

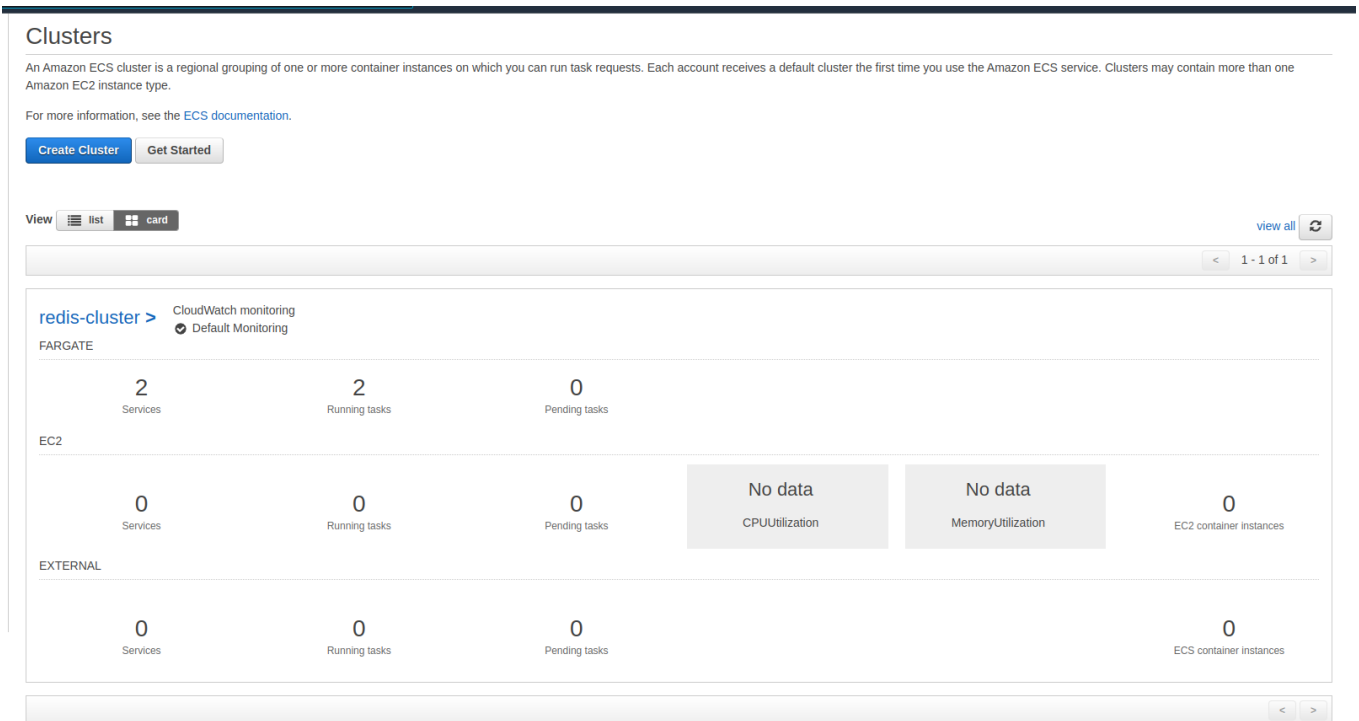
ASSIGNMENT 4.5
on
Cloud Deployment

Submitted by:
Haseebullah Shaikh (2303.KHI.DEG.015)
and
Faiza Gulzar Ahmed (2303.khi.deg.001)

Dated: 16th May 2023

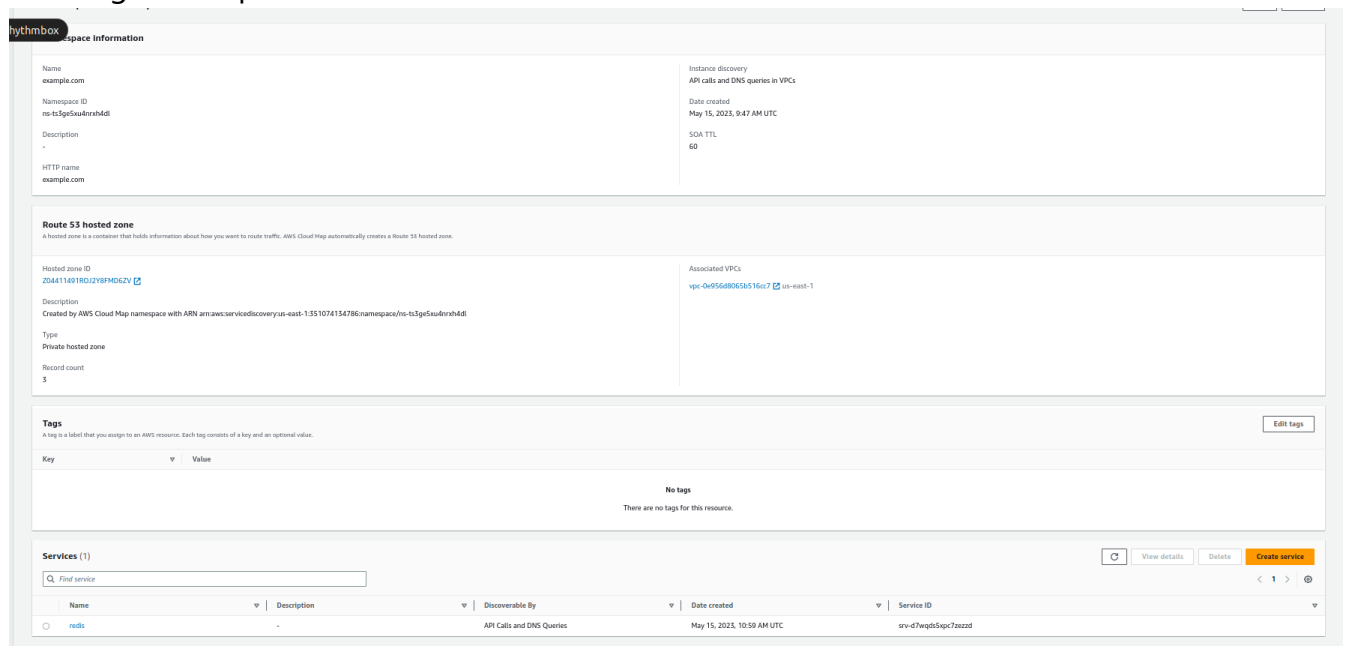
Solution:

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

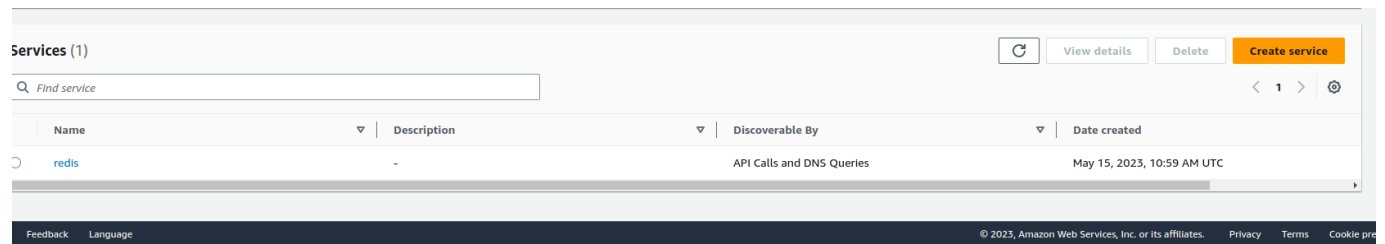


2- Setting up service recovery to enable commination between redis and counter app using same vpc.

Creating name space

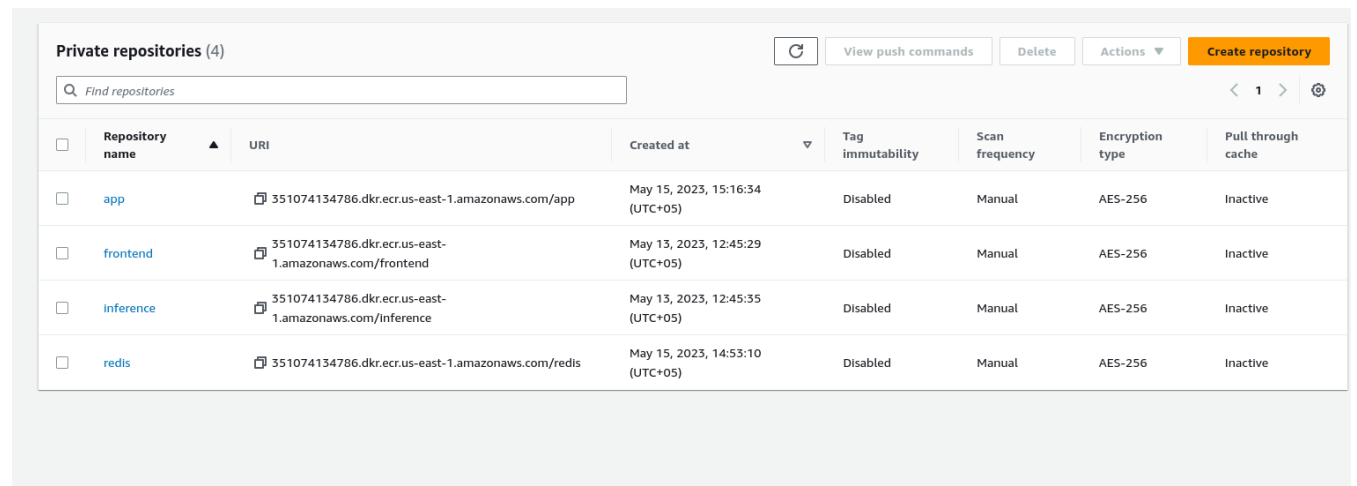


Creating Service in the namespace



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

```
faizakiyani@all-MS-7035: ~/Documents/day_1_microservices/integrating_flask_redis$ aws ecr get-login-password --region us-east-1 | sudo docker login --username AWS --password-stdin 351074134786.dkr.ecr.us-east-1.amazonaws.com
[sudo] password for faizakiyani:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded

faizakiyani@all-MS-7035: ~/Documents/day_1_microservices/integrating_flask_redis$ sudo docker pull redis
Error response from daemon: Conflict: unable to delete 9391e296122d (must be forced) - image is referenced in multiple repositories
Using default tag: latest
latest: Pulling from library/redis
Digest: sha256:ea30bef6a1424d032295b99db20a869fc8db76331091543b7a80175ced7d887
Status: Downloaded newer image for redis:latest
docker.io/library/redis:latest
faizakiyani@all-MS-7035: ~/Documents/day_1_microservices/integrating_flask_redis$ sudo docker tag redis:latest 351074134786.dkr.ecr.us-east-1.amazonaws.com/redis:latest
faizakiyani@all-MS-7035: ~/Documents/day_1_microservices/integrating_flask_redis$ sudo docker push 351074134786.dkr.ecr.us-east-1.amazonaws.com/redis:latest
The push refers to repository [351074134786.dkr.ecr.us-east-1.amazonaws.com/redis]
47998e638469: Layer already exists
c4afa995e3ec: Layer already exists
df132c87bdb2: Layer already exists
bee68ae43a83: Layer already exists
a29f3c086730: Layer already exists
8553b91047da: Layer already exists
latest: digest: sha256:9ca9747b233100676a48cc7806131586213fa5dab86dd1972d6a8732e3a84a4d size: 1573
faizakiyani@all-MS-7035: ~/Documents/day_1_microservices/integrating_flask_redis$
```

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
----> 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 351074134786.dkr.ecr.us-east-1.amazonaws.com/app:latest
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
faizakiyani@all-MS-7D35:~/Documents/day_1_microservices/integrating_flask_redis$
```

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.

Task Definitions

Task definitions specify the container information for your application, such as how many containers are part of your task, what resources they will use, how they are linked together, and which host ports they will use. [Learn more](#)

Create new Task Definition Create new revision Actions Last updated on May 16, 2023 12:06:26 PM (0m ago)	
Status: ACTIVE INACTIVE	
<input type="text" value="Filter in this page"/> < 1-2 > Page size 50	
<input type="checkbox"/> Task Definition	Latest revision status
<input type="checkbox"/> app-task-updated	ACTIVE
<input type="checkbox"/> redis-task	ACTIVE

When creating a new revision for your task definition, you need to create a new revision and then make the required changes to the task definition.

[Create new revision](#) [Actions](#)

[Builder](#) [JSON](#) [Tags](#)

Task definition name

Task role **None**
Optional IAM role that tasks can use to make API requests to authorized AWS services.
Create an Amazon Elastic Container Service Task Role in the [IAM Console](#)

Network mode ⓘ
If you choose <default>, ECS will start your container using Docker's default networking mode, which is Bridge on Linux and NAT on Windows. Windows tasks support the <default> and awsvpc network modes.

Operating system family **Linux**

Compatibilities **EC2, FARGATE**

Requires compatibilities **FARGATE**

Task execution IAM role

This role is required by tasks to pull container images and publish container logs to Amazon CloudWatch on your behalf. If you do not have the `ecsTaskExecutionRole` already, we can create one for you.

Task execution role [ecsTaskExecutionRole](#)

Setting the environment 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
<div><div></div>task-container-updated</div>	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 environment variables, so app service can access the redis through these environment 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 port: ', redis_port)
```

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

Create new revision

Actions

Builder

JSON

Tags

Task definition name

redis-task

Task role

None

Optional IAM role that tasks can use to make API requests to authorized AWS services.
Create an Amazon Elastic Container Service Task Role in the [IAM Console](#)

Network mode

awsvpc

If you choose <default>, ECS will start your container using Docker's default networking mode, which is Bridge on Linux and NAT on Windows. Windows tasks support the <default> and awsvpc network modes.

Operating system family

Linux

Compatibilities

EC2, FARGATE

Requires compatibilities

FARGATE

Task execution IAM role

This role is required by tasks to pull container images and publish container logs to Amazon CloudWatch on your behalf. If you do not have the ecsTaskExecutionRole already, we can create one for you.

Task execution role

[ecsTaskExecutionRole](#)

Container definitions

Container Name	Image	CPU Units	GPU	Inference Accelerator	Hard/Soft memory limits (MiB)	Essential
redis-container	351074134786.dkr.ecr.us-...	0			--/--	true
Details						
Port Mappings						
Host Port	Container Port	Protocol				
6379	6379	tcp				
Environment Variables						
Key	Value/ValueFrom					
No environment variables						
Environment Files						
Source	Location					
No environment files						
Container Ordering						
Container Name	Condition					
No container ordering						
Container Timeouts						
Start timeout:						
Stop timeout:						
Docker labels						
Key	Value					

Container Path	Source Volume	Read only
No mount points		
Volumes from		
Source Container	Read only	
No volumes from		
Limits		
Name	Soft limit	Hard limit
No ulimit		
Elastic Inference		
Accelerator		
None		
Log Configuration		
Log driver: awslogs		
Key	Value	
awslogs-group	/ecs/redis-task	
awslogs-region	us-east-1	
awslogs-stream-prefix	ecs	

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

Clusters > redis-cluster > Service: redis

Service : redis

Update Delete

Cluster	redis-cluster	Desired count	1
Status	ACTIVE	Pending count	0
Task definition	redis-task:3	Running count	1
Service type	REPLICA		
Launch type	FARGATE		
Service role	AWSServiceRoleForECS		
Created By	arn:aws:iam::351074134786:root		

Details Tasks Events Auto Scaling Deployments Metrics Tags Logs

Load Balancing

Load Balancer Name	Container Name	Container Port
No load balancers		

Network Access

Allowed VPC	vpc-0e956d8065b516cc7
Allowed subnets	subnet-0c33544d561876a39,subnet-06d8b7a8ee19d5b95
Security groups*	sg-0fcbc9cb3d418c9e
Auto-assign public IP	ENABLED

Service discovery

Service discovery endpoint	redis.example.com
Service discovery name	redis

DNS record typ...	Containin...	TTL
-------------------	--------------	-----

The new Amazon ECS console will become the default for all users starting 2023. You can opt-in to the new console today to use the new simplified workflows for deploying tasks and services, dark mode, task definition JSON editor, and new ECS features. You can continue to use the classic console for any unsupported features in the new experience.

Clusters > redis-cluster > Service: app

Service : app

Update Dele

Cluster	redis-cluster	Desired count	1
Status	ACTIVE	Pending count	0
Task definition	app-task-updated:2	Running count	1
Service type	REPLICA		
Launch type	FARGATE		
Service role	AWSServiceRoleForECS		
Created By	arn:aws:iam::351074134786:root		

Details Tasks Events Auto Scaling Deployments Metrics Tags Logs

Load Balancing

Load Balancer Name	Container Name	Container Port
No load balancers		

Network Access

Allowed VPC	vpc-0e956d8065b516cc7
Allowed subnets	subnet-0c33544d561876a39,subnet-06d8b7a8ee19d5b95
Security groups*	sg-098235271ccc9d603
Auto-assign public IP	ENABLED

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

← → × ⚠ Not secure | ec2-52-90-168-255.compute-1.amazonaws.com:5000

Hello World! I have been seen 1 times.

← → ↻ ⚠ Not secure | ec2-52-90-168-255.compute-1.amazonaws.com:5000

Hello World! I have been seen 5 times.

The End 😊