

# Part 1

## Screenshot 1:

Screenshot demonstrates testing my Tarot Card application using IntelliJ's integrated console environment. Both server and client programs were executed through IntelliJ's run configurations with command line arguments (localhost 32000), showing successful socket communication and proper output formatting within the IDE's console panels.

The screenshot shows the IntelliJ IDEA interface with two terminal panes. The left pane, titled 'TarotCardServer', displays the command used to start the server: `C:\Users\death\.jdks\openjdk-21.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.1\lib\jpda.jar" -jar TarotCardServer.jar`. It also shows the message: "Tarot Card Server started on port 32000". The right pane, titled 'TarotCardClient', displays the command used to start the client: `C:\Users\death\.jdks\openjdk-21.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.1\lib\jpda.jar" -jar TarotCardClient.jar`. It shows the output of a tarot card reading: "TAROT CARD READING", followed by "Past: The Fool", "Present: Temperance", and "Future: The Moon". The process finished with exit code 0.

## Screenshot 2:

Screenshot shows the transition from local testing to AWS cloud deployment. The left panel displays the server successfully running on AWS EC2 instance (ip-172-31-30-45), while the panel below shows IntelliJ's run configuration being updated to connect the client to the AWS server's public IP address (18.212.99.251:32000) instead of localhost. The right panel shows the successful connection via IntelliJ's console, demonstrating successful cloud deployment.

```

Preparing : 1/1
Installing : java-1.8.0-amazon-corretto-devel-1:1.8.0_462.b08-1.amzn2023.x86_64
1/1
Running scriptlet: java-1.8.0-amazon-corretto-devel-1:1.8.0_462.b08-1.amzn2023.x86_64
Verifying : java-1.8.0-amazon-corretto-devel-1:1.8.0_462.b08-1.amzn2023.x86_64
1/1

installed:
java-1.8.0-amazon-corretto-devel-1:1.8.0_462.b08-1.amzn2023.x86_64

complete!
ec2-user@ip-172-31-30-45 ~]$ javac TarotCardServer.java
ec2-user@ip-172-31-30-45 ~]$ javac TarotCardClient.java
ec2-user@ip-172-31-30-45 ~]$ java TarotCardServer 32000
Tarot Card Server started on port 32000
netstat -an | grep :32000
C[ec2-user@ip-172-31-30-45 ~]$ ss -tuln | grep 32000
ec2-user@ip-172-31-30-45 ~]$ java TarotCardServer 32000
Tarot Card Server started on port 32000

Run/Debug Configurations
+ Application
  TarotCardServer
  TarotCardClient (selected)

Name: TarotCardClient
Run on: Local machine
Store as project file

Build and run
java 21 SDK of 'Tarot' TarotCardClient
18.212.99.251 32000
Working directory: C:\Users\death\IdeaProjects\TarotCardDraw
Environment variables: Environment variables or .env files
Open run/debug tool window when started

Instance type: t2.micro
VPC ID: vpc-034b78dde71
Subnet ID: subnet-0046b587c5
Instance ARN: arn:aws:ec2:us-east-1:123456789012:instance/i-01234567890123456
Networking
Monitoring disabled
Allowed image:
Launch time

Project TarotCardDraw Version control TarotCardClient Run TarotCardClient
TarotCardDraw C:\Users\death\IdeaProjects\TarotCardDraw
  .idea
  out
  src
    TarotCardClient
    TarotCardServer.java
    .gitignore
    TarotCardDraw.iml
External Libraries
Run TarotCardClient
C:\Users\death\.jdks\openjdk-21.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022.3.1\lib\idea_rt.jar=5119:localhost:8000" -Dfile.encoding=UTF-8 -jar C:\Users\death\IdeaProjects\TarotCardDraw\tarotcardclient.jar
=====
TAROT CARD READING
=====
Past: The Emperor
Present: The Sun
Future: Justice
=====
Process finished with exit code 0

```

## Screenshot 3:

Screenshot shows the AWS EC2 terminal using systemctl to check the status of the tarot service. The output displays the service as loaded and enabled but currently inactive (dead), indicating the tarot server application has been configured as a system service but is not currently running on the EC2 instance as it needs a reboot.

```
[ec2-user@ip-172-31-30-45 ~]$ sudo systemctl status tarot
● tarot.service - Tarot service
  Loaded: loaded (/etc/systemd/system/tarot.service; enabled; preset: disabled)
  Active: inactive (dead)
[ec2-user@ip-172-31-30-45 ~]$
```

## Screenshot 4:

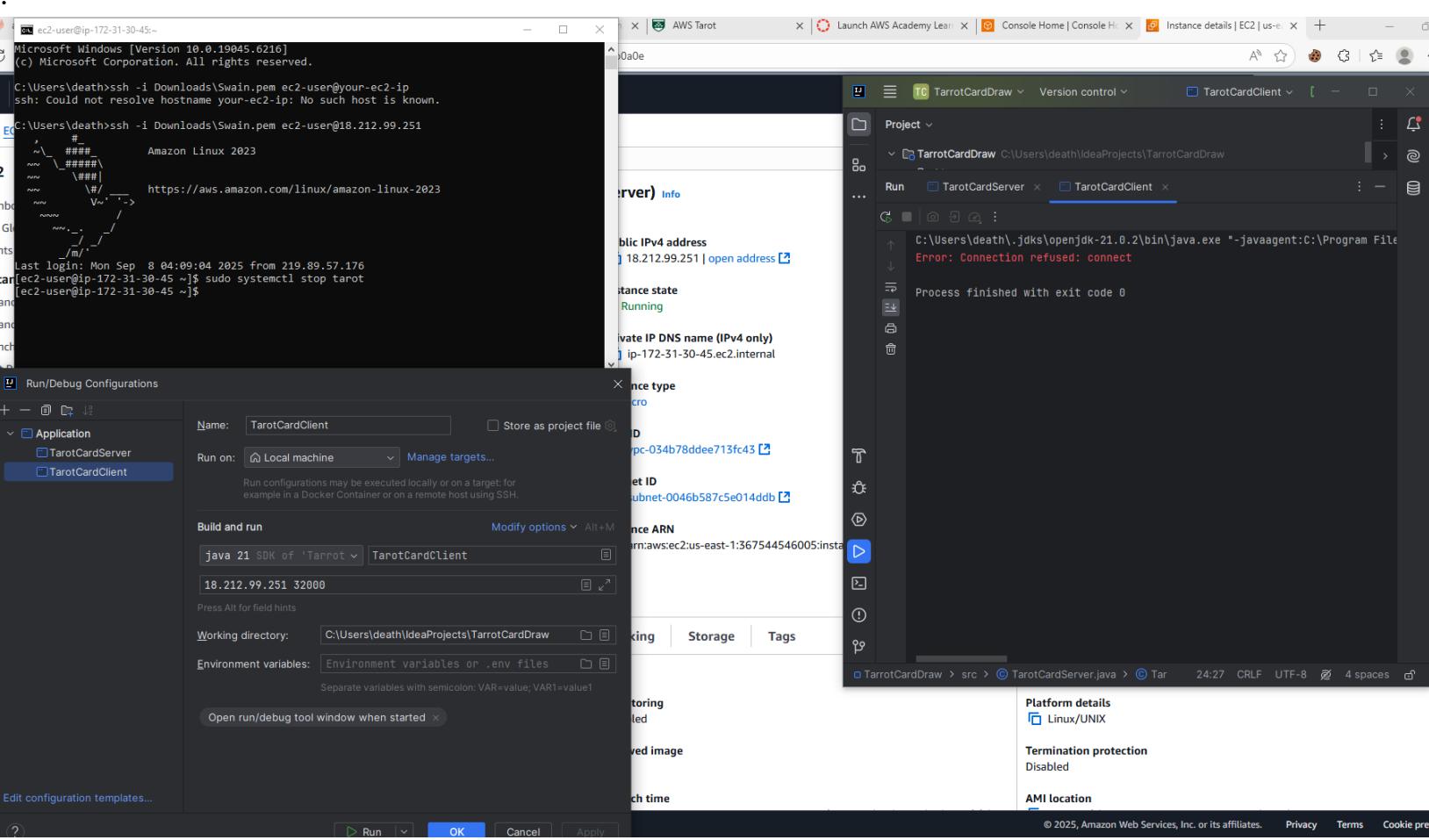
These screenshots demonstrate successful automatic startup after reboot, with Image 1 showing the systemctl status revealing the tarot service is active (running) since 06:24:28 UTC (rebooted 29 seconds ago) and system logs confirming automatic startup without manual intervention, while Images 2 and 3 show stopping of the tarot service and getting a fail then the client successfully connecting to 18.212.99.251 32000 and receiving a complete tarot reading when i start tarot service(i forgot to try connect when the service was dead before reboot), proving the systemd auto start mechanism works correctly.

