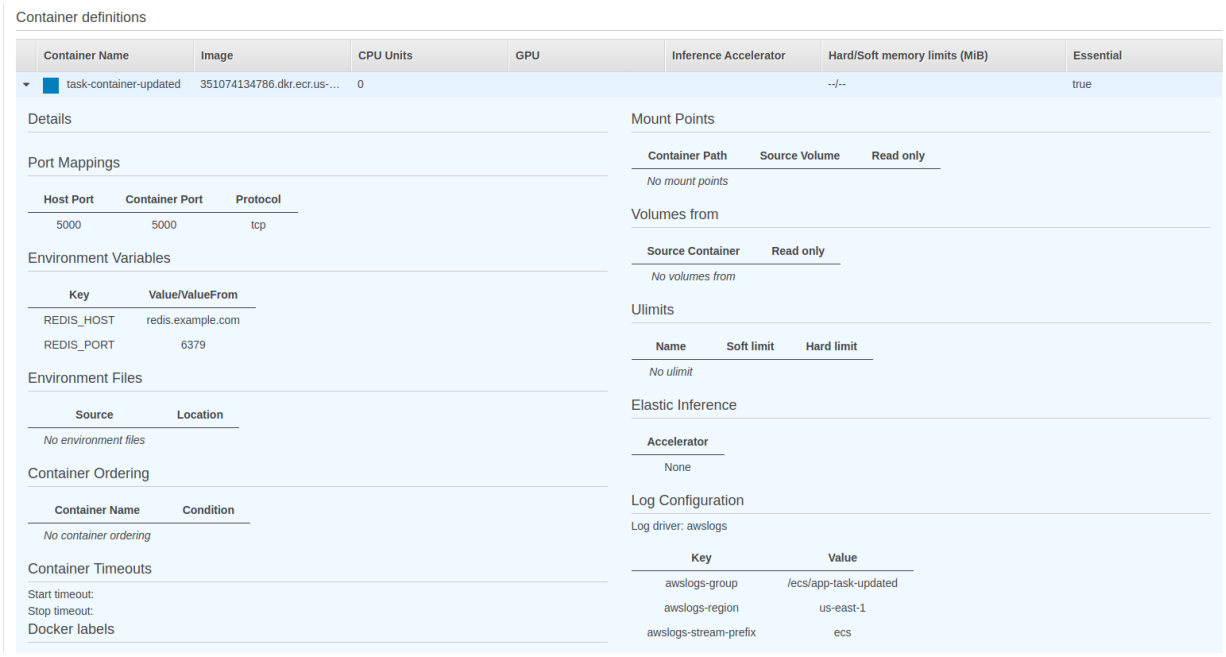
```
faizakiyani@all-MS-7D35:~/Documents/day_1_microservices/integrating_flask_redis$ sudo docker tag app:latest 351074134786.dkr.ecr.us-east-1.amazonaws.com/app:latest
faizakiyani@all-MS-7D35:~/Documents/day_1_microservices/integrating_flask_redis$ sudo docker push 351074134786.dkr.ecr.us-east-1.amazonaws.com/app:latest
The push refers to repository [351074134786.dkr.ecr.us-east-1.amazonaws.com/app]
c4f64a7eeb31: Pushed

d8f256de729c: Layer already exists

ddb074597168: Layer already exists

d4ffc3d535fd: Layer already exists

a13c86518dd1: Layer already exists

dc6a6a6fc818: Layer already exists

faf320a00dfc: Layer already exists

37411a7a419e: Layer already exists

208977ac81d7: Layer already exists

bb01bd7e32b5: Layer already exists

latest: digest: sha256:044939ec909badd03b7cd4a7a50706c86efbf05cf413d3fc271f4f17c6d7f270 size: 2412
```

4. Created task definition for each service specified provided memory and cpu, created container for each service to add image. Assigned appropriate ports and enviroment variables.





Setting the enviroment variables REDIS_HOST and REDIS_PORT, that we have updated in existing given app.



