

Docker installation and setup

The screenshot shows the AWS EC2 Instances page. A green notification bar at the top indicates "Successfully initiated rebooting of i-0929c67f7d1bb0a0e". The main area displays the "Instance summary for i-0929c67f7d1bb0a0e (Tarot Card Server)". The instance is currently stopped. The "Details" tab is selected, showing a terminal session window. The terminal output shows the user has stopped several services (tarot, tarot.service) and disabled their targets. The user then runs the command "sudo service docker start", which successfully starts the Docker service.

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.

```
ec2-user@ip-172-31-30-45:~
```

```
Installed:
container-selinux-3:2.233.0-1.amzn2023.noarch
containerd-2.0.5-1.amzn2023.0.2.x86_64
docker-25.0.8-1.amzn2023.0.5.x86_64
iptables-libs-1.8.8-3.amzn2023.0.2.x86_64
iptables-nft-1.8.8-3.amzn2023.0.2.x86_64
libcgroup-3.0-1.amzn2023.0.1.x86_64
libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
libnftnl-1.2.2-2.amzn2023.0.2.x86_64
pigz-2.5-1.amzn2023.0.3.x86_64
runc-1.2.6-1.amzn2023.0.1.x86_64

Complete!
[ec2-user@ip-172-31-30-45 ~]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[ec2-user@ip-172-31-30-45 ~]$
```

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
    cgroups
  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
TarrotCardDraw.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.

```
[ec2-user@ip-172-31-30-45 src]$ docker run -t -i -p 32000:32000 tcs &
[1] 29187
[ec2-user@ip-172-31-30-45 src]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
2335a3e74fdf      tcs                "java TarotCardServe..."   24 seconds ago    Up 23 seconds     0.0.0.0:32000->32000/tcp, :::32000->32000/tcp
[ec2-user@ip-172-31-30-45 src]$
```

The screenshot shows a developer's workstation with several windows open:

- IntelliJ IDEA:** A Java application named "TarotCardClient" is running. The terminal output shows the application has started and is performing a "TAROT CARD READING". It lists the past, present, and future tarot cards: "The Sun", "The Chariot", and "The Hierophant". The server IP is listed as 172.17.0.2.
- CloudWatch Logs:** A log stream titled "1bb0a0e" is displayed, showing the same log entries as the IntelliJ terminal.
- AWS Management Console - EC2:** An instance named "infallible_hellman" is shown. Key details include:
 - Public IPv4 address: 18.234.111.42
 - Private IP4 addresses: 172.31.30.45
 - Instance state: Running
 - Private IP DNS name (IPv4 only): ip-172-31-30-45.ec2.internal
 - Instance type: t2.micro
 - VPC ID: vpc-034b78dddee713fc43
 - Subnet ID: subnet-0046b587c5e014ddb
 - Instance ARN: arn:aws:ec2:us-east-1:367544546005:instance/i-0929
- Run/Debug Configurations:** A dialog box for the "TarotCardClient" configuration. It shows the "Application" section with "TarotCardServer" and "TarotCardClient" selected. The "Run" section is set to "java 21 SDK of 'TarotCardDraw' module" and "Local machine".

Testing the containerized server locally using `docker run -t -i -p 32000:32000 tcs &` and verifying it's running with `docker ps`. The container shows proper port mapping (0.0.0.0:32000->32000/tcp) and successful client connection from local machine, demonstrating the containerization works correctly.

Docker hub repo

Welcome to Docker Home, Samuel

Access and manage your Docker Desktop, Build Cloud, Scout, and Hub products, and get access to resources for learning, support, and account settings, including billing management.

Get started with Docker guidance | Learn about Docker concepts

Docker products

docker.desktop

Innovate with Docker Desktop

Your command center for innovative local and cloud native container development.

[Go to download](#) | [Launch Docker Desktop](#)

buildcloud

Build with Docker Build Cloud

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

[Go to Build Cloud](#)

scout

Secure with Docker Scout

Address security issues before they hit production through actionable insights across the software supply chain.

[Go to Scout](#)

docker.hub

Explore with Docker Hub

The platform to discover, distribute, store, and serve cloud native components, including container images.

[Go to Hub](#)

Testcontainers Cloud by docker

Test with Testcontainers Cloud

Scale out your automated testing capabilities through cloud-based ephemeral containers.

[Go to Testcontainers Cloud](#)

Give feedback

<https://app.docker.com>

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.