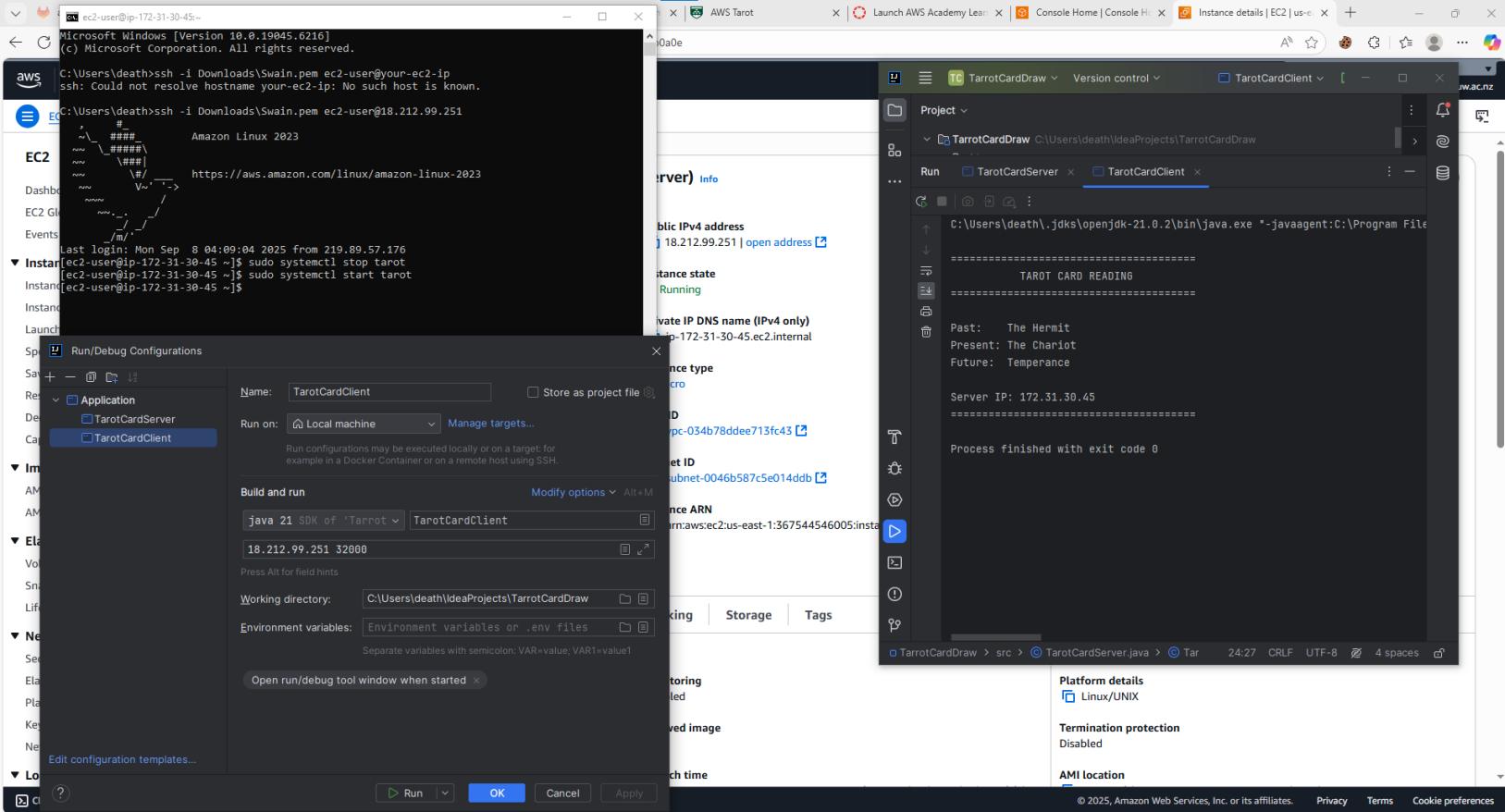
```

#_
##_      Amazon Linux 2023
##_\#####\
## \###|
## \#/   V~'-'-> https://aws.amazon.com/linux/amazon-linux-2023
##  /
##_*_/
/_/ /_
/_m/_

Last login: Mon Sep  8 05:52:54 2025 from 219.89.57.176
[ec2-user@ip-172-31-30-45 ~]$ sudo systemctl status tarot
● tarot.service - Tarot service
   Loaded: loaded (/etc/systemd/system/tarot.service; enabled; preset: disabled)
   Active: active (running) since Mon 2025-09-08 06:24:28 UTC; 29s ago
     Main PID: 1965 (run.sh)
        Tasks: 11 (limit: 1111)
       Memory: 30.2M
          CPU: 137ms
         CGroup: /system.slice/tarot.service
                   └─1965 /bin/sh /home/ec2-user/run.sh
                     ├─1969 java TarotCardServer 32000

Sep 08 06:24:28 ip-172-31-30-45.ec2.internal systemd[1]: Started tarot.service - Tarot service.
Sep 08 06:24:29 ip-172-31-30-45.ec2.internal run.sh[1969]: Tarot Card Server started on port 32000
[ec2-user@ip-172-31-30-45 ~]$
```





## Screenshot 5:

These screenshots demonstrate successful scaling of the Tarot Card Server, showing the client connecting to both the original manually configured instance and a new AMI launched instance with different server IP addresses, the AWS console displaying both instances running simultaneously, and the AMI dashboard confirming creation of an AMI.

Amazon Machine Images (AMIs) (1) <a href="#">Info</a>									
<a href="#">Owned by me</a>		<a href="#">Find AMI by attribute or tag</a>							
	Name	AMI name	AMI ID	Source	Owner	Visibility	Status	Creation date	
	NWEN Tarot image	ami-030aebcb35267b7a4	367544546005/NWEN Tarot image	367544546005	Private	<span>Available</span>	2025/09/13 14:18 GMT+		

Screenshot showing the AWS Cloud9 IDE interface with multiple windows open:

- Top Bar:** AWS logo, Search bar, [Alt+S] button, Account ID: 3675-4454-6005, United States (N. Virginia), vclabs/user4229981-trailsamu@myvuw.ac.nz
- Left Sidebar:** EC2 > Instances, Instances (2) Info, Instances (2) table with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, Public IPv4 IP, Elastic IP.
- Middle Left Window:** Project view for TarotCardClient, showing code output for "TAROT CARD READING". The output shows the Tarot Card Reader has identified the Sun, Strength, and High Priestess cards. It also lists the server IP as 172.31.18.31 and the process finished with exit code 0.
- Middle Right Window:** Run/Debug Configurations dialog for Tarot Server Image. Configuration details:
  - Name: TarotCardClient
  - Run on: Local machine
  - Build and run: java 21 SDK of 'Tarrot' TarotCardClient
  - Working directory: C:\Users\death\IdeaProjects\TarotCardDraw
  - Environment variables: Environment variables or .env files
- Bottom Left Window:** Project view for TarotCardClient, showing code output for "TAROT CARD READING". The output shows the Tarot Card Reader has identified Death, The Magician, and The Star cards. It also lists the server IP as 172.31.30.45 and the process finished with exit code 0.
- Bottom Right Window:** Run/Debug Configurations dialog for Tarot Card Server. Configuration details:
  - Name: TarotCardClient
  - Run on: Local machine
  - Build and run: java 21 SDK of 'Tarrot' TarotCardClient
  - Working directory: C:\Users\death\IdeaProjects\TarotCardDraw
  - Environment variables: Environment variables or .env files

## Part 2

### Screenshot 1

The dashboard shows the custom VPC that we made (screenshot was taken with the following steps completed)

Screenshot 2

This screenshot shows the AWS VPC configuration for the Tarot VPC. The main details are:

- VPC ID:** vpc-0a6a07b36a0e74cab
- State:** Available
- Tenancy:** default
- Default VPC:** No
- Block Public Access:** Off
- DNS hostnames:** Disabled
- DNS resolution:** Enabled
- Main network ACL:** acl-0fc481729d189921
- IPv6 CIDR (Network border group):** -
- IPv4 CIDR:** 192.168.0.0/23
- Route 53 Resolver DNS Firewall rule groups:** Failed to load rule groups
- Main route table:** rtb-0d6264bcfe291ad79
- IPv6 pool:** -
- Owner ID:** 367544546005

The Resource map shows the following components:

- VPC:** Your AWS virtual network (Tarot VPC)
- Subnets (2):** Subnets within this VPC (us-east-1a: Sub1, us-east-1b: Sub2)
- Route tables (1):** Route network traffic to resources (rtb-0d6264bcfe291ad79)
- Network Connections (1):** Connections to other networks (Tarot GW)

## Screenshot 2

This screenshot shows the AWS subnet configuration with two subnets (Sub2 and Sub1) created in different availability zones as required by the assignment. Both subnets are configured with the correct IPv4 CIDR blocks (192.168.0.0/24 and 192.168.1.0/24) and are associated with the Tarot VPC.

