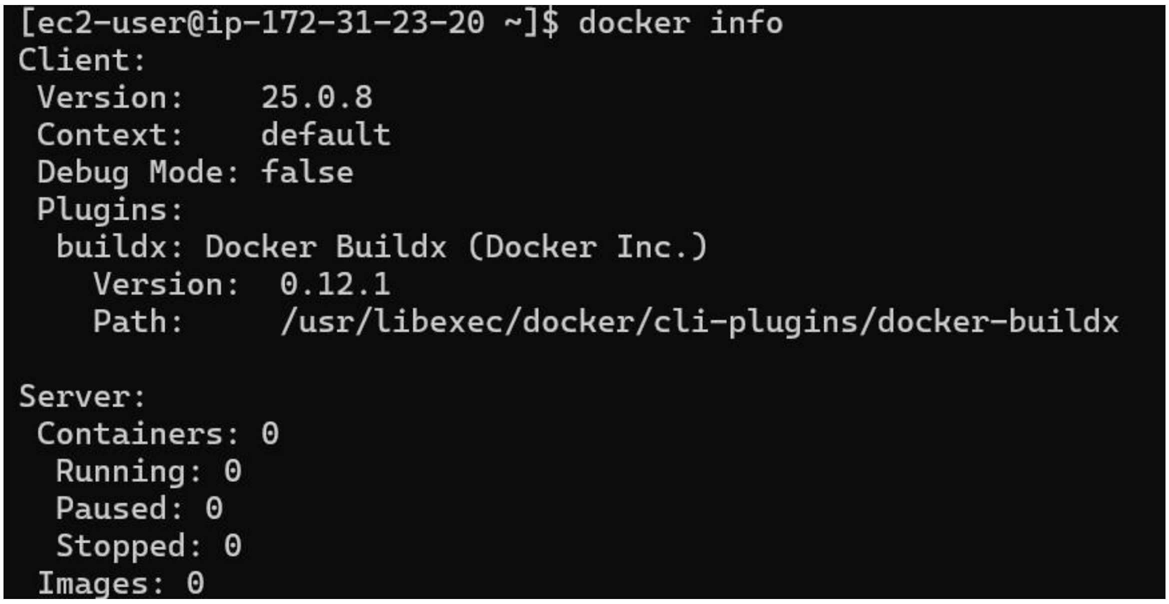
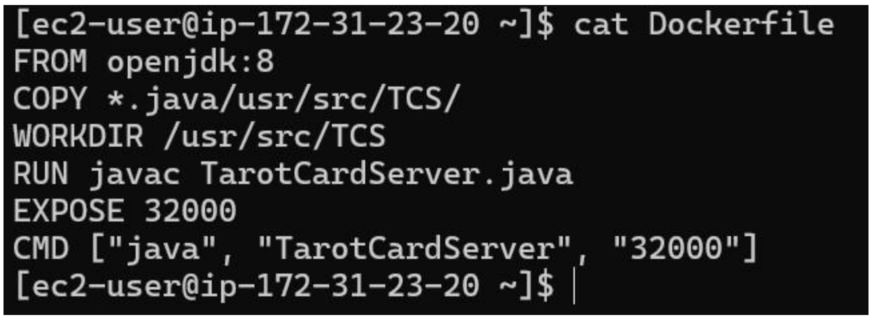
makhmemaks 300662813



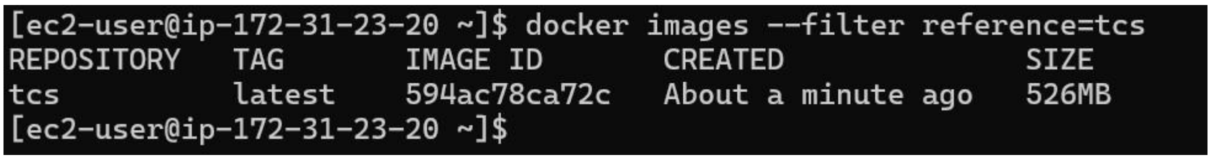
```
[ec2-user@ip-172-31-23-20 ~]$ docker info
Client:
 Version:    25.0.8
 Context:    default
 Debug Mode: false
 Plugins:
  buildx: Docker Buildx (Docker Inc.)
    Version:  0.12.1
    Path:     /usr/libexec/docker/cli-plugins/docker-buildx

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
```

Docker is installed on the ec2 server.



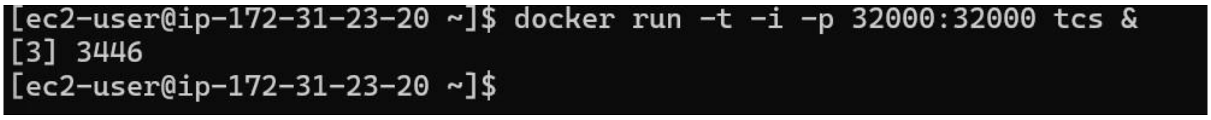
```
[ec2-user@ip-172-31-23-20 ~]$ cat Dockerfile
FROM openjdk:8
COPY *.java/usr/src/TCS/
WORKDIR /usr/src/TCS
RUN javac TarotCardServer.java
EXPOSE 32000
CMD ["java", "TarotCardServer", "32000"]
[ec2-user@ip-172-31-23-20 ~]$
```

A file called "Dockerfile" is created with the port the server TarotCardServer uses (32000).



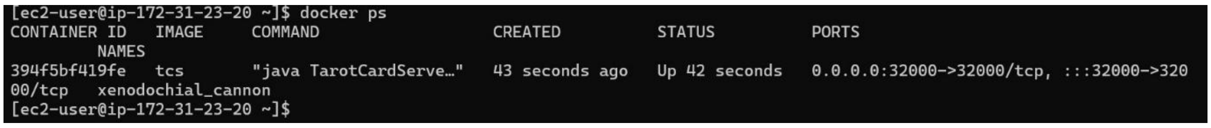
```
[ec2-user@ip-172-31-23-20 ~]$ docker images --filter reference=tcs
REPOSITORY   TAG       IMAGE ID       CREATED              SIZE
tcs          latest    594ac78ca72c   About a minute ago   526MB
[ec2-user@ip-172-31-23-20 ~]$
```

After using docker build -t tcs, we built a new Docker image. We checked if the image was built correctly using docker images –filter reference=tcs. It was built correctly as per the screen shot above.



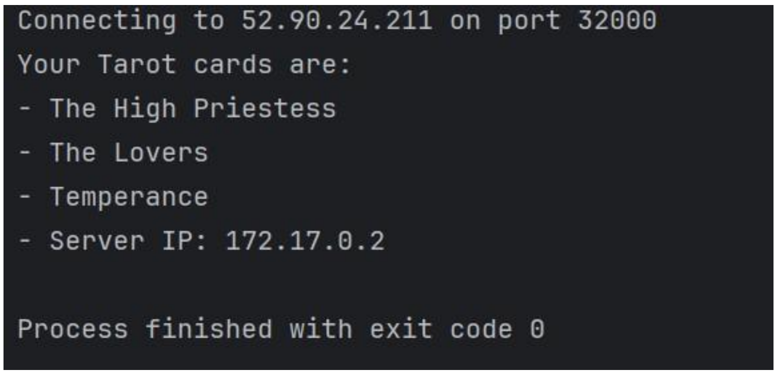
```
[ec2-user@ip-172-31-23-20 ~]$ docker run -t -i -p 32000:32000 tcs &
[3] 3446
[ec2-user@ip-172-31-23-20 ~]$
```

We mapped the exposed port on the container to the port on our host machine, using docker run -t -i -p 32000:32000 tcs & and received the response 3446.



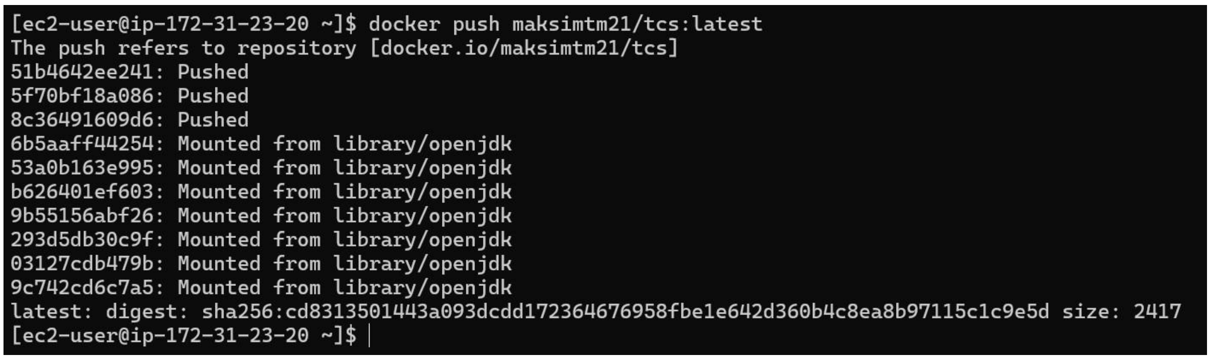
```
[ec2-user@ip-172-31-23-20 ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS
        NAMES
394f5bf419fe   tcs       "java TarotCardServe…"   43 seconds ago   Up 42 seconds   0.0.0.0:32000->32000/tcp, :::32000->320
00/tcp   xenodochial_cannon
[ec2-user@ip-172-31-23-20 ~]$
```

We made sure everything was running correctly using docker ps.



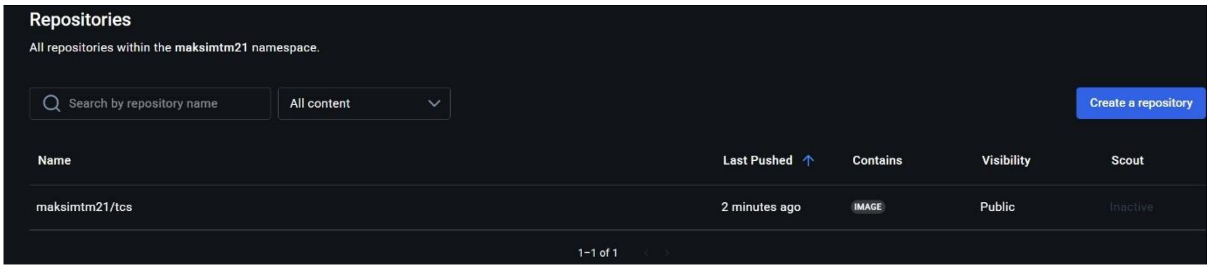
```
Connecting to 52.90.24.211 on port 32000
Your Tarot cards are:
- The High Priestess
- The Lovers
- Temperance
- Server IP: 172.17.0.2

Process finished with exit code 0
```

Running my client on my host machine we see everything working well and a response from the server.



```
[ec2-user@ip-172-31-23-20 ~]$ docker push maksimtm21/tcs:latest
The push refers to repository [docker.io/maksimtm21/tcs]
51b4642ee241: Pushed
5f70bf18a086: Pushed
8c36491609d6: Pushed
6b5aaff44254: Mounted from library/openjdk
53a0b163e995: Mounted from library/openjdk
b626401ef603: Mounted from library/openjdk
9b55156abf26: Mounted from library/openjdk
293d5db30c9f: Mounted from library/openjdk
03127cdb479b: Mounted from library/openjdk
9c742cd6c7a5: Mounted from library/openjdk
latest: digest: sha256:cd8313501443a093dcdd172364676958fbe1e642d360b4c8ea8b97115c1c9e5d size: 2417
[ec2-user@ip-172-31-23-20 ~]$ |
```

After logging into docker on the ec2, we used docker push maksimtm21/tcs:latest to push/upload onto my docker hub repository.

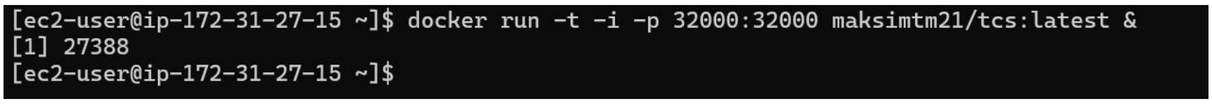


Repositories

All repositories within the **maksimtm21** namespace.

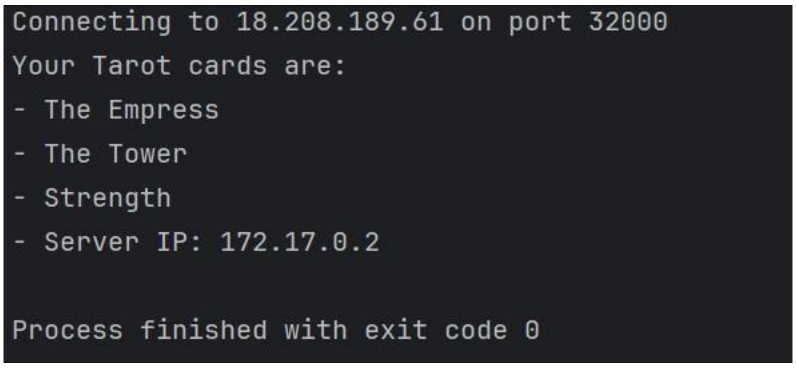
Name	Last Pushed ↑	Contains	Visibility	Scout
maksimtm21/tcs	2 minutes ago	IMAGE	Public	Inactive

Here is our tcs uploaded and stored on my docker repository.



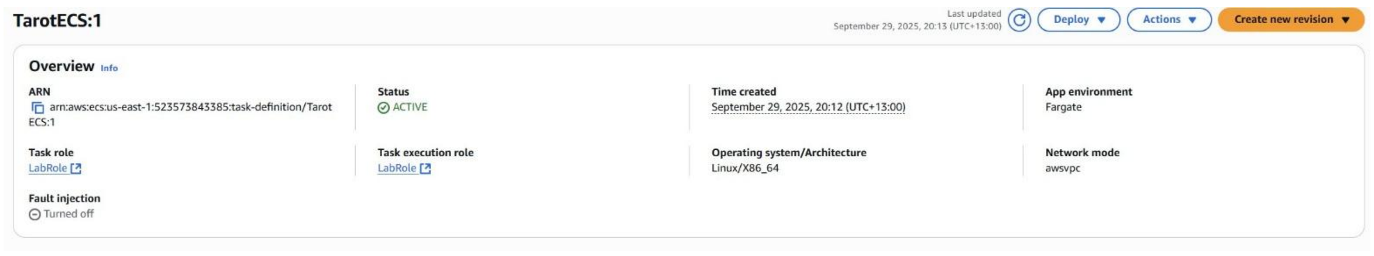
```
[ec2-user@ip-172-31-27-15 ~]$ docker run -t -i -p 32000:32000 maksimtm21/tcs:latest &
[1] 27388
[ec2-user@ip-172-31-27-15 ~]$
```

After creating a vanilla ec2 server we pulled the tcs from the docker repository using docker pull docker.io/maksimtm21/tcs/latest. Then we ran it using docker run -t -i -p 32000:32000 maksimtm21/tcs/latest &.



```
Connecting to 18.208.189.61 on port 32000
Your Tarot cards are:
- The Empress
- The Tower
- Strength
- Server IP: 172.17.0.2

Process finished with exit code 0
```

Then we tested the client with the new ec2 vanilla server with the TarotCardServer repository we pulled from docker. It was functional and printed the expected results.

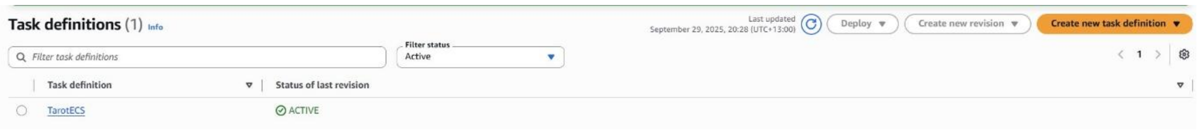


TarotECS:1

Last updated September 29, 2025, 20:13 (UTC+13:00)

Overview

ARN	Status	Time created	App environment
arn:aws:ecs:us-east-1:523573843385:task-definition/TarotECS:1	ACTIVE	September 29, 2025, 20:12 (UTC+13:00)	Fargate
Task role	Task execution role	Operating system/Architecture	Network mode
LabRole	LabRole	Linux/X86_64	awsvpc
Fault injection			
Turned off			



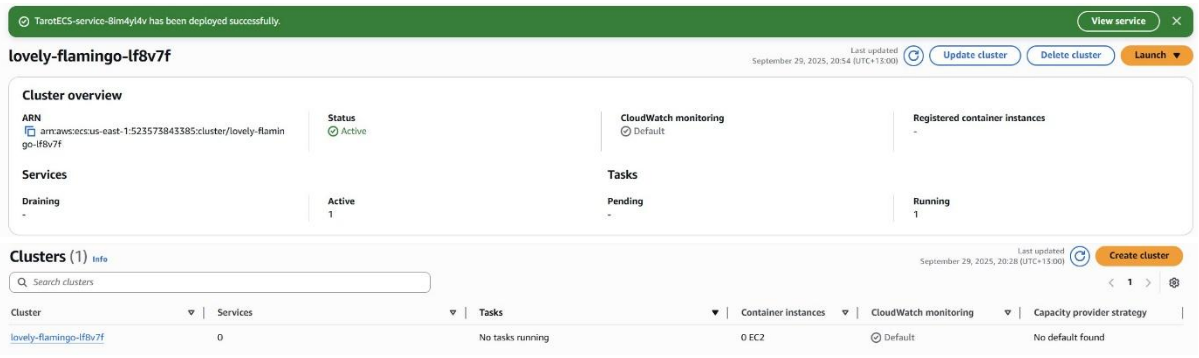
Task definitions (1)

Last updated September 29, 2025, 20:28 (UTC+13:00)

Filter status: Active

Task definition	Status of last revision
TarotECS	ACTIVE

This was the created ECS (TarotECS:1). It contained in the image box the location of my docker image from the docker repository (docker.io/maksimtm21/tcs:latest). Also, the port mapping at 32000.



TarotECS-service-8im4yl4v has been deployed successfully.

lovely-flamingo-lf8v7f

Last updated September 29, 2025, 20:54 (UTC+13:00)

Cluster overview

ARN	Status	CloudWatch monitoring	Registered container instances
arn:aws:ecs:us-east-1:523573843385:cluster/lovely-flamingo-lf8v7f	Active	Default	-

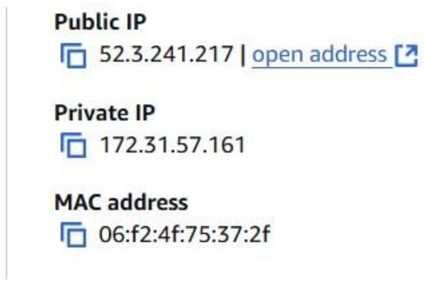
Services		Tasks	
Draining	Active	Pending	Running
-	1	-	1

Clusters (1)

Last updated September 29, 2025, 20:28 (UTC+13:00)

Cluster	Services	Tasks	Container instances	CloudWatch monitoring	Capacity provider strategy
lovely-flamingo-lf8v7f	0	No tasks running	0 EC2	Default	No default found

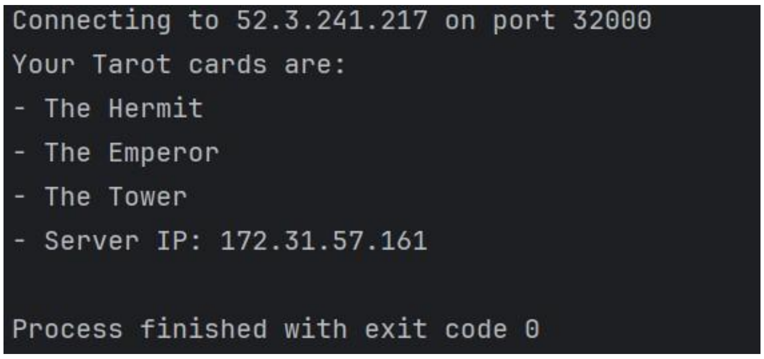
The created cluster (lovely-flamingo-lf8v7f). We populated the cluster with one task from the task definitions (TarotECS). We created it with a new security group.


Public IP
52.3.241.217 | open address

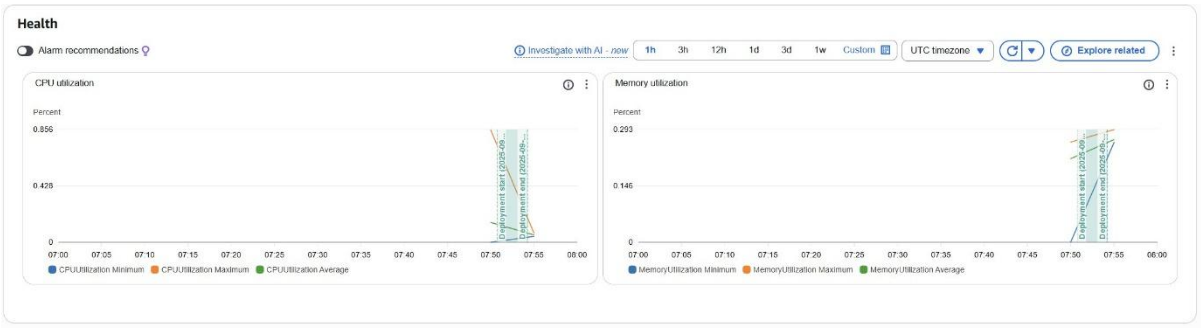
Private IP
172.31.57.161

MAC address
06:f2:4f:75:37:2f

This was the Public IP address 54.3.241.217 that was used with the client to test if our server was working.

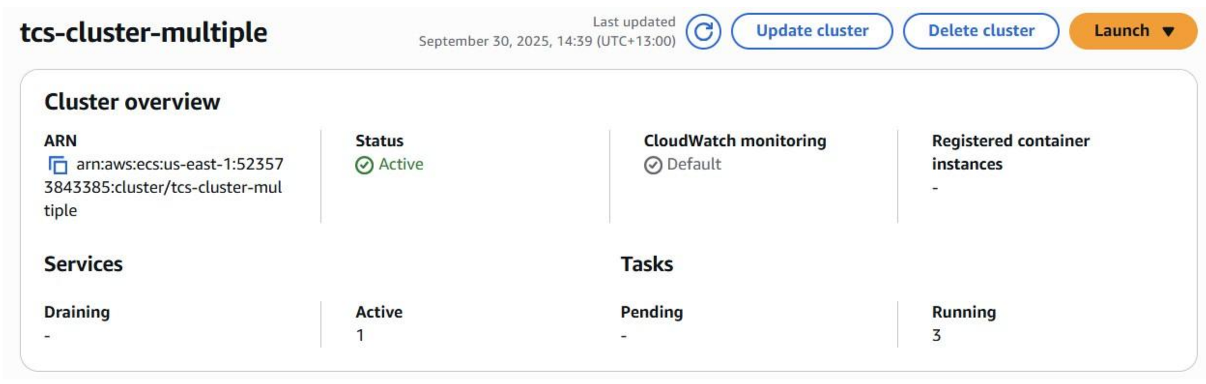

```
Connecting to 52.3.241.217 on port 32000
Your Tarot cards are:
- The Hermit
- The Emperor
- The Tower
- Server IP: 172.31.57.161

Process finished with exit code 0
```

Using the public IP address 54.3.241.217 it proved successful with appropriately working and displaying the three tarot cards.



This was the result of the single client call in the health pane.

Question 1:



tcs-cluster-multiple

Last updated
September 30, 2025, 14:39 (UTC+13:00)

Cluster overview

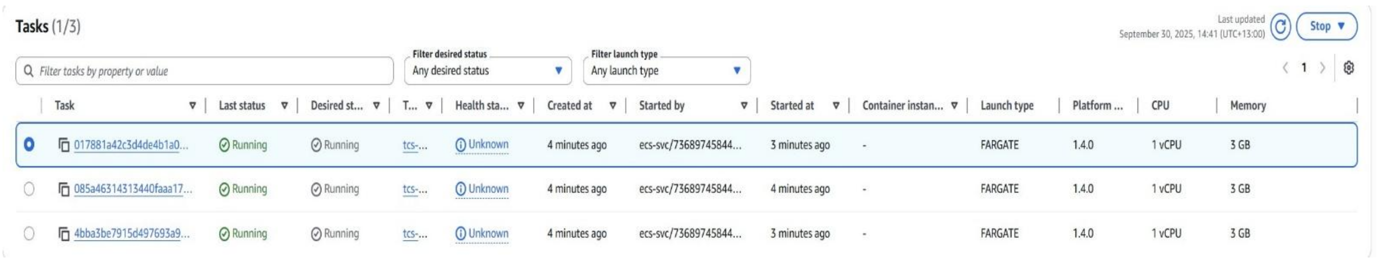
ARN	Status	CloudWatch monitoring	Registered container instances
arn:aws:ecs:us-east-1:523573843385:cluster/tcs-cluster-multiple	Active	Default	-

Services

Draining	Active
-	1

Tasks

Pending	Running
-	3

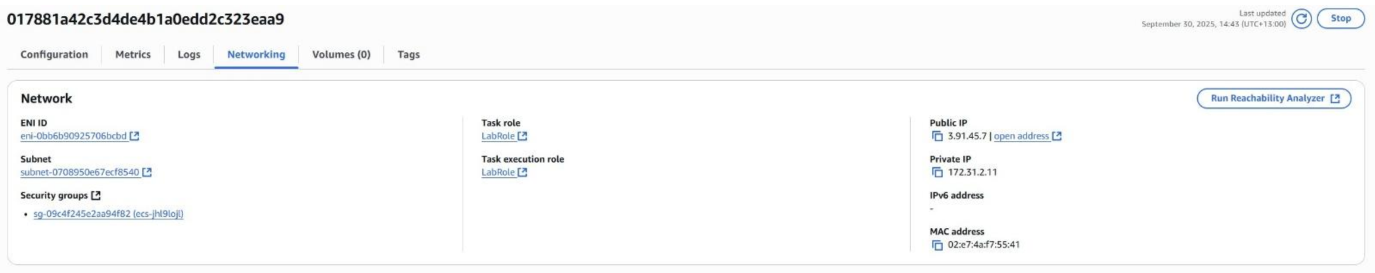


Tasks (1/3)

Last updated
September 30, 2025, 14:41 (UTC+13:00)

Filter desired status: Any desired status
Filter launch type: Any launch type

Task	Last status	Desired st...	T...	Health sta...	Created at	Started by	Started at	Container instan...	Launch type	Platform ...	CPU	Memory
017881a42c3d4de4b1a0...	Running	Running	tcs-...	Unknown	4 minutes ago	ecs-svc/73689745844...	3 minutes ago	-	FARGATE	1.4.0	1 vCPU	3 GB
085a46314313440faaa17...	Running	Running	tcs-...	Unknown	4 minutes ago	ecs-svc/73689745844...	4 minutes ago	-	FARGATE	1.4.0	1 vCPU	3 GB
4bba3be7915d497693a9...	Running	Running	tcs-...	Unknown	4 minutes ago	ecs-svc/73689745844...	3 minutes ago	-	FARGATE	1.4.0	1 vCPU	3 GB



017881a42c3d4de4b1a0edd2c323eaa9

Last updated
September 30, 2025, 14:43 (UTC+13:00)

Network

ENI ID
eni-0bb6b90925706bcbd

Subnet
subnet-0708950e67ecf8540

Security groups
- sg-09c4f245e2aa94f82 (ecs-jhl9lojl)

Task role
LabRole

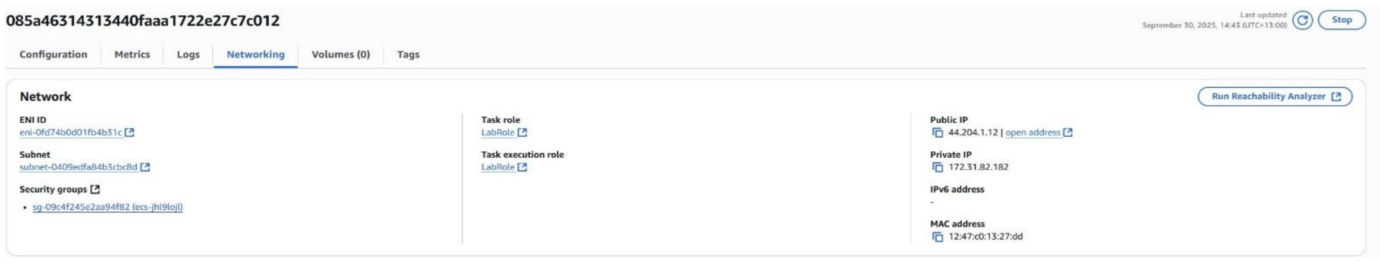
Task execution role
LabRole

Public IP
3.91.45.7 | open address

Private IP
172.31.2.11

IPv6 address
-

MAC address
02:e7:4a:f7:55:41



085a46314313440faaa1722e27c7c012

Last updated
September 30, 2025, 14:43 (UTC+13:00)

Network

ENI ID
eni-0fd74b0d01fb4b31c

Subnet
subnet-0409edfa84b3cbc8d

Security groups
- sg-09c4f245e2aa94f82 (ecs-jhl9lojl)

Task role
LabRole

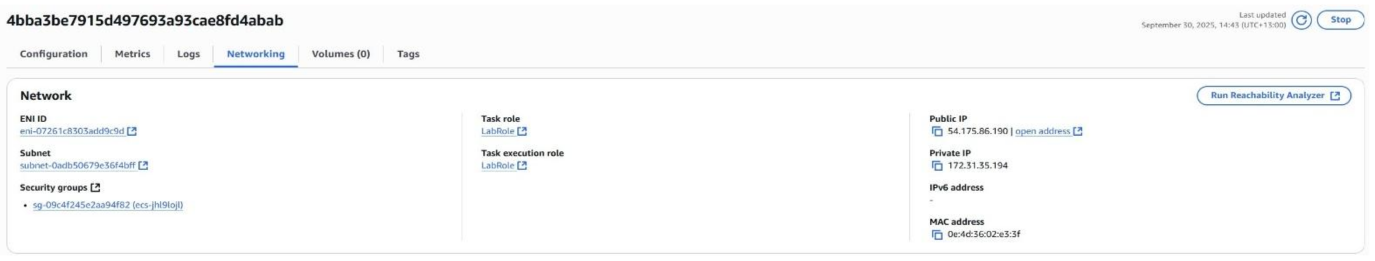
Task execution role
LabRole

Public IP
44.204.1.12 | open address

Private IP
172.31.82.182

IPv6 address
-

MAC address
12:47:c0:13:27:dd



4bba3be7915d497693a93cae8fd4abab

Last updated
September 30, 2025, 14:43 (UTC+13:00)

Network

ENI ID
eni-07261c8303add9c9d

Subnet
subnet-0adb50679e36f4bff

Security groups
- sg-09c4f245e2aa94f82 (ecs-jhl9lojl)

Task role
LabRole

Task execution role
LabRole

Public IP
54.175.86.190 | open address

Private IP
172.31.35.194

IPv6 address
-

MAC address
0e:4d:36:02:e3:3f

Each replica is unique due to it’s assigned private IP address, which is provided by the Elastic Network Interface and is different for each task (172.31.2.11, 172.31.82.182, 172.31.35.194). This allows independent routing and operation within the cluster.

Question 2:

To manage the clients talking to replicas better we could, implement a Load balancer. This will improve the system by:

Traffic Distribution, the load balancer evenly distributes the incoming TCP traffic on port 32000 across all 3 tasks based on a target group. This ensures no single task is overwhelmed.

The load balancer with continuously perform health checks to verify each task availability. If a task fails, the load balancer automatically removes it from rotation and distributes the traffic to the healthy tasks.

Scalability, as more people use the task, the load balancer supports auto scaling by integrating with ECS. If the traffic spikes, ECS can launch addition tasks, and the load balancer can include them in the pool.