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

aws Search [Alt+S] United States (N. Virginia) Account ID: 3675-4454-6005

EC2 Instances

Instances (6) Info

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
<input type="checkbox"/>	Tarot 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	-

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:
    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: 991cc3363c290ff074e0609f2b3034c7286ecbe0
  runt version: 6c52b3fc541fb26fe8c374df5f8112a0a5dbda66
  init version: de40ad0
  Security Options:
    seccomp
    Profile: builtin
    cgroups
  Kernel Version: 6.1.150-173.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: 289ec6f8-b426-4589-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 ~]$ 

```

EC2 Instances

Instance summary for i-04066ddfceab9b158 (TCS test) Info

Updated 4 minutes ago

Instance ID	i-04066ddfceab9b158	Public IPv4 address	44.212.51.185 open address
IPv6 address	-	Instance state	Running
Hostname type	IP name: ip-192-168-0-93.ec2.internal	Private IP DNS name (IPv4 only)	ip-192-168-0-93.ec2.internal
Answer private resource DNS name	-	Instance type	t3.micro
Auto-assigned IP address	44.212.51.185 [Public IP]	VPC ID	vpc-0a6a07b36a0e74cab (Tarot VPC)
IAM Role	-	Subnet ID	subnet-048e6bc29c891fcf6 (Sub1)
IMDSv2	Required	Instance ARN	arn:aws:ec2:us-east-1:367544546005:
Operator	-		

Details Status and alarms Monitoring Security Networking Storage Tags

Instance details Info

AMI ID	ami-08982f1c5bf93d976	Monitoring	disabled
AMI name	al2023-ami-2023.8.20250915.0-kernel-6.1-x86_64	Allowed image	-
Stop protection		Launch time	-

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 NAMES 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.

TC TarotCardDraw Version control TarotCardClient Launch AWS Account Instance details nwenv assign 3 context.docx samuel trail | Do traitsam/tcs | +

Project TarotCardClient TarotCardServer.java TarotCardClient.java