Screenshot 3,4

This screenshot shows the AWS Subnet configuration for two subnets: Sub2 and Sub1.

**subnet-0215ee08b9a0fbda8 / Sub2**

- Details**
- Subnet ID:** subnet-0215ee08b9a0fbda8
- IPv4 CIDR:** 192.168.1.0/24
- Availability Zone:** use1-az2 (us-east-1b)
- Network ACL:** acl-0fc481729d189921
- Auto-assign customer-owned IPv4 address:** No
- IPv6 CIDR reservations:** -
- Resource name DNS AAAA record:** Disabled
- State:** Available
- IPv6 CIDR:** -
- VPC:** vpc-0a6a07b36a0e74cab | Tarot VPC
- Auto-assign public IPv4 address:** No
- Outpost ID:** -
- Hostname type:** -
- Subnet ARN:** arn:aws:ec2:us-east-1:367544546005:subnet/subnet-0215ee08b9a0fbda8
- Available IPv4 addresses:** 251
- Network border group:** us-east-1
- Default subnet:** No
- Customer-owned IPv4 pool:** -
- IPv6-only:** No
- DNS64:** Disabled
- Block Public Access:** Off
- IPv6 CIDR association ID:** -
- Route table:** rtb-0d6264bcfe291ad79
- Auto-assign IPv6 address:** No
- IPv4 CIDR reservations:** -
- Resource name DNS A record:** -

**subnet-048e6bc29c891fcf6 / Sub1**

- Details**
- Subnet ID:** subnet-048e6bc29c891fcf6
- IPv4 CIDR:** 192.168.0.0/24
- Availability Zone:** use1-az1 (us-east-1a)
- Network ACL:** acl-0fc481729d189921
- Auto-assign customer-owned IPv4 address:** No
- IPv6 CIDR reservations:** -
- Resource name DNS AAAA record:** Disabled
- State:** Available
- IPv6 CIDR:** -
- VPC:** vpc-0a6a07b36a0e74cab | Tarot VPC
- Auto-assign public IPv4 address:** No
- Outpost ID:** -
- Hostname type:** -
- Subnet ARN:** arn:aws:ec2:us-east-1:367544546005:subnet/subnet-048e6bc29c891fcf6
- Available IPv4 addresses:** 251
- Network border group:** us-east-1
- Default subnet:** No
- Customer-owned IPv4 pool:** -
- IPv6-only:** No
- DNS64:** Disabled
- Block Public Access:** Off
- IPv6 CIDR association ID:** -
- Route table:** rtb-0d6264bcfe291ad79
- Auto-assign IPv6 address:** No
- IPv4 CIDR reservations:** -
- Resource name DNS A record:** -

Image 3: This screenshot shows the route table's subnet associations tab, displaying that both subnets (Sub1 and Sub2) are associated with the main route table, confirming the proper linking of subnets to the routing configuration.

Image 4: This screenshot shows the route table's routes configuration with two entries the local route for internal VPC traffic (192.168.0.0/23) and the internet gateway route (0.0.0.0/0) that enables external internet access, completing the routing setup for the VPC.

Screenshot of the AWS VPC Route Tables details page for route table ID rtb-0d6264bcfe291ad79.

**Details**

- Route table ID: rtb-0d6264bcfe291ad79
- Main: Yes
- Owner ID: 367544546005
- Explicit subnet associations: 2 subnets (Sub2 and Sub1)
- Edge associations: None

**Subnet associations** (2)

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
Sub2	subnet-0215ee08b9a0fbda8	192.168.1.0/24	-
Sub1	subnet-048e6bc29c891fcf6	192.168.0.0/24	-

**Subnets without explicit associations** (0)

No subnets without explicit associations. All your subnets are associated with a route table.

Screenshot of the AWS VPC Route Tables details page for route table ID rtb-0d6264bcfe291ad79.

**Details**

- Route table ID: rtb-0d6264bcfe291ad79
- Main: Yes
- Owner ID: 367544546005
- Explicit subnet associations: 2 subnets
- Edge associations: None

**Routes** (2)

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	igw-0df84a6a022baf626	Active	No	Create Route
192.168.0.0/23	local	Active	No	Create Route Table

## Screenshot 5

This screenshot shows the security group Tarot VPC Security with the required inbound rules configured one for SSH (port 22) and one for Custom TCP (port 32000), both allowing traffic from anywhere (0.0.0.0/0). The security group is properly associated with the custom Tarot VPC and will control access to the EC2 instances.

The screenshot displays the AWS VPC Security Groups console. The top navigation bar shows the account ID (3675-4454-6005) and region (United States (N. Virginia)). The main page shows a success message: "Security group (sg-0fcfd1b3d59432be4c | Tarot VPC Security) was created successfully". The left sidebar includes links for VPC dashboard, EC2 Global View, Filter by VPC, Virtual private cloud (Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, Carrier gateways, DHCP option sets, Elastic IPs, Managed prefix lists, NAT gateways, Peering connections, Route servers), Security (Network ACLs, Security groups), and PrivateLink and Lattice (Getting started, Endpoints, Endpoint services, Service networks, Lattice services, Resource configurations).

The central area shows the "sg-0fcfd1b3d59432be4c - Tarot VPC Security" details. It lists the security group name (Tarot VPC Security), security group ID (sg-0fcfd1b3d59432be4c), owner (367544546005), description (security group for Tarot VPC), and VPC ID (vpc-0a6a07b36a0e74cab). It also shows the inbound rules count (2 Permission entries) and outbound rules count (1 Permission entry).

The "Inbound rules" tab is selected, displaying two rules:

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
-	sgr-0fcadd2a8b75848ed	IPv4	Custom TCP	TCP	32000	0.0.0.0/0	-
-	sgr-0fc45e96ca971d822	IPv4	SSH	TCP	22	0.0.0.0/0	-

## Screenshot 6,7

These two screenshots show the successful creation of the Tarot\_Template launch template configured with the custom AMI, t3.micro instance type, and the Tarot VPC Security group. The security group ID (sg-0fcfd1b3d59432be4c) matches between both images, confirming that the launch template is properly linked to the correct security group even though the name doesn't display in the template details view.

**Tarot\_Template (lt-0bbc849a35d14aab1)**

**Launch template details**

Launch template ID <a href="#">lt-0bbc849a35d14aab1</a>	Launch template name <a href="#">Tarot_Template</a>	Default version <a href="#">1</a>	Owner arn:aws:sts::367544546005:assumed-role/voclabs/user4229981=trailsamu@myvuw.ac.nz
--	--	--------------------------------------	---

**Details** | **Versions** | **Template tags**

**Launch template version details**

Version 1 (Default)	Description <a href="#">Template for Tarot Server</a>	Date created <a href="#">2025-09-13T03:30:50.000Z</a>	Created by arn:aws:sts::367544546005:assumed-role/voclabs/user4229981=trailsamu@myvuw.ac.nz
<b>Instance details</b>		Storage	Resource tags
Network interfaces		Advanced details	
AMI ID <a href="#">ami-030aebcb35267b7a4</a>	Instance type <a href="#">t3.micro</a>	Availability Zone -	Availability Zone Id -
Key pair name <a href="#">Swain</a>	Security groups -	Security group IDs <a href="#">sg-0fcfcd1b3d59432be4c</a>	

**EC2**

[Search](#) [Alt+S]

Account ID: 3675-4454-6005  
voclabs/user4229981=trailsamu@myvuw.ac.nz

[EC2](#) > [Security Groups](#) > sg-0fcfcd1b3d59432be4c - Tarot VPC Security

**sg-0fcfcd1b3d59432be4c - Tarot VPC Security**

<b>Details</b>	<b>Security group ID</b> <a href="#">sg-0fcfcd1b3d59432be4c</a>	<b>Description</b> <a href="#">security group for Tarot VPC</a>	<b>VPC ID</b> <a href="#">vpc-0a6a07b36a0e74cab</a>
Security group name <a href="#">Tarot VPC Security</a>	Owner <a href="#">367544546005</a>	Inbound rules count 2 Permission entries	Outbound rules count 1 Permission entry