Container definitions

Container Name	Image	CPU Units	GPU	Inference Accelerator	Hard/Soft memory limits (MiB)	Essential
task-container-updated	351074134786.dkr.ecr.us-…	0			--/--	true

Details

Port Mappings

Host Port	Container Port	Protocol
5000	5000	tcp

Environment Variables

Key	Value/ValueFrom
REDIS_HOST	redis.example.com
REDIS_PORT	6379

Environment Files

Source	Location
No environment files	

Container Ordering

Container Name	Condition
No container ordering	

Container Timeouts

Start timeout:
Stop timeout:

Docker labels

Mount Points

Container Path	Source Volume	Read only
No mount points		

Volumes from

Source Container	Read only
No volumes from	

Ulimits

Name	Soft limit	Hard limit
No ulimit		

Elastic Inference

Accelerator
None

Log Configuration

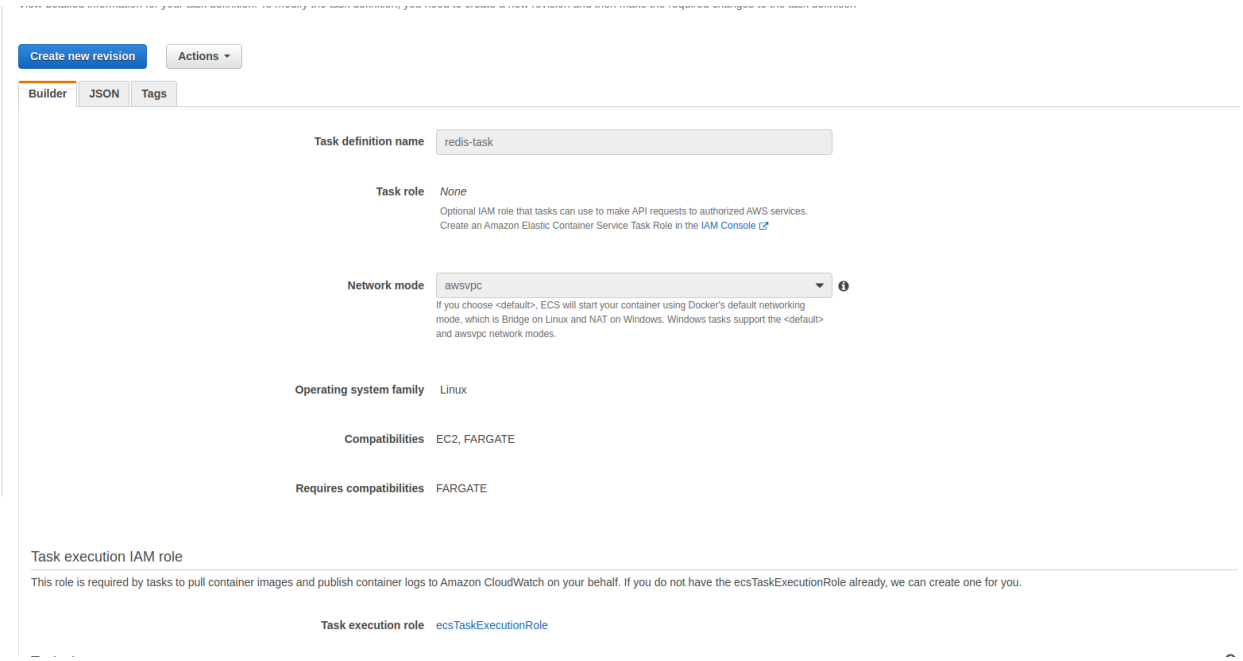
Log driver: awslogs

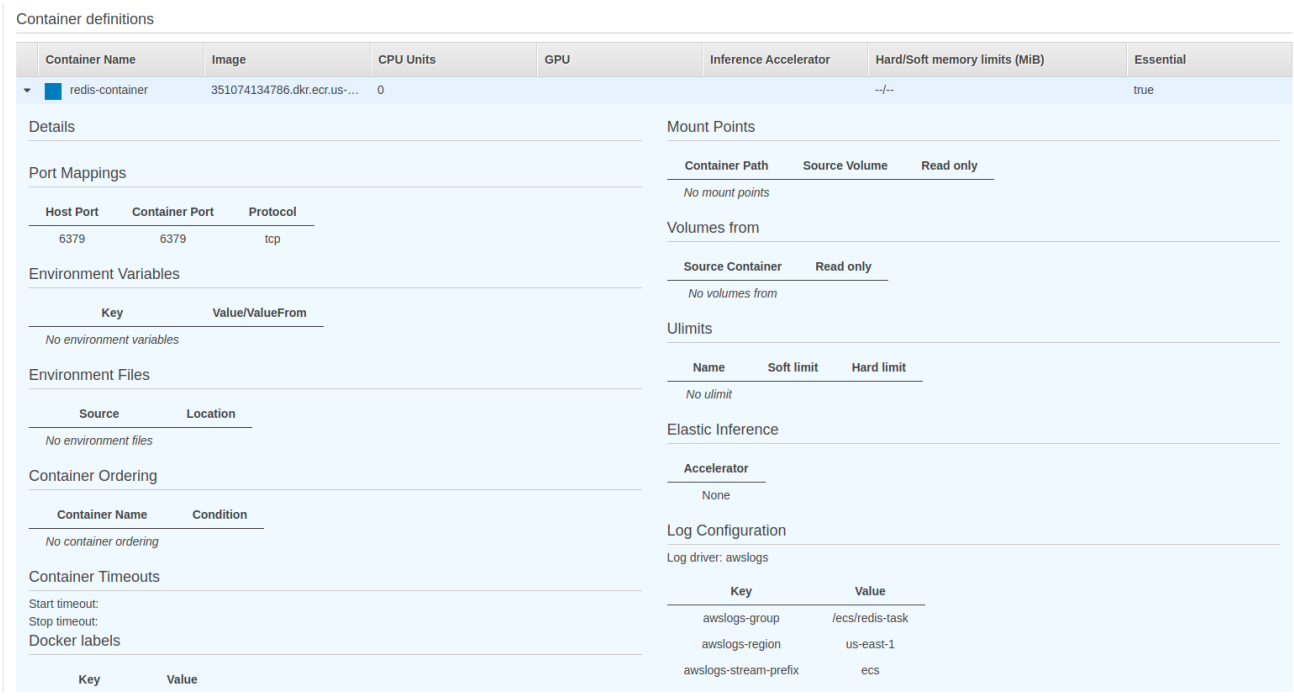
Key	Value
awslogs-group	/ecs/app-task-updated
awslogs-region	us-east-1
awslogs-stream-prefix	ecs

Updated app for creating enviroment variables, so app service can access the redis through these enviroment variables.

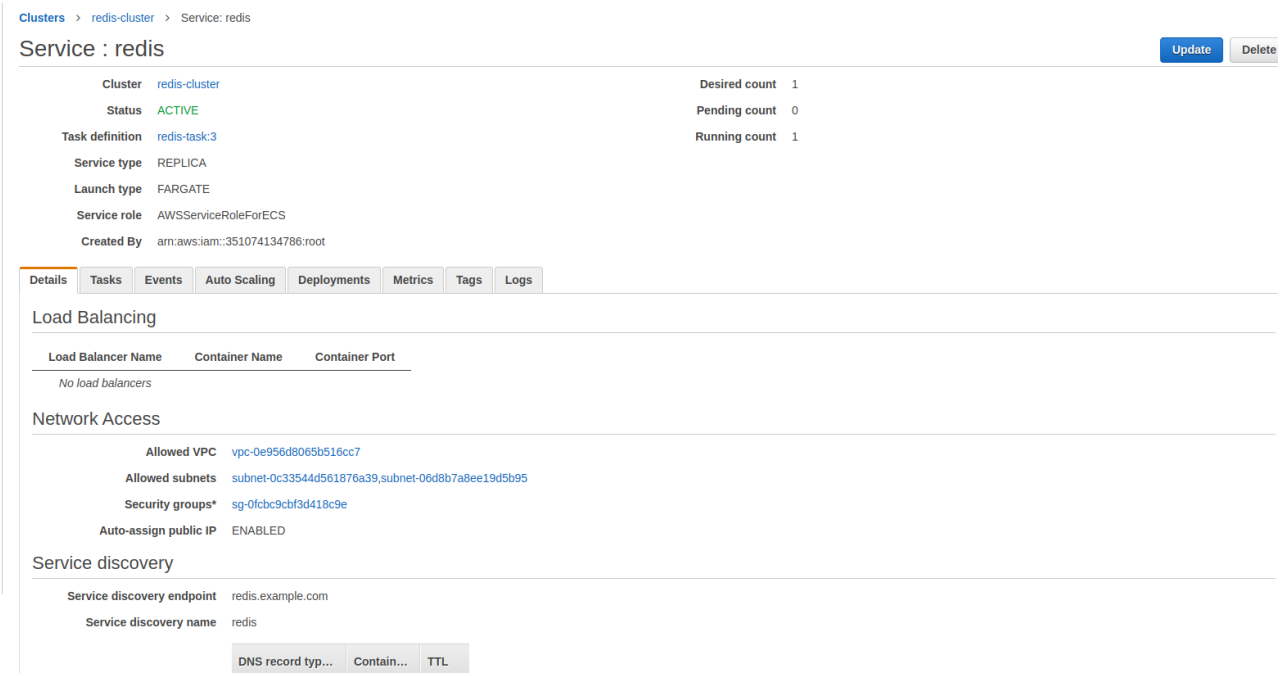
```
app = Flask(__name__)
redis_host = os.environ.get('REDIS_HOST')
redis_port = int(os.environ.get('REDIS_PORT'))
print('My Redis host: ', redis_host)
print('My Redis post: ', redis_port)
```

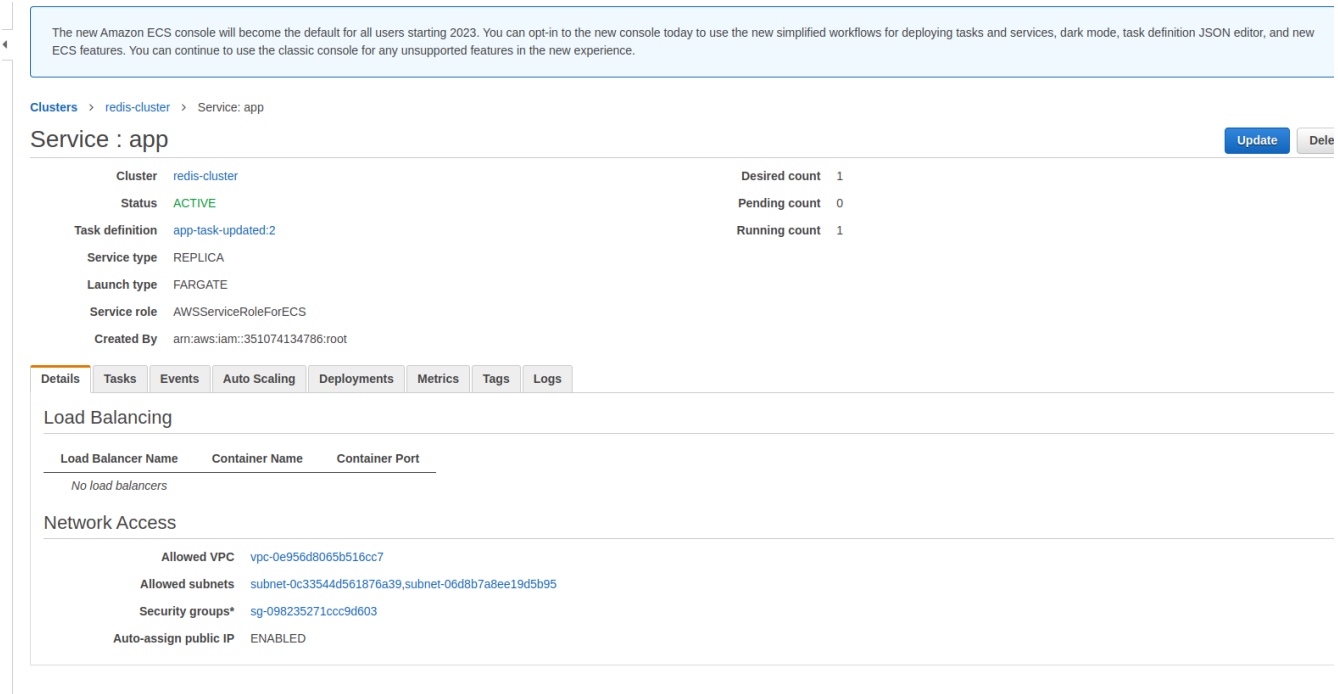
Creating task for redis service adding container for image and setting appropriate port.



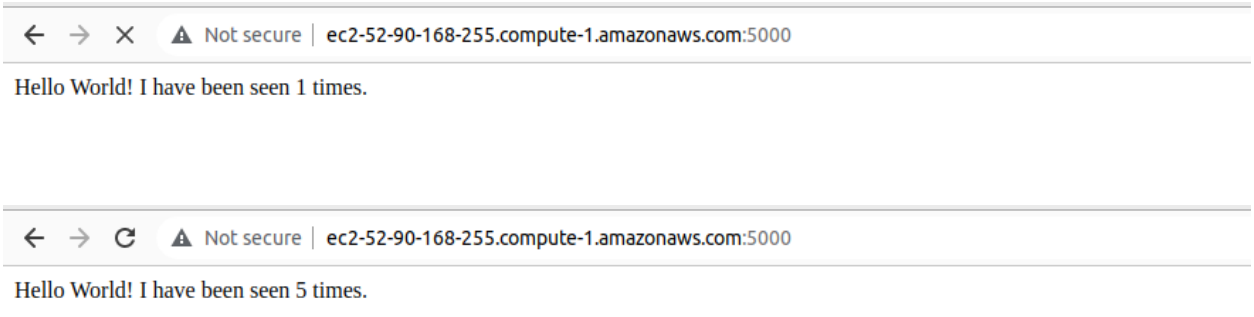


5. Created two ECS services redis and app in previously created cluster.





Verifying app is accessible at port 5000 and can communicate with redis for storing visits counts in cache.



The End ☺