```
C:\Users\death\.jdks\openjdk-21.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.3.1\lib\idea_rt.jar" -Dfile.encoding=UTF-8 -jar C:\Users\death\IdeaProjects\TarotCardDraw\tarotcarddraw.jar
```

=====
TAROT CARD READING
=====

Past: Death
Present: The Star
Future: Justice

Server IP: 172.17.0.2
=====

Process finished with exit code 0

tarotCardDraw > src > TarotCardClient 5:1 CRLF UTF-8 4 spaces

IMDSv2
Required
Operator
-

Details Status and alarms Monitoring Security Networking Storage Tags

▼ Instance details Info
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

Run/Debug Configurations

Name: TarotCardClient Store as project file
Run on: Local machine Manage targets...
Run configurations may be executed locally or on a target: for example in a Docker Container or on a remote host using SSH.

Run Modify options Alt+M
java 21 SDK of 'TarotCardDraw' module
TarotCardClient
44.212.51.185 32000
Press Alt for field hints
Working directory: C:\Users\death\deapProjects\TarotCardDraw
Environment variables: Environment variables or .env files Separate variables with semicolon: VAR=value; VAR1=value
Open run/debug tool window when started
Allow multiple instances

Run OK Cancel Apply

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

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

AWS/Fargate stuff

aws Search [Alt+S] United States (N. Virginia) Account ID: 3675-4454-6005 vclabs/user4229981@trailsamu@myvuw.ac.nz

Amazon Elastic Container Service > Task definitions > TCS_task > Revision 1 > Containers

TCS_task:1

Last updated September 24, 2025, 16:53 (UTC+12:00) Deploy Actions Create new revision

ARN arn:aws:ecs:us-east-1:367544546005:task-definition/TCS_task:1	Status ACTIVE	Time created September 24, 2025, 14:52 (UTC+12:00)	App environment Fargate
Task role LabRole	Task execution role LabRole	Operating system/Architecture Linux/X86_64	Network mode awsvpc
Fault injection Turned off			

Containers JSON Task placement Volumes (0) Requires attributes Tags

Task size

Task CPU
1,024 units (1 vCPU)

Task CPU maximum allocation for containers

CPU (unit) 0 100 200 300 400 500 600 700 800 900 1000
TCS_container Shared task CPU

Task memory
3,072 MiB (3 GB)

Task memory maximum allocation for container memory reservation

Memory (MiB) 0 200 400 600 800 1000 1200 1400 1600 1800 2000 2200 2400 2600 2800 3000
TCS_container Shared task memory

Container: TCS_container Info

Essential container

Details JSON

Image
docker.io/trailsamu/tcs:latest

Private registry
Turned off

Secrets Manager ARN or name

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

aws Search [Alt+S] United States (N. Virginia) Account ID: 3675-4454-6005 vclabs/user4229981@trailsamu@myvuw.ac.nz

Amazon Elastic Container Service > Clusters > supportive-squirrel-zc0nfz > Services

supportive-squirrel-zc0nfz

Last updated September 24, 2025, 14:57 (UTC+12:00) Update cluster Delete cluster Launch

ARN arn:aws:ecs:us-east-1:367544546005:cluster/supportive-squirrel-zc0nfz	Status Active	CloudWatch monitoring Default	Registered container instances -
Services		Tasks	
Draining	Active	Pending	Running

Services Tasks Infrastructure Metrics Scheduled tasks Configuration Tags

Services (0) Info

Last updated September 24, 2025, 14:57 (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
No services	No services to display.						

Create

ECS Cluster creation with AWS Fargate serverless infrastructure.

Screenshot of the AWS Elastic Container Service (ECS) Cluster Overview page for the cluster "supportive-squirrel-zc0nfz".

The cluster is active and has one task running. The service "TCS_task-service-lw28vpid" is listed with one task running.

Cluster Overview:

- ARN: arn:aws:ecs:us-east-1:367544546005:cluster/supportive-squirrel-zc0nfz
- Status: Active
- CloudWatch monitoring: Default
- Registered container instances: -

Services:

- Draining: -
- Active: 1

Tasks:

- Pending: -
- Running: 1

Services Tab:

- Services (1) Info
- Last updated: September 24, 2025, 15:49 (UTC+12:00)
- Filter services by value: TCS_task-service-lw28vpid
- Filter launch type: Any launch type
- Filter scheduling strategy: Any scheduling strategy
- Manage tags, Update, Delete service, Create buttons

Service name	ARN	Status	Launch type	Task definition	Deployments and tasks	Last deployment
TCS_task-service-lw28vpid	arn:aws:ecs:us-e	Active	REPLICA	FARGATE	TCS_task:1	1/1 Tasks running Completed View

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

Screenshot of the AWS Elastic Container Service (ECS) Cluster Overview page for the cluster "supportive-squirrel-zc0nfz".

The cluster is active and has one task running. The service "TCS_task-service-lw28vpid" is listed with one task running.

Cluster Overview:

- ARN: arn:aws:ecs:us-east-1:367544546005:cluster/supportive-squirrel-zc0nfz
- Status: Active
- CloudWatch monitoring: Default
- Registered container instances: -

Services:

- Draining: -
- Active: 1

Tasks:

- Pending: -
- Running: 1

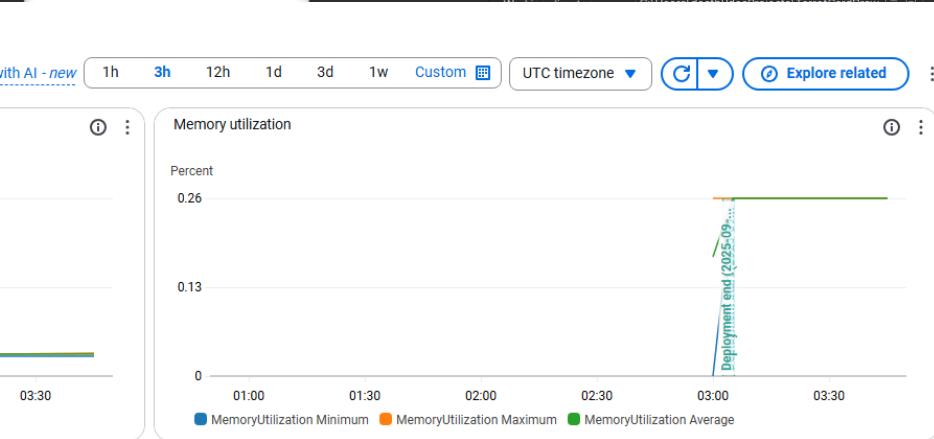
Tasks Tab:

- Tasks (1)
- Last updated: September 24, 2025, 15:50 (UTC+12:00)
- Filter tasks by property or value: 2ee248600bca4f8492797...
- Filter desired status: Any desired status
- Filter launch type: Any launch type
- Stop, Run new task buttons

Task	Last status	Desired st...	Tas...	Health sta...	Created at	Started by	Started at	Group	Container
2ee248600bca4f8492797...	Running	Running	TCS_t...	Unknown	47 minutes ago	ecs-svc/94577443257...	47 minutes ago	service:TCS_task-servic...	-

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

The screenshot shows a Java IDE interface with multiple tabs open. On the left, the project structure for 'TarotCardDraw' is visible, containing 'TarotCardClient', 'TarotCardServer.java', '.gitignore', and 'TarotCardDraw.iml'. The main editor window displays 'TarotCardClient.java' with code related to printing tarot card readings. A terminal window below shows the output of the application, which includes the text 'TAROT CARD READING' and three cards: 'Past: Wheel of Fortune', 'Present: The Hierophant', and 'Future: The Chariot'. The server IP is listed as 172.31.76.29. The process finished with exit code 0.



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 > supportive-squirrel-zc0nfz > Services

supportive-squirrel-zc0nfz

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

Cluster overview

ARN arn:aws:ecs:us-east-1:367544546005:cluster/supportive-squirrel-zc0nfz	Status Active	CloudWatch monitoring Default	Registered container instances -
Services Draining -	Active 2	Tasks Pending -	Running 4

Services | Tasks | Infrastructure | Metrics | Scheduled tasks | Configuration | Tags

Services (2) Info

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 View
TCS_task-service-lw28vpid	arn:aws:ecs:us-e...	Active	REPLICA	FARGATE	TCS_task:1	1/1 Tasks running	Completed View

aws | Search [Alt+S]

Amazon Elastic Container Service > Clusters > supportive-squirrel-zc0nfz > Services > TCS_3 > Tasks

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)

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

aws | Search [Alt+S]

Amazon Elastic Container Service > Clusters > supportive-squirrel-zc0nfz > Services > TCS_3 > Tasks > 99c75fe72ef54d31a6de6c726fc7cdfd > Configuration

Account ID: 56/5-4434-6005 | volelabs/user4229981=trailsamu@myuw.ac.nz

Configuration

Operating system/Architecture Linux/X86_64	Capacity provider -	ENI ID eni-0a209fc2409159635	Public IP 44.193.84.0 open address
CPU / Memory 1 vCPU 3 GB	Launch type Fargate	Network mode awsvpc	Private IP 172.31.14.126
Platform version 1.4.0	Task definition: revision TCS_task:1	Subnet subnet-07175916b7aa1253b	MAC address 02:f4:9a:c8:ab:7f
Task execution role LabRole	Task group service:TCS_3 View service	Configure task scale-in protection	
Task role LabRole			
Fault injection Turned off			
ECS Exec Info Turned off			
Task scale-in protection			

AWS Search [Alt+S] United States (N. Virginia) Account ID: 3675-4454-6005 voclabs/user4229981=trailsamu@myvuw.ac.nz

Amazon Elastic Container Service > Clusters > supportive-squirrel-zcOnfz > Services > TCS_3 > Tasks > 9b0229b9a36c4a95b6194f99d8eca8c > Configuration

Configuration

Operating system/Architecture Linux/X86_64	Capacity provider -	ENI ID eni-06b1ff7b86e999b930	Public IP 100.26.199.83 open address
CPU Memory 1 vCPU 3 GB	Launch type Fargate	Network mode awsvpc	Private IP 172.31.44.143
Platform version 1.4.0	Task definition: revision TCS_task1	Subnet subnet-0649c49530d7463e4	MAC address 0:e:b5:95:e8:30:2b
Task execution role LabRole	Task group service:TCS_3 View service		
Task role LabRole			
Fault injection <input checked="" type="radio"/> Turned off			
ECS Exec Info <input checked="" type="radio"/> Turned off			
Task scale-in protection			
Configure task scale-in protection			
Container details for TCS_container			

Tell us what you think

AWS Search [Alt+S] United States (N. Virginia) Account ID: 3675-4454-6005 voclabs/user4229981=trailsamu@myvuw.ac.nz

Amazon Elastic Container Service > Clusters > supportive-squirrel-zcOnfz > Services > TCS_3 > Tasks > b35415d973c748f6b106c0c886909624 > Configuration

Configuration

Operating system/Architecture Linux/X86_64	Capacity provider -	ENI ID eni-0dd83ddad011a7ea6	Public IP 54.236.33.46 open address
CPU Memory 1 vCPU 3 GB	Launch type Fargate	Network mode awsvpc	Private IP 172.31.50.169
Platform version 1.4.0	Task definition: revision TCS_task1	Subnet subnet-00e31af0f7ee1d843	MAC address 06:90:66:d4:43:53
Task execution role LabRole	Task group service:TCS_3 View service		
Task role LabRole			
Fault injection <input checked="" type="radio"/> Turned off			
ECS Exec Info <input checked="" type="radio"/> Turned off			
Task scale-in protection			
Configure task scale-in protection			
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.