**Inbound rules** | **Outbound rules** | **Sharing - new** | **VPC associations - new** | **Tags**

**Inbound rules (2)**

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
-	sgr-0fcadd2a8b75848ed	IPv4	Custom TCP	TCP	32000	0.0.0.0/0	-
-	sgr-04c45e96ca971d822	IPv4	SSH	TCP	22	0.0.0.0/0	-

## Screenshot 8,9,10,11,12

These screenshots demonstrate the successful testing phase of the launch template creation, showing the EC2 instances dashboard with four running instances (two original named instance and two unnamed instances created from the template), followed by comprehensive testing of each instance through SSH connections and Tarot card client connections across different IP addresses. The testing confirms that instances created from the launch template in both subnets are properly configured with functional SSH access on port 22 and the custom Tarot card server running on port 32000, validating that the template works correctly.

```
ec2-user@ip-192-168-0-150:~$  
Microsoft Windows [Version 10.0.19045.6332]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\death\cd Downloads  
  
C:\Users\death\Downloads>ssh -i Swain.pem ec2-user@3.224.147.171  
The authenticity of host '3.224.147.171 (3.224.147.171)' can't be established.  
ED25519 key fingerprint is SHA256:UAI56EFL6Z/WScBvEuKFBzQK1hwMcsJpt/BU+M.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '3.224.147.171' (ED25519) to the list of known hosts.  
  
A newer release of "Amazon Linux" is available.  
Version 2023.8.20250908:  
Run "/usr/bin/dnf check-release-update" for full release and version update info  
,  
~~~  
~~~ # Amazon Linux 2023  
~~~ #####  
~~~ \###/  
~~~ \###/  
~~~ \#/. https://aws.amazon.com/linux/amazon-linux-2023  
~~~ \~-.>  
~~~  
~~~  
~~~  
~~~  
~~~  
~~~  
~~~  
Last login: Mon Sep 8 06:24:52 2025 from 219.89.57.176  
[ec2-user@ip-192-168-0-150 ~]$ sudo systemctl status tarot  
● tarot.service - Tarot service  
   Loaded: loaded (/etc/systemd/system/tarot.service; enabled; preset: disabled)  
   Active: active (running) since Sat 2025-09-13 03:41:39 UTC; 4min 54s ago  
     Main PID: 1402 (run.sh)  
        Tasks: 11 (limit: 1057)  
       Memory: 30.1M  
          CPU: 386ms  
      CGroup: /system.slice/tarot.service  
              └─1402 /bin/sh /home/ec2-user/run.sh  
                  ├─1437 java TarotCardServer 32000  
  
Sep 13 03:41:39 ip-172-31-30-45.ec2.internal systemd[1]: Started tarot.service  
Sep 13 03:41:39 ip-192-168-0-150.ec2.internal run.sh[1437]: Tarot Card Server  
[ec2-user@ip-192-168-0-150 ~]$
```

EC2 > Instances > i-08fc1b7eeeaa4d6f52

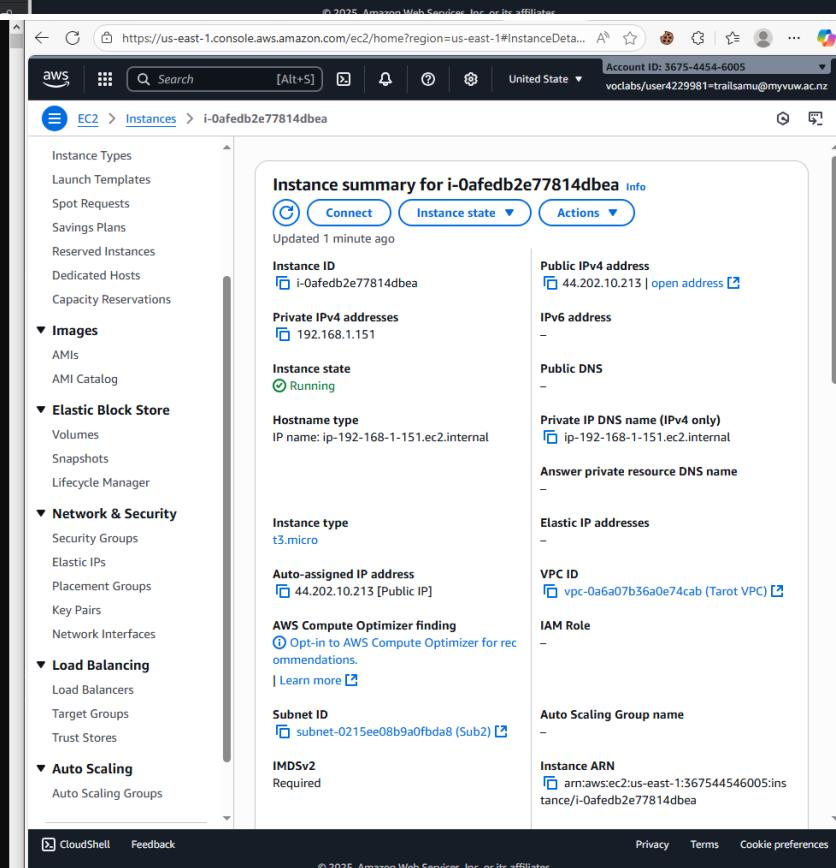
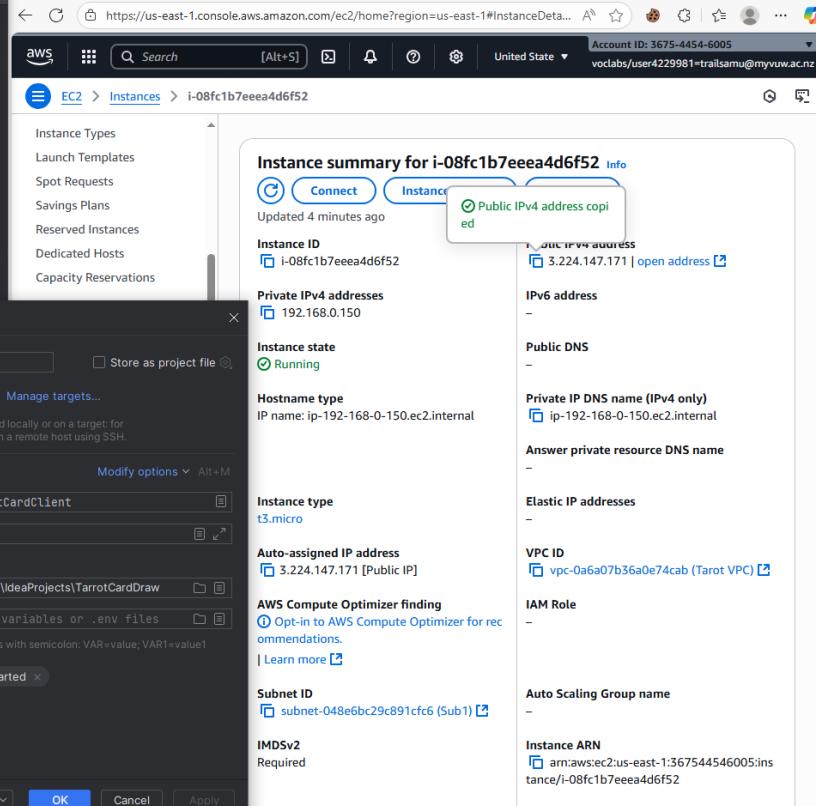
Instance Types  
Launch Templates  
Spot Requests  
Savings Plans  
Reserved Instances  
Dedicated Hosts  
Capacity Reservations  
Images  
AMIs  
AMI Catalog  
Elastic Block Store  
Volumes  
Snapshots  
Lifecycle Manager  
Network & Security  
Security Groups  
Elastic IPs  
Placement Groups  
Key Pairs  
Network Interfaces  
Load Balancing  
Load Balancers  
Target Groups  
Trust Stores  
Auto Scaling  
Auto Scaling Groups

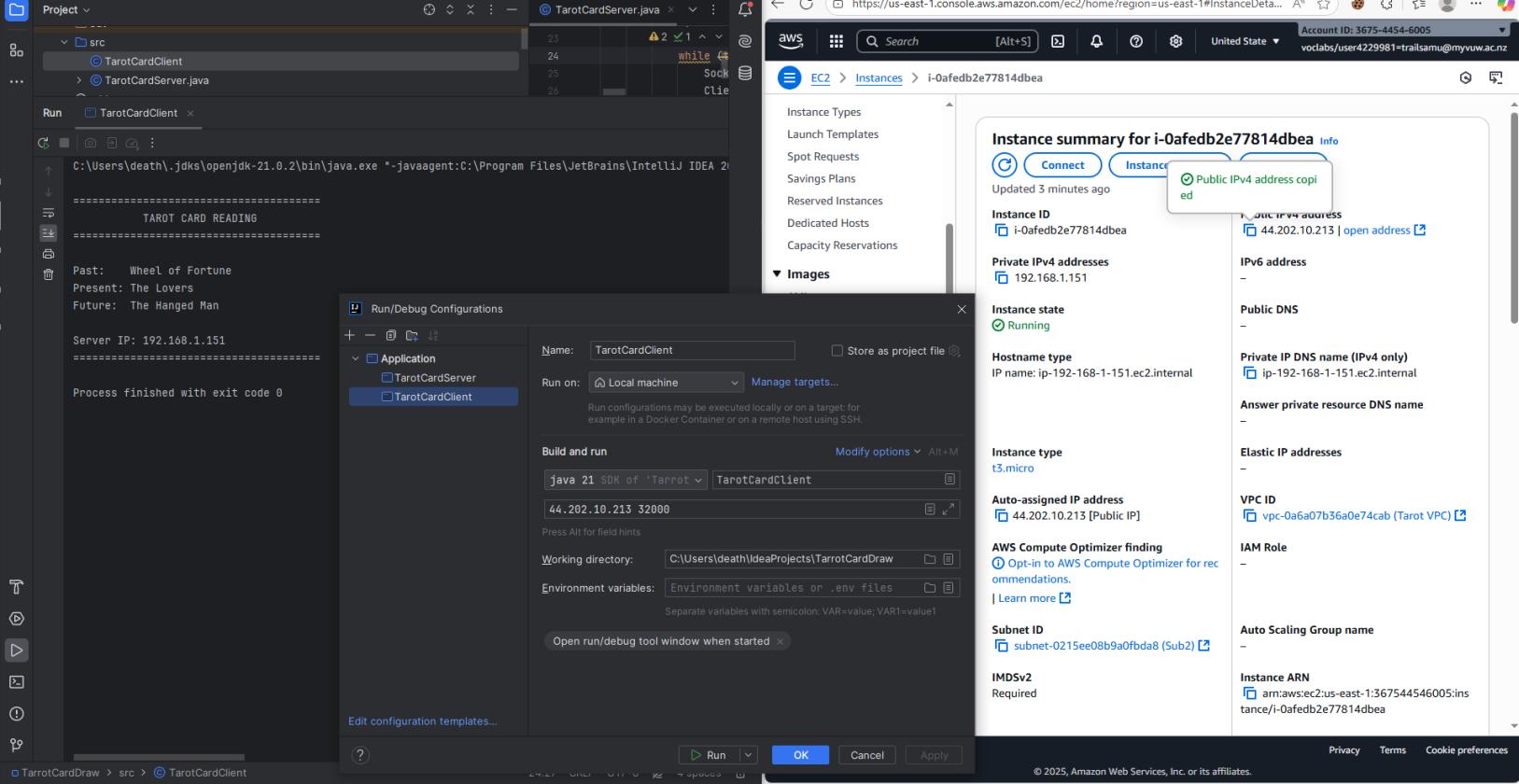
Account ID: 3675-4454-6005  
voclabs/user@229981=trailsamu@myvuw.ac.nz

**Instance summary for i-08fc1b7eeeaa4d6f52**

Updated 2 minutes ago

Instance ID	i-08fc1b7eeeaa4d6f52	Public IPv4 address	3.224.147.171   open address
Private IPv4 addresses	192.168.0.150	IPv6 address	-
Instance state	Running	Public DNS	-
Hostname type	IP name: ip-192-168-0-150.ec2.internal	Private IP DNS name (IPv4 only)	ip-192-168-0-150.ec2.internal
Instance type	t3.micro	Answer private resource DNS name	-
Auto-assigned IP address	3.224.147.171 [Public IP]	Elastic IP addresses	-
AWS Compute Optimizer finding	<a href="#">Opt-in to AWS Compute Optimizer for recommendations.</a>	VPC ID	vpc-0a6a07b36a0e74cab (Tarot VPC)
IMDSv2	Required	IAM Role	-
Subnet ID	subnet-048e6bc29c891fcf6 (Sub1)	Auto Scaling Group name	-
Instance ARN	arn:aws:ec2:us-east-1:367544546005:instance/i-08fc1b7eeeaa4d6f52		





## Screenshot 13,14

These screenshots demonstrate the successful creation of the load balancing infrastructure, showing the Tarot-TargetGroup configured for TCP traffic on port 32000 with zero registered targets (as expected before auto scaling group creation), and the Tarot-Balancer network load balancer in provisioning status, properly configured as internet facing across two availability zones in the custom VPC with a listener that forwards port 32000 traffic to the target group, completing the load balancer setup.

EC2 < Tarot-TargetGroup

## Tarot-TargetGroup

**Details**

arn:aws:elasticloadbalancing:us-east-1:367544546005:targetgroup/Tarot-TargetGroup/d5225b5039079bfc

Target type Instance	Protocol : Port TCP: 32000	VPC vpc-0a6a07b36a0e74cab	IP address type IPv4
Load balancer Tarot-Balancer			
Total targets 0	Healthy 0	Unhealthy 0	Unused 0
Initial 0			Draining 0

Targets | Monitoring | Health checks | Attributes | Tags

### Registered targets (0)

No registered targets  
You have not registered targets to this group yet

Register targets

EC2 > Load balancers > Tarot-Balancer

Successfully created load balancer

## Tarot-Balancer

**Details**

Load balancer type Network	Status Provisioning	VPC vpc-0a6a07b36a0e74cab	Load balancer IP address type IPv4
Scheme Internet-facing	Hosted zone Z26RNL4JYFTOTI	Availability Zones subnet-048e6bc29c891fcf6 us-east-1a (use1-az1) subnet-0215ee08b9a0fbda8 us-east-1b (use1-az2)	Date created September 16, 2025, 14:44 (UTC+12:00)
Load balancer ARN arn:aws:elasticloadbalancing:us-east-1:367544546005:loadbalancer/net/Tarot-Balancer/eb0bb8be380a8ddd		DNS name Info Tarot-Balancer-eb0bb8be380a8ddd.elb.us-east-1.amazonaws.com (A Record)	

Listeners | Network mapping | Resource map | Security | Monitoring | Integrations | Attributes | Capacity | Tags

### Listeners (1)

A listener checks for connection requests using the protocol and port that you configure. Traffic received by a Network Load Balancer listener is forwarded to the selected target group.

Filter listeners

Protocol:Port	Default action	ARN	Security policy	Default SSL/TLS certificate	ALPN policy	Tags
TCP:32000	Forward to target group Tarot-TargetGroup	ARN	Not applicable	Not applicable	None	0 tags

# Screenshot 15,16,17,18

## Auto scaling group settings

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

EC2 > Auto Scaling groups > Create Auto Scaling group

Step 3 - optional  
 Integrate with other services  
 Step 4 - optional  
 Step 5 - optional  
 Add notifications  
Step 6 - optional  
 Add tags  
Step 7  
 Review

### Load balancing Info

Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.

No load balancer  
 Attach to an existing load balancer  
Traffic to your Auto Scaling group will not be fronted by a load balancer.  
 Attach to a new load balancer  
Quickly create a basic load balancer to attach to your Auto Scaling group.

### Attach to an existing load balancer

Select the load balancers that you want to attach to your Auto Scaling group.

Choose from your load balancer target groups  
This option allows you to attach Application, Network, or Gateway Load Balancers.  
 Choose from Classic Load Balancers

Existing load balancer target groups  
Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.  
Select target groups  
Tarot-TargetGroup | TCP  
Network Load Balancer: Tarot-Balancer

### VPC Lattice integration options Info

To improve networking capabilities and scalability, integrate your Auto Scaling group with VPC Lattice. VPC Lattice facilitates communications between AWS services and helps you connect and manage your applications across compute services in AWS.

Select VPC Lattice service to attach  
 No VPC Lattice service  
VPC Lattice will manage your Auto Scaling group's network access and connectivity with other services.  
 Attach to VPC Lattice service  
Incoming requests associated with specified VPC Lattice target groups will be routed to your Auto Scaling group.

Create new VPC Lattice service [Create](#)

### Application Recovery Controller (ARC) zonal shift - new Info

During an Availability Zone impairment, target instance launches towards other healthy Availability Zones.

Enable zonal shift  
New instance launches will be retargeted towards healthy Availability Zones until the zonal shift is canceled.

### Health checks

Health checks increase availability by replacing unhealthy instances. When you use multiple health checks, all are evaluated, and if at least one fails, instance replacement occurs.

EC2 health checks  
 Always enabled

Additional health check types - optional [Info](#)  
 Turn on Elastic Load Balancing health checks [Recommended](#)  
Enable CloudWatch Metrics whenever instances are available in multiple regions. When instances are unhealthy, CloudWatch Metrics can measure if the instance is healthy.

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

EC2 > Auto Scaling groups > Create Auto Scaling group

Step 1  
 Choose launch template  
 Choose instance launch options  
 Step 2  
 Step 3 - optional  
 Integrate with other services  
 Step 4 - optional  
 Configure group size and scaling  
 Step 5 - optional  
 Add notifications  
 Step 6 - optional  
 Add tags  
 Step 7  
 Review

### Choose instance launch options Info

Choose the VPC network environment that your instances are launched into, and customize the instance types and purchase options.

#### Instance type requirements Info

You can keep the same instance attributes or instance type from your launch template, or you can choose to override the launch template by specifying different instance attributes or manually adding instance types.

Launch template	Version	Description
Tarot_LaunchTemplate <a href="#">Edit</a> lt-0bbc845a35d14aab1	2	-
Instance type		
t3.micro		

[Override launch template](#)

### Network Info

For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

**VPC**  
Choose the VPC that defines the virtual network for your Auto Scaling group.  
vpc-0a60703636a0e74cab (Tarot VPC)  
192.168.0.0/23  
[Create a VPC](#)

**Availability Zones and subnets**  
Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.  
Select Availability Zones and subnets  
use1-aZ1 (us-east-1a) | subnet-048e6bc29c891fc6 (Sub1)  
192.168.0.0/24  
use1-aZ2 (us-east-1b) | subnet-0215ec08b5a0fbda8 (Sub2)  
192.168.1.0/24  
[Create a subnet](#)

**Availability Zone distribution - new**  
Auto Scaling automatically balances instances across Availability Zones. If launch failures occur in a zone, select a strategy.

Balanced best effort  
If launches fail in one Availability Zone, Auto Scaling will attempt to launch in another healthy Availability Zone.  
 Balanced only  
If launches fail in one Availability Zone, Auto Scaling will continue to attempt to launch in the unhealthy Availability Zone to preserve balanced distribution.

[Cancel](#) [Skip to review](#) [Previous](#) [Next](#)

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

EC2 > Auto Scaling groups > Create Auto Scaling group

**Step 1 Choose launch template**

- Choose launch template
- Choose instance launch options
- Step 3 - optional Integrate with other services
- Step 4 - optional Configure group size and scaling
- Step 5 - optional Add notifications
- Step 6 - optional Add tags
- Step 7 Review

**Choose launch template Info**  
Specify a launch template that contains settings common to all EC2 Instances that are launched by this Auto Scaling group.

**Name**  
**Auto Scaling group name**  
Enter a name to identify the group.  
**Tarot Scaling Group**  
Must be unique to this account in the current Region and no more than 255 characters.

**Launch template Info**  
Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.  
**Tarot\_Template**  
Create a launch template Info Version 2 Cancel Next

**Description**  
-

<b>Launch template</b> <a href="#">Tarot_Template</a> lt-0bbcb849a35d14aab1	<b>Instance type</b> t3.micro
<b>AMI ID</b> ami-030aebcb35267b7a4	<b>Security groups</b> -
<b>Key pair name</b> Swain	<b>Security group IDs</b> sg-0fc1b3d59432be4c <a href="#">Edit</a>
<b>Additional details</b>	
<b>Storage (volumes)</b> -	<b>Date created</b> Sat Sep 13 2025 15:33:33 GMT+1200 (New Zealand Standard Time)

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

EC2 > Auto Scaling groups > Create Auto Scaling group

**Step 1 Choose launch template**

- Choose launch template
- Choose instance launch options
- Step 3 - optional Integrate with other services
- Step 4 - optional Configure group size and scaling
- Step 5 - optional Add notifications
- Step 6 - optional Add tags
- Step 7 Review

**Configure group size and scaling - optional Info**  
Define your group's desired capacity and scaling limits. You can optionally add automatic scaling to adjust the size of your group.

**Group size Info**  
Set the initial size of the Auto Scaling group. After creating the group, you can change its size to meet demand, either manually or by using automatic scaling.

**Desired capacity type**  
Choose the unit of measurement for the desired capacity value. vCPUs and Memory(GiB) are only supported for mixed instances groups configured with a set of instance attributes.

**Units (number of instances)**

**Desired capacity**  
Specify your group size.  
2

**Scaling Info**  
You can resize your Auto Scaling group manually or automatically to meet changes in demand.

**Scaling limits**  
Set limits on how much your desired capacity can be increased or decreased.

**Min desired capacity** 2 Equal or less than desired capacity

**Max desired capacity** 3 Equal or greater than desired capacity

**Automatic scaling - optional**  
Choose whether to use a target tracking policy Info  
You can set up other metric-based scaling policies and scheduled scaling after creating your Auto Scaling group.

No scaling policies Your Auto Scaling group will remain at its initial size and will not dynamically resize to meet demand.

Target tracking scaling policy Choose a CloudWatch metric and target value and let the scaling policy adjust the desired capacity in proportion to the metric's value.

**Scaling policy name**  
Target Tracking Policy

**Metric type Info**

## Screenshot 19

This screenshot shows the Tarot Scaling Group configuration page, displaying the setup including the launch template selection (Tarot\_Template), network configuration with the custom VPC and both subnets across different availability zones, load balancer settings, instance type requirements, and health check configurations. The interface shows all

The key components properly configured for the auto scaling group that will automatically manage instance creation and integrate with the load balancer to distribute traffic across multiple availability zones.

This screenshot shows the AWS EC2 Auto Scaling Groups console for the 'Tarot Scaling Group'. The left sidebar includes navigation links for EC2, Dashboard, EC2 Global View, Events, Instances (with sub-links for 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 (Load Balancers, Target Groups, Trust Stores), Auto Scaling (Auto Scaling Groups), and Settings. The main content area displays the 'Tarot Scaling Group Capacity overview' with fields for Desired capacity (2), Scaling limits (Min - Max) (2 - 3), Desired capacity type (Units (number of instances)), Status (Not running), and Date created (Tue Sep 16 2025 14:53:19 GMT+1200 (New Zealand Standard Time)). Below this are tabs for Details, Integrations - new, Automatic scaling, Instance management, Instance refresh, Activity, and Monitoring. The 'Details' tab is selected, showing the 'Launch template' configuration. It includes fields for Launch template (arn:aws:autoscaling:us-east-1:1367544546005:autoScalingGroup:34cd4d12-f551-40d2-8b75-1d8dbce12480:autoScalingGroupName/Tarot Scaling Group), AMI ID (ami-030aeccb35267b7a4), Instance type (t1.micro), Owner (arn:aws:sts::367544546005:assumed-role/vocabs/user4229981-tralsamui@myvuw.ac.nz), and Create time (Sat Sep 13 2025 15:33:33 GMT+1200 (New Zealand Standard Time)). Other launch template fields include Version (2), Security groups, Security group IDs (sg-0fd1b5d59432bedc), Description, Storage (volumes), Key pair name (Swain), and Request Spot Instances (No). Below the launch template is a 'Network' section with Availability Zones (use1-az2 (us-east-1b), use1-az1 (us-east-1a)), Subnet ID (subnet-0215ee08b9a0fbd8), and Availability Zone distribution (Balanced best effort). A note indicates that Load balancing and VPC Lattice options have moved to the new integrations tab. The 'Health checks' section shows Health check type (EC2, ELB) and Health check grace period (300). Each section has an 'Edit' button in the top right corner.

## Screenshot 20,21,22

These three screenshots demonstrate the successful deployment and testing of the auto scaling group, showing the EC2 instances dashboard with multiple running instances created by the auto scaling group, followed by successful client testing of the two specific instances. The testing confirms that the auto scaling group is automatically creating properly configured instances across different availability zones that can be accessed both directly via the Tarot card client application.

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
Tarot Card Se...	i-0929c67f7d1bb0a0e	Running	t2.micro	2/2 checks passed	<a href="#">View alarms +</a>	us-east-1c	ec2-13-218-39-247.co...	13.218.39.247	-
Tarot Server I...	i-0615b6393f55e995d	Running	t3.micro	3/3 checks passed	<a href="#">View alarms +</a>	us-east-1c	ec2-34-227-52-184.co...	34.227.52.184	-
	i-0dd53392b27ffa18a	Running	t3.micro	3/3 checks passed	<a href="#">View alarms +</a>	us-east-1a	-	44.199.210.217	-
	i-057fc970243050c53	Running	t3.micro	3/3 checks passed	<a href="#">View alarms +</a>	us-east-1b	-	3.84.112.148	-
	i-08fc1b7eeeaa4df6f52	Terminated	t3.micro	-	<a href="#">View alarms +</a>	us-east-1a	-	-	-
	i-0afedb2e77814dbea	Terminated	t3.micro	-	<a href="#">View alarms +</a>	us-east-1b	-	-	-

Select an instance

CloudShell Feedback

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

Project

TarotCardDraw C:\Users\death\IdeaProjects\TarotCardDraw

Run TarotCardClient

```
C:\Users\death\.jdks\openjdk-21.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.1\lib\idea_rt.jar" -Dfile.encoding=UTF-8 -jar C:\Users\death\IdeaProjects\TarotCardDraw\tarotcarddraw.jar
```

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

Past: The Hierophant  
Present: The Devil  
Future: The Tower

Server IP: 192.168.0.108  
=====

Process finished with exit code 0

Run/Debug Configurations

Name: TarotCardClient  
Run on: Local machine  
Build and run: java 21 SDK of 'TarotCardDraw' module  
Working directory: C:\Users\death\IdeaProjects\TarotCardDraw  
Environment variables: Environment variables or .env files  
Edit configuration templates...

OK Cancel Apply

CloudShell Feedback

© 2025, Amazon Web Services, Inc. or its affiliates.

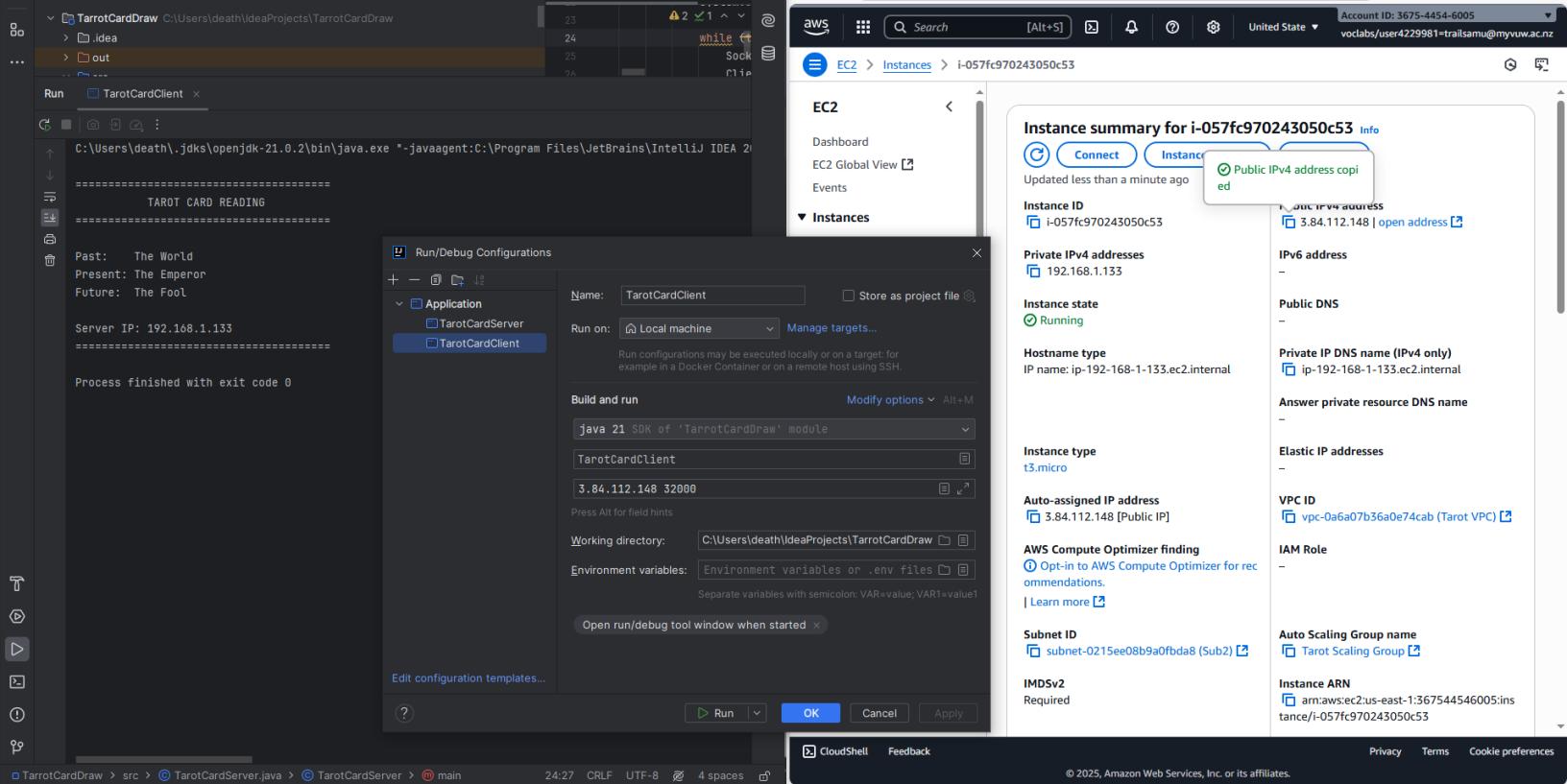
EC2 > Instances > i-0dd53392b27ffa18a

**Instance summary for i-0dd53392b27ffa18a**

Instance ID	i-0dd53392b27ffa18a	Public IPv4 address	44.199.210.217   open address
Private IPv4 addresses	192.168.0.108	IPv6 address	-
Instance state	Running	Public DNS	-
Hostname type	IP name: ip-192-168-0-108.ec2.internal	Private IP DNS name (IPv4 only)	ip-192-168-0-108.ec2.internal
Instance type	t3.micro	Answer private resource DNS name	-
Auto-assigned IP address	44.199.210.217 [Public IP]	Elastic IP addresses	-
AWS Compute Optimizer finding	Opt-in to AWS Compute Optimizer for recommendations.	VPC ID	vpc-0a6a07b36a0e74cab (Tarot VPC)
IAM Role	-	Auto Scaling Group name	Tarot Scaling Group
Subnet ID	subnet-048e6bc29c891cf6 (Sub1)	Instance ARN	arn:aws:ec2:us-east-1:367544546005:instance/i-0dd53392b27ffa18a
IMDSv2	Required	Privacy Terms Cookie preferences	

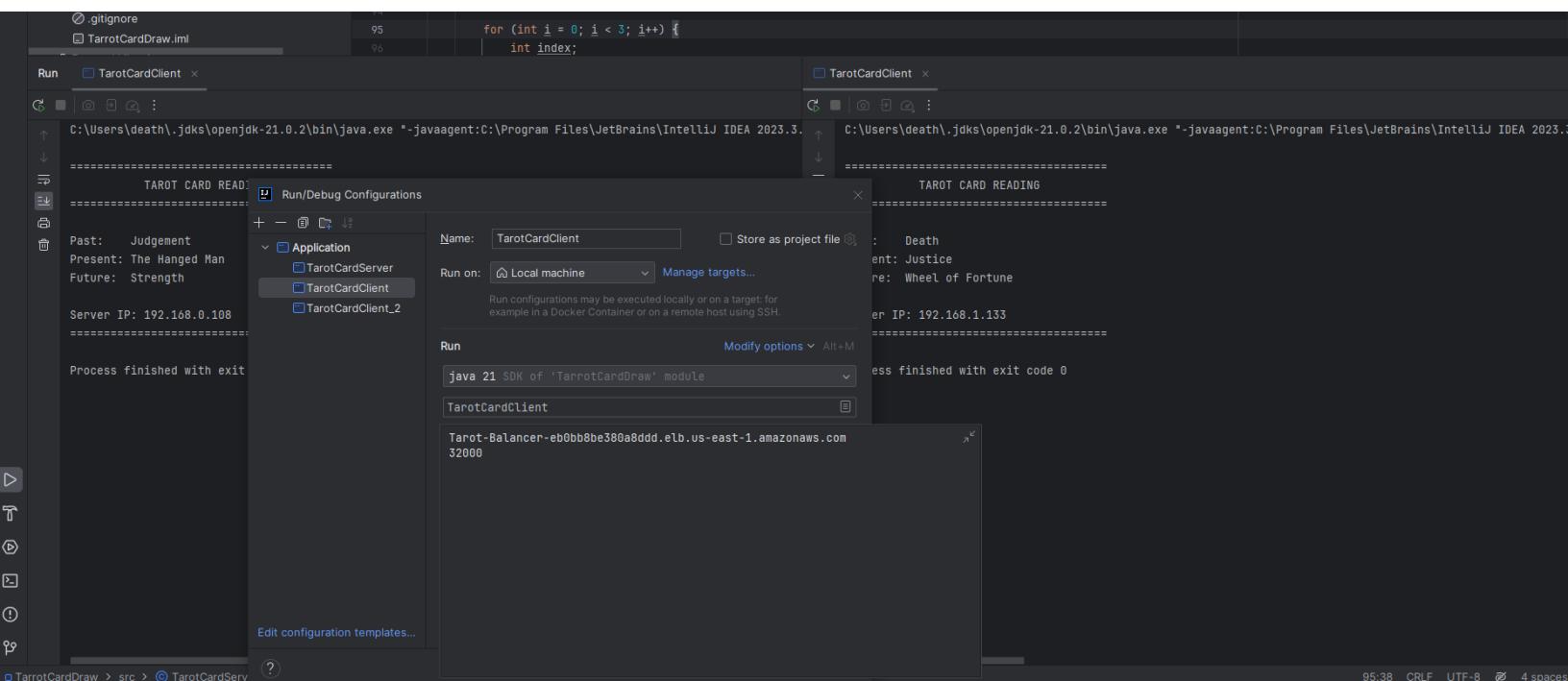
CloudShell Feedback

© 2025, Amazon Web Services, Inc. or its affiliates.



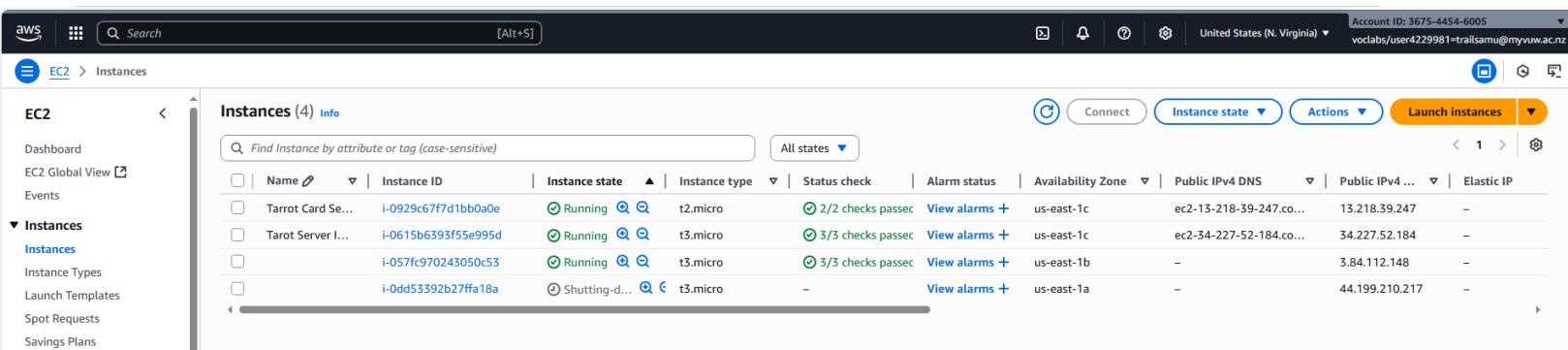
## Screenshot 23

This screenshot demonstrates the successful testing of the load balancer functionality, showing the Tarot card client connecting to the load balancer's DNS name on port 32000, which transparently routes traffic to backend instances running in different availability zones (as evidenced by the different internal IP addresses 192.168.1.132 and 192.168.0.108 being contacted through the same load balancer endpoint). This confirms that the load balancer is properly distributing client requests across multiple instances.



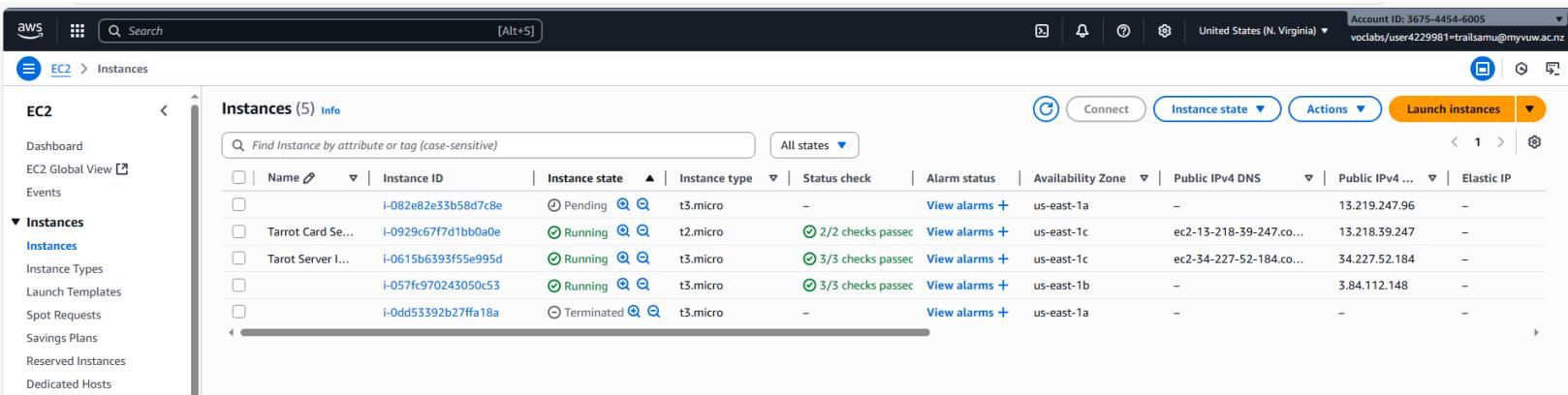
# Notes:

When I terminated one instance, the auto scaling group immediately detected the termination and automatically launched a new replacement instance to maintain the desired capacity. The load balancer continued routing traffic to the remaining healthy instance, ensuring no service interruption occurred.



This screenshot shows the AWS EC2 Instances page with 4 instances listed. The instances are:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 IP	Elastic IP
Tarrot Card Se...	i-0929c67f7d1bb0a0e	Running	t2.micro	2/2 checks passed	<a href="#">View alarms</a>	us-east-1c	ec2-13-218-39-247.co...	13.218.39.247	-
Tarot Server I...	i-0615b6393f55e995d	Running	t3.micro	3/3 checks passed	<a href="#">View alarms</a>	us-east-1c	ec2-34-227-52-184.co...	34.227.52.184	-
	i-057fc970243050c53	Running	t3.micro	3/3 checks passed	<a href="#">View alarms</a>	us-east-1b	-	3.84.112.148	-
	i-0dd53392b27ffa18a	Shutting-down	t3.micro	-	<a href="#">View alarms</a>	us-east-1a	-	44.199.210.217	-



This screenshot shows the AWS EC2 Instances page with 5 instances listed. The instances are:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 IP	Elastic IP
	i-082e82e33b58d7c8e	Pending	t3.micro	-	<a href="#">View alarms</a>	us-east-1a	-	13.219.247.96	-
	i-0929c67f7d1bb0a0e	Running	t2.micro	2/2 checks passed	<a href="#">View alarms</a>	us-east-1c	ec2-13-218-39-247.co...	13.218.39.247	-
	i-0615b6393f55e995d	Running	t3.micro	3/3 checks passed	<a href="#">View alarms</a>	us-east-1c	ec2-34-227-52-184.co...	34.227.52.184	-
	i-057fc970243050c53	Running	t3.micro	3/3 checks passed	<a href="#">View alarms</a>	us-east-1b	-	3.84.112.148	-
	i-0dd53392b27ffa18a	Terminated	t3.micro	-	<a href="#">View alarms</a>	us-east-1a	-	-	-

AWS EC2 Target groups

**Tarot-TargetGroup**

**Details**

arn:aws:elasticloadