

AWS EC2 instance dashboard showing the Tarot Card Server instance after stopping existing services to prepare for Docker installation. The green notification indicates successful instance rebooting to ensure a clean state before containerizing the TarotCard server application.

Installation of Docker. The terminal shows the successful installation of container runtime dependencies. Demonstrating that the EC2 instance is being prepared as a Docker development environment after cleaning up previous services.


```
[ec2-user@ip-172-31-30-45 ~]$ 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
Server Version: 25.0.8
Storage Driver: overlay2
  Backing Filesystem: xfs
  Supports d_type: true
  Using metacopy: false
  Native Overlay Diff: true
  userxattr: false
Logging Driver: json-file
Cgroup Driver: systemd
Cgroup Version: 2
Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local splunk syslog
Swarm: inactive
Runtimes: io.containerd.runc.v2 runc
Default Runtime: runc
Init Binary: docker-init
containerd version: fb4c30d4ede3531652d86197bf3fc9515e5276d9
runc version: 6c52b3fc541fb26fe8c374d5f58112a0a5dbda66
init version: de40ad0
Security Options:
  seccomp
   Profile: builtin
  cgroupns
Kernel Version: 6.1.147-172.266.amzn2023.x86_64
Operating System: Amazon Linux 2023.8.20250818
OSType: linux
Architecture: x86_64
CPUs: 1
Total Memory: 949.4MiB
Name: ip-172-31-30-45.ec2.internal
ID: f3dcc424-4083-4037-9660-1e63113aa382
Docker Root Dir: /var/lib/docker
Debug Mode: false
Experimental: false
Insecure Registries:
  127.0.0.0/8
Live Restore Enabled: false

[ec2-user@ip-172-31-30-45 ~]$
```

Output of docker info command showing Docker daemon is running successfully. This confirms the Docker installation was completed correctly and the service is operational on the cleaned instance.

Creating docker image

```
[ec2-user@ip-172-31-30-45 ~]$ ls
TarotCardDraw.iml  out  run.sh  src
[ec2-user@ip-172-31-30-45 ~]$ cd src
[ec2-user@ip-172-31-30-45 src]$ ls
ClientHandler.class  TarotCardClient.java  TarotCardServer.java
TarotCardClient.class  TarotCardServer.class
[ec2-user@ip-172-31-30-45 src]$ cat > Dockerfile << 'EOF'
> FROM openjdk:8
> COPY *.java /usr/src/TCS/
> WORKDIR /usr/src/TCS
> RUN javac TarotCardServer.java
> EXPOSE 32000
> CMD ["java", "TarotCardServer", "32000"]
> EOF
[ec2-user@ip-172-31-30-45 src]$ 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-172-31-30-45 src]$
```

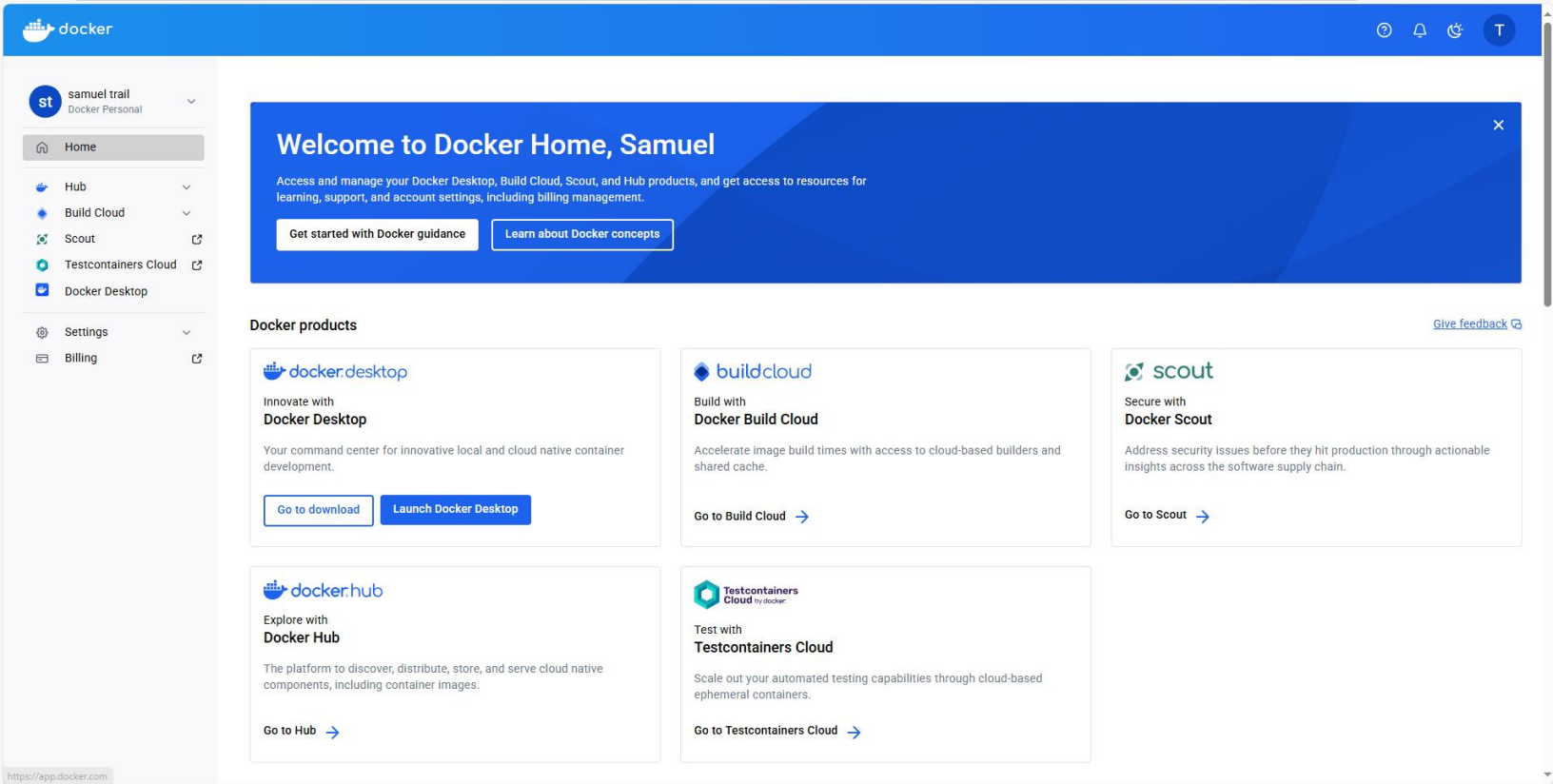
Creation of the Dockerfile using the cat command. The Dockerfile contains instructions to build a container image from OpenJDK 8 base image, copy Java source files, compile the TarotCardServer.java, expose port 32000, and execute the server on startup.

```
[ec2-user@ip-172-31-30-45 src]$ docker images --filter reference=tcs
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
tcs           latest   51fa9aa8b170   8 seconds ago  526MB
[ec2-user@ip-172-31-30-45 src]$
```

Successful execution of docker build -t tcs . command and docker images --filter reference=tcs showing the newly created container image.

[illegible]

Docker hub repo



Docker Hub dashboard showing successful account creation and repo setup.

```
[ec2-user@ip-172-31-30-45 src]$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED      STATUS
PORTS
2335a3e74fdf   tcs       "java TarotCardServe..." 12 minutes ago Up 12 minute
s   0.0.0.0:32000->32000/tcp, :::32000->32000/tcp   infallible_hellman
[ec2-user@ip-172-31-30-45 src]$ docker tag tcs trailsamu/tcs
[ec2-user@ip-172-31-30-45 src]$ docker push trailsamu/tcs
Using default tag: latest
The push refers to repository [docker.io/trailsamu/tcs]
21b608edf65a: Pushed
5f70bf18a086: Pushed
8a6ebb8f03d8: 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: sha256:4a4d52a92b56971ba67de35460e1f8d5b6c95fda1cf83e562146ed11
9525b20c size: 2417
[ec2-user@ip-172-31-30-45 src]$
```

Pushing the container image to Docker Hub using docker tag and docker push commands. The push process shows multiple layers being uploaded, with the final confirmation that the image is now available in the trailsamu/tcs repo.

hub

Explore

My Hub

Search Docker Hub

CtrlK

T

trailsamu

Docker Personal

Repositories

Collaborations

Settings

Default privacy

Notifications

Billing

Usage

Pulls

Storage

Repositories / tcs / General

trailsamu/tcs

Last pushed 2 minutes ago

Add a description

Add a category

General

Tags

Image Management

Collaborators

Webhooks

Settings

Tags

DOCKER SCOUT INACTIVE

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	less than 1 day	2 minutes

See all

Repository overview

INCOMPLETE

An overview describes what your image does and how to run it. It displays in the public view of your repository once you have pushed some content.

Add overview

Using 0 of 1 private repositories.

Docker commands

To push a new tag to this repository:

docker push trailsamu/tcs:tagname

Public view

buildcloud

Build with Docker Build Cloud

Accelerate image build times with access to cloud-based builders and shared cache.

Docker Build Cloud executes builds on optimally-dimensioned cloud infrastructure with dedicated per-organization isolation.

Get faster builds through shared caching across your team, native multi-platform support, and encrypted data transfer - all without managing infrastructure.

Go to Docker Build Cloud

Docker Hub repo page showing the successfully uploaded TCS container image with "latest" tag, confirming the image is publicly accessible for deployment on other systems.

Instance testing

EC2

Dashboard

EC2 Global View

Events

Instances

Instances (6)

Find Instance by attribute or tag (case-sensitive)

All states

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
<input type="checkbox"/>	Tarrot Card Se...	i-0929c67f7d1bb0a0e	Running	t2.micro	Initializing	View alarms +	us-east-1c	ec2-52-90-100-194.co...	52.90.100.194	-
<input type="checkbox"/>	TCS test	i-04066ddfceab9b158	Running	t3.micro	Initializing	View alarms +	us-east-1a	-	44.212.51.185	-

Instance summary for i-04066ddfceab9b158 (TCS test)

Updated 4 minutes ago

Instance ID

i-04066ddfceab9b158

IPv6 address

-

Hostname type

IP name: ip-192-168-0-93.ec2.internal

Answer private resource DNS name

-

Auto-assigned IP address

44.212.51.185 [Public IP]

IAM Role

-

IMDSv2

Required

Operator

-

Public IPv4 address

44.212.51.185 | open address

Instance state

Running

Private IP DNS name (IPv4 only)

ip-192-168-0-93.ec2.internal

Instance type

t3.micro

VPC ID

vpc-0a6a07b36a0e74cab (Tarot VPC)

Subnet ID

subnet-048e6bc29c891cf6 (Sub1)

Instance ARN

arn:aws:ec2:us-east-1:367544546005:i-

Details

Status and alarms

Monitoring

Security

Networking

Storage

Tags

Instance details

AMI ID

ami-08982f1c5b93d976

AMI name

al2023-ami-2023.8.20250915.0-kernel-6.1-x86_64

Stop protection

-

Monitoring

disabled

Allowed image

-

Launch time

-

Connect

Instance state

Actions

Launch instances

aws

Search

[Alt+S]

EC2

Instances

i-04066ddfceab9b158

Dashboard

EC2 Global View

Events

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

Images

AMIs

AMI Catalog

Elastic Block Store

Volumes

Snapshots

Lifecycle Manager

Network & Security

Security Groups

Elastic IPs

Placement Groups

Key Pairs

Network Interfaces

Load Balancing

Terminal

Last login: Wed Sep 24 02:36:43 2025 from 219.89.7.183

[ec2-user@ip-192-168-0-93 ~]\$ 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

Server Version: 25.0.8

Storage Driver: overlay2

Backing Filesystem: xfs

Supports d_type: true

Using metacopy: false

Native Overlay Diff: true

userxattr: false

Logging Driver: json-file

Cgroup Driver: systemd

Cgroup Version: 2

Plugins:

Volumes: local

Network: bridge host ipvlan macvlan null overlay

Log: awslogs fluentd gcplogs jelf journald json-file local splunk syslog

Swarm: inactive

Runtimes: io.containerd.runc.v2 runc

Default Runtime: runc

Init Binary: docker-init

containerd version: 991cc3363c290ff074e069f2b3834c7286ecbe0

runc version: 6c52b3fc541fb26fe8c374d5f58112a0a5dbda66

init version: de40ad0

Security Options:

seccomp

Profile: builtin

cgroupns

Kernel Version: 6.1.150-174.273.amzn2023.x86_64

Operating System: Amazon Linux 2023.8.20250915

OSType: linux

Architecture: x86_64

CPUs: 2

Total Memory: 904.8MiB

Name: ip-192-168-0-93.ec2.internal

ID: 289ac6f8-b426-4580-ba2f-051c0bba3e2f

Docker Root Dir: /var/lib/docker

Debug Mode: false

Experimental: false

Insecure Registries:

127.0.0.0/8

Live Restore Enabled: false

[ec2-user@ip-192-168-0-93 ~]\$

Creation of a new TCS test EC2 instance and installation of Docker on the clean system

```
[ec2-user@ip-192-168-0-93 ~]$ docker pull trailsamu/tcs:latest
latest: Pulling from trailsamu/tcs
001c52e26ad5: Pull complete
d9d4b9b6e964: Pull complete
2068746827ec: Pull complete
9daef329d350: Pull complete
d85151f15b66: Pull complete
52a8c426d30b: Pull complete
8754a66e0050: Pull complete
a90a813f4202: Pull complete
4f4fb700ef54: Pull complete
9fb97ac96f68: Pull complete
Digest: sha256:4a4d52a92b56971ba67de35460e1f8d5b6c95fda1cf83e562146ed119525b20c
Status: Downloaded newer image for trailsamu/tcs:latest
docker.io/trailsamu/tcs:latest
[ec2-user@ip-192-168-0-93 ~]$ docker run -t -i -p 32000:32000 trailsamu/tcs:latest &
[1] 27459
[ec2-user@ip-192-168-0-93 ~]$ docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED        STATUS        PORTS
51d08f80205d   trailsamu/tcs:latest "java TarotCardServe..." 35 seconds ago Up 33 seconds 0.0.0.0:
32000->32000/tcp, :::32000->32000/tcp   youthful_napier
[ec2-user@ip-192-168-0-93 ~]$
```

Pulling the container image from Docker Hub using docker pull trailsamu/tcs:latest and running it on the test instance. This proves the containerized application works.

TarotCardDraw

Version control

TarotCardClient

TarotCardServer.java

TarotCardClient.java

C:\Users\death\.jdk\openjdk-21.0.2\bin\java.exe -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA\bin\idea_rt.jar -Didea.config.path=C:\Users\death\.idea\workspace\TarotCardDraw -Didea.platform.prefix=java -jar C:\Users\death\IdeaProjects\TarotCardDraw\tarot-card-draw-0.0.1-SNAPSHOT-jar-with-dependencies.jar

TAROT CARD READING

Past: Death
Present: The Star
Future: Justice

Server IP: 172.17.0.2

Process finished with exit code 0

AMI Catalog

Elastic Block Store

Volumes

Snapshots

Lifecycle Manager

Network & Security

Security Groups

Elastic IPs

Placement Groups

Key Pairs

Network Interfaces

Load Balancing

IMDSv2

Required

Operator

-

Details

Status and alarms

Monitoring

Security

Networking

Storage

Tags

Instance details

AMI ID

ami-08982f1c5bf93d976

AMI name

al2023-ami-2023.8.20250915.0-kernel-6.1-x86_64

Stop protection

Monitoring

disabled

Allowed image

-

Launch time

-

Public IPv4 address

44.212.51.185 | open address

Private IPv4 addresses

192.168.0.93

Public DNS

Instance state

Running

Private IP DNS name (IPv4 only)

ip-192-168-0-93.ec2.internal

Instance type

t3.micro

VPC ID

vpc-0a6a07b36a0e74cab (Tarot VPC)

Subnet ID

subnet-048e6bc29c891cfc6 (Sub1)

Instance ARN

arn:aws:ec2:us-east-1:367544546005:instance/i-0a6a07b36a0e74cab

Run/Debug Configurations

Application

TarotCardServer

TarotCardClient

TarotCardClient_2

Name

TarotCardClient

Run on

Local machine

Run

java 21 SDK of 'TarotCardDraw' module

TarotCardClient

44.212.51.185 32000

Working directory

C:\Users\death\IdeaProjects\TarotCardDraw

Environment variables

Environment variables or .env files

Open run/debug tool window when started

Allow multiple instances

Run

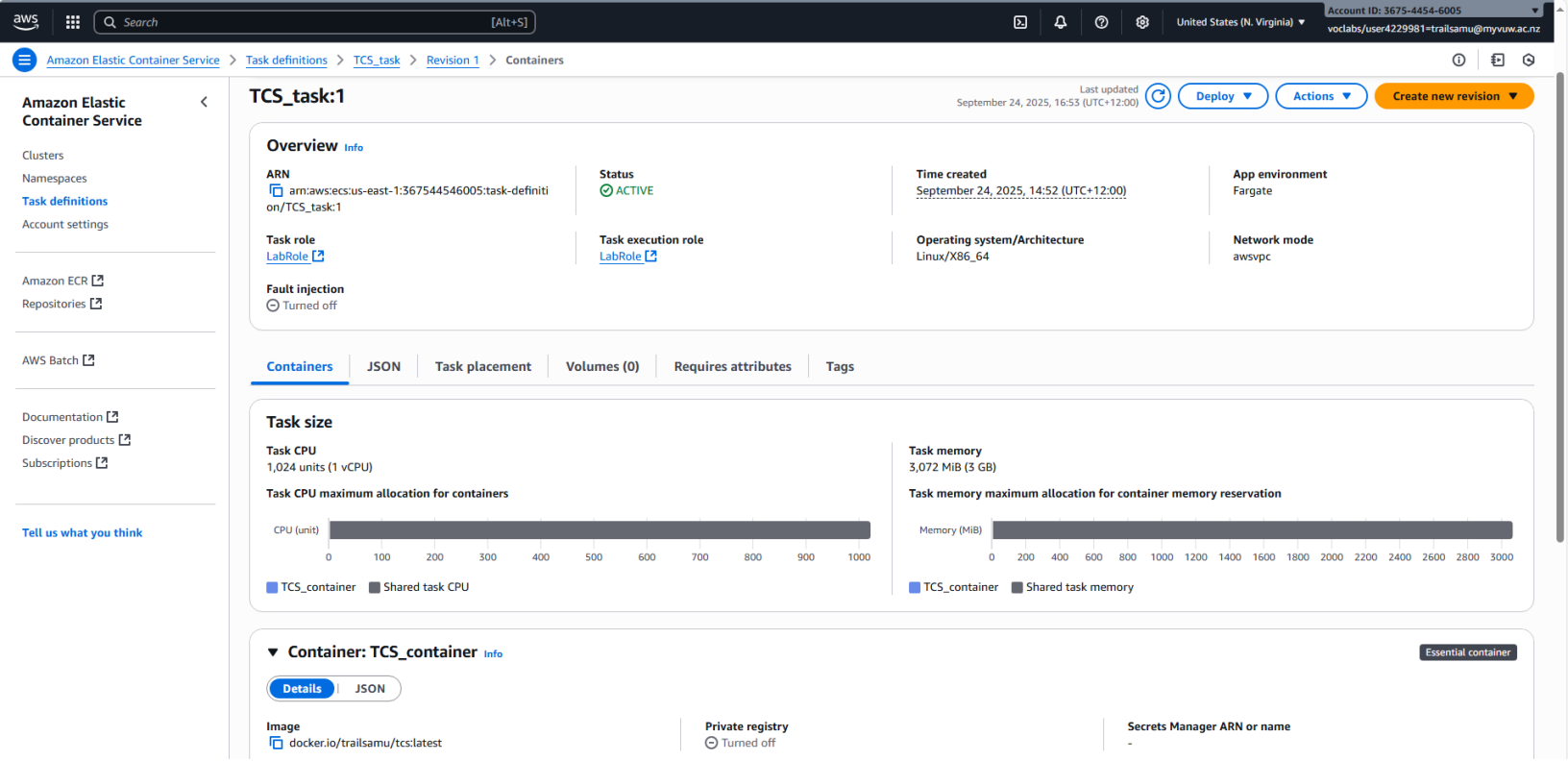
OK

Cancel

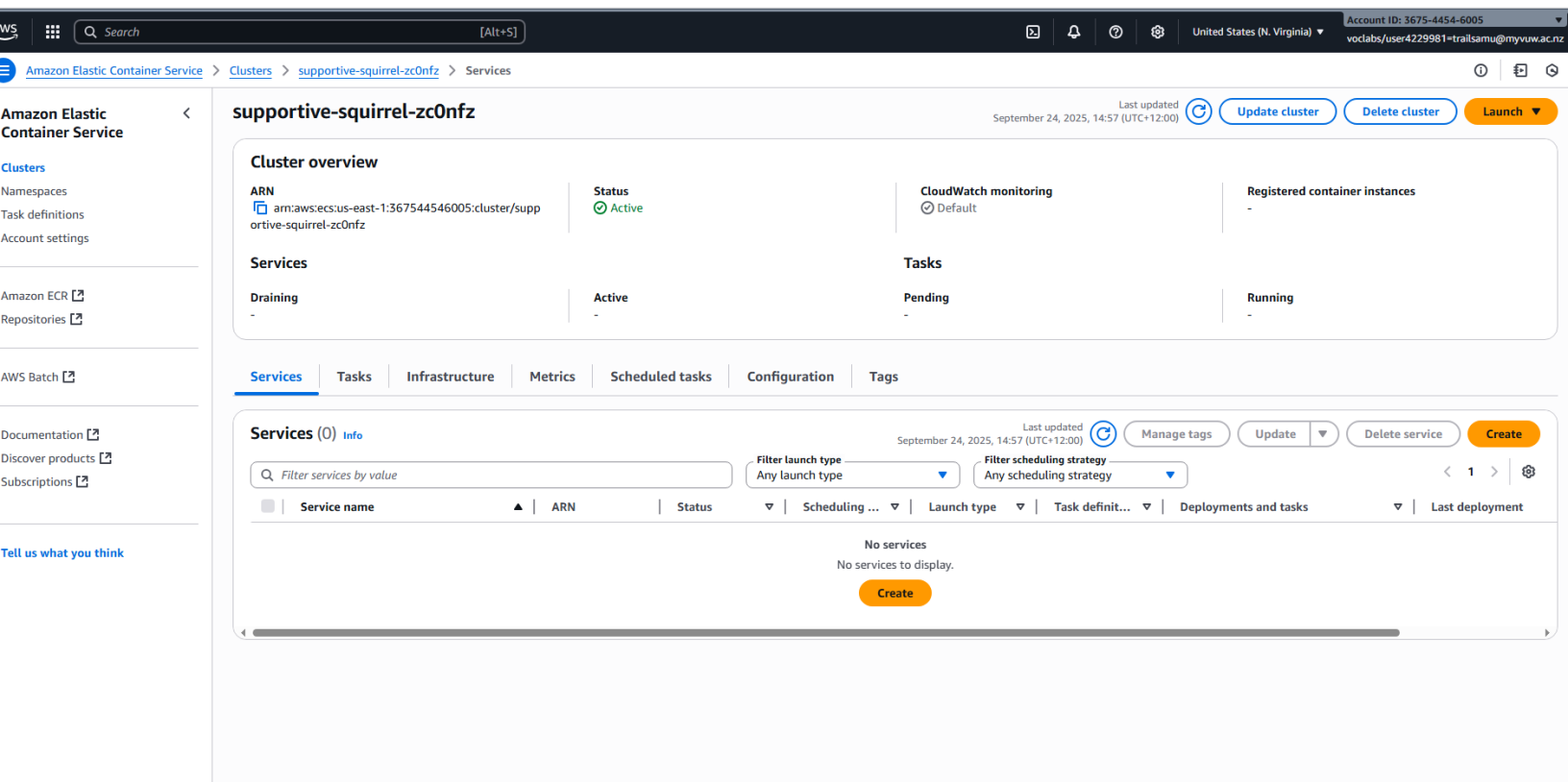
Apply

Client successfully connecting to the containerized server on the test instance, receiving a different tarot card reading

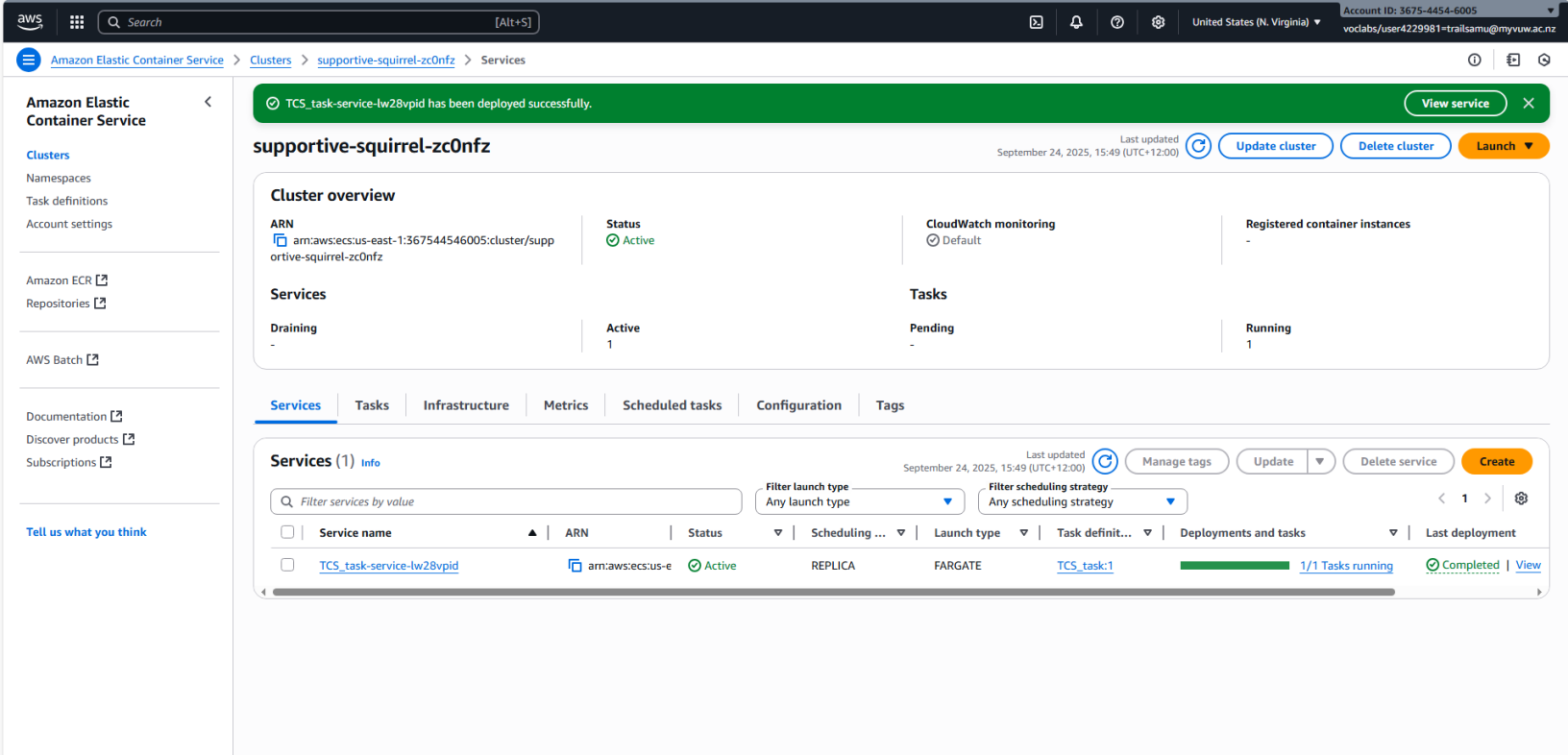
AWS/Fargate stuff



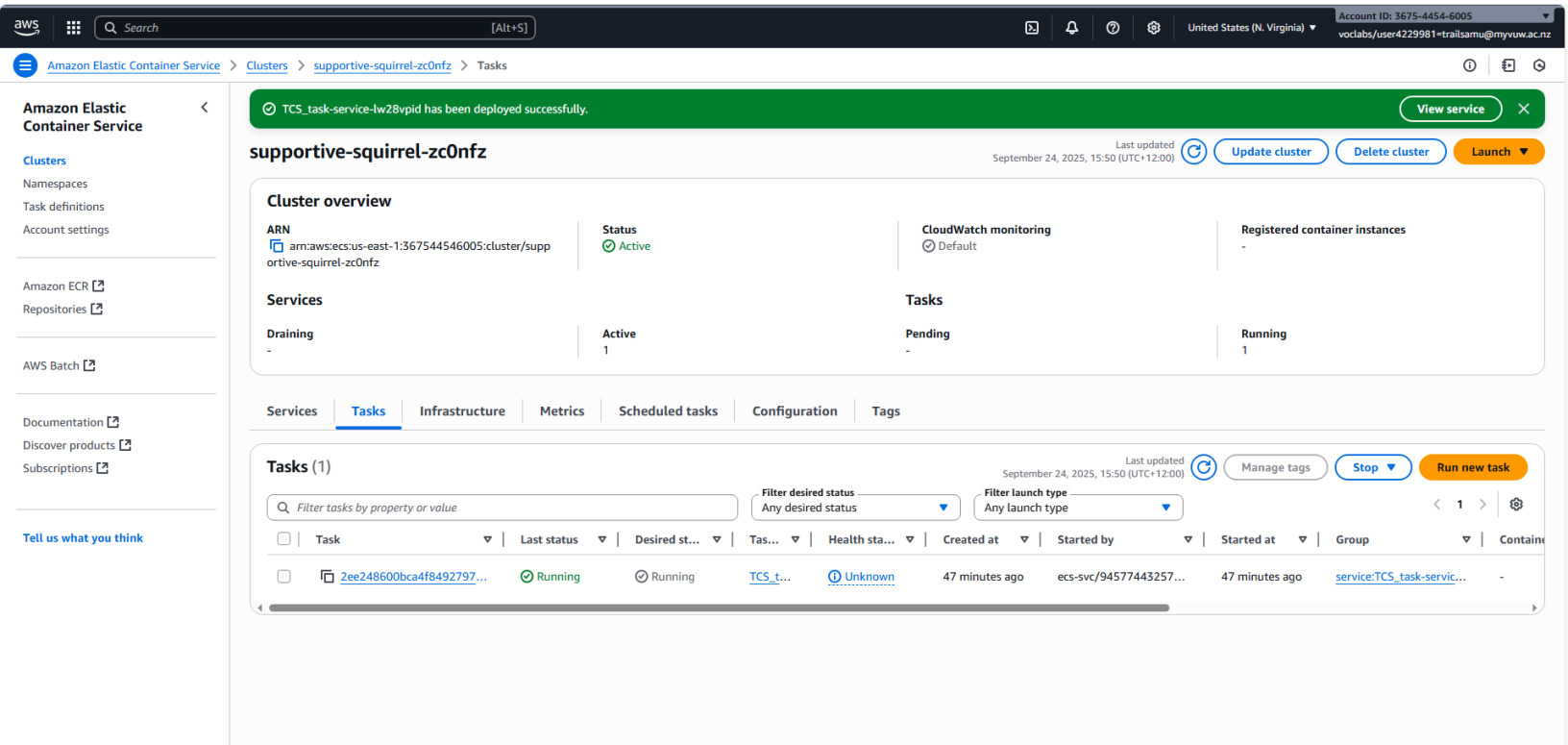
ECS Task Definition creation showing the configuration for containerized deployment using AWS Fargate. The task definition specifies the Docker image location (trailsamu/tcs:latest)



ECS Cluster creation with AWS Fargate serverless infrastructure.



ECS Service deployment showing successful creation of the containerized service. The service is configured to maintain one running task using the Fargate launch type for serverless execution



ECS Tasks overview showing one task successfully running in the cluster.

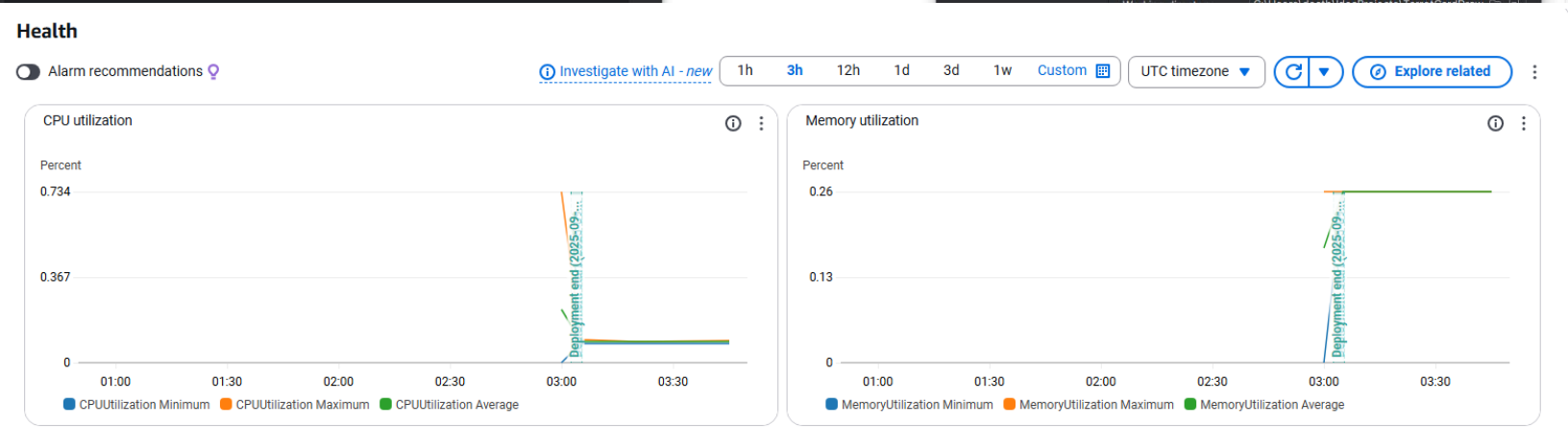
IDE screenshot showing TarotCardClient.java and TarotCardServer.java files. The output window displays the tarot reading results: Past: Wheel of Fortune, Present: The Hierophant, Future: The Chariot. The server IP is 172.31.76.29. The process finished with exit code 0.

Configuration details for the application:

- ENI ID: [eni-0bf1b744ec7f2c1a4](#)
- Network mode: awsvpc
- Subnet: [subnet-025ec8627489f75a7](#)
- Public IP: 44.200.168.239 (open address)
- Private IP: 172.31.76.29
- MAC address: 16:ff:ed:f7:b3:c1

Run/Debug Configurations window:

- Name: TarotCardClient
- Run on: Local machine
- Run: java 21 SDK of 'TarotCardDraw' module
- Class: TarotCardClient
- Arguments: 44.200.168.239 32000



Replicas and Notes

Each replica gets its own unique IP address (both public and private). Even though they're running identical container images with the same code, AWS assigns each task instance its own network interface and IP addresses.

Amazon Elastic Container Service

Clusters

Namespaces

Task definitions

Account settings

Amazon ECR

Repositories

AWS Batch

Documentation

Discover products

Subscriptions

Tell us what you think

TCS_task-service-lw28vpid has been deployed successfully.

TCS_3 has been deployed successfully.

supportive-squirrel-zc0nfz

Last updated
September 24, 2025, 16:00 (UTC+12:00)

Update cluster

Delete cluster

Launch

Cluster overview

ARN
arn:aws:ecs:us-east-1:367544546005:cluster/supportive-squirrel-zc0nfz

Status
Active

CloudWatch monitoring
Default

Registered container instances
-

Services

Tasks

Draining
-

Active
2

Pending
-

Running
4

Services (2)

Info

Last updated
September 24, 2025, 16:00 (UTC+12:00)

Manage tags

Update

Delete service

Create

Filter services by value

Filter launch type
Any launch type

Filter scheduling strategy
Any scheduling strategy

Service name

ARN

Status

Scheduling ...

Launch type

Task definit...

Deployments and tasks

Last deployment

TCS_3

arn:aws:ecs:us-e

Active

REPLICA

FARGATE

TCS_task:1

3/3 Tasks running

Completed

TCS_task-service-lw28vpid

arn:aws:ecs:us-e

Active

REPLICA

FARGATE

TCS_task:1

1/1 Tasks running

Completed

Amazon Elastic Container Service

Clusters

Namespaces

Task definitions

Account settings

Amazon ECR

Repositories

AWS Batch

Documentation

Discover products

Subscriptions

Service overview

Info

Status
Active

Tasks (3 Desired)
0 Pending | 3 Running

Task definition: revision
TCS_task:1

Deployment status
Success

Health and metrics

Tasks

Logs

Deployments

Events

Configuration and networking

Service auto scaling

Tags

Tasks (1/3)

Filter tasks by property or value

Filter desired status
Any desired status

Filter launch type
Any launch type

Task

Last status

Desired st...

Task ...

Health sta...

Created at

Started by

Started at

Container instan...

Launch

99c75fe72ef54d31a6de6...

Running

Running

TCS_tas...

Unknown

4 minutes ago

ecs-svc/04687667015...

3 minutes ago

-

FARGAT

9b0229b9a36c4a95b619...

Running

Running

TCS_tas...

Unknown

4 minutes ago

ecs-svc/04687667015...

4 minutes ago

-

FARGAT

b35415d973c748f6b106...

Running

Running

TCS_tas...

Unknown

4 minutes ago

ecs-svc/04687667015...

3 minutes ago

-

FARGAT

Amazon Elastic Container Service

Clusters

Namespaces

Task definitions

Account settings

Amazon ECR

Repositories

AWS Batch

Documentation

Discover products

Subscriptions

Configuration

Operating system/Architecture
Linux/X86_64

CPU | Memory
1 vCPU | 3 GB

Platform version
1.4.0

Task execution role
LabRole

Task role
LabRole

Fault injection
Turned off

ECS Exec
Turned off

Capacity provider
-

Launch type
Fargate

Task definition: revision
TCS_task:1

Task group
service:TCS_3

ENI ID
eni-0a209fc2409159635

Network mode
awsvpc

Subnet
subnet-07175916b7aa1253b

Public IP
44.193.84.0

Private IP
172.31.14.126

MAC address
02:f4:9a:c8:ab:7f

Task scale-in protection

Configure task scale-in protection

Amazon Elastic Container Service

Configuration

Operating system/Architecture: Linux/X86_64

CPU | Memory: 1 vCPU | 3 GB

Platform version: 1.4.0

Task execution role: [LabRole](#)

Task role: [LabRole](#)

Fault injection: ☐ Turned off

ECS Exec: [Info](#) ☐ Turned off

Task scale-in protection: [Configure task scale-in protection](#)

Capacity provider: -

Launch type: Fargate

Task definition: revision: [TCS_task:1](#)

Task group: service:TCS_3 | [View service](#)

ENI ID: [eni-06b1f7b86e999b930](#)

Network mode: awsvpc

Subnet: [subnet-0649c49530d7463e4](#)

Public IP: [100.26.199.83](#) | [open address](#)

Private IP: [172.31.44.143](#)

MAC address: [0e:b5:95:e8:30:2b](#)

Container details for TCS_container

Amazon Elastic Container Service

Configuration

Operating system/Architecture: Linux/X86_64

CPU | Memory: 1 vCPU | 3 GB

Platform version: 1.4.0

Task execution role: [LabRole](#)

Task role: [LabRole](#)

Fault injection: ☐ Turned off

ECS Exec: [Info](#) ☐ Turned off

Task scale-in protection: [Configure task scale-in protection](#)

Capacity provider: -

Launch type: Fargate

Task definition: revision: [TCS_task:1](#)

Task group: service:TCS_3 | [View service](#)

ENI ID: [eni-0dd83ddad011a7ea6](#)

Network mode: awsvpc

Subnet: [subnet-00e31af0f7ee1d843](#)

Public IP: [54.236.33.46](#) | [open address](#)

Private IP: [172.31.50.169](#)

MAC address: [06:90:66:d4:43:53](#)

Container details for TCS_container

Load Balancer Solution: To provide a single endpoint for clients, a load balancer is implemented. Instead of clients connecting directly to individual replica IPs, they connect to a single load balancer endpoint that distributes requests across all healthy replicas automatically. This provides better availability, automatic failover, and distributes load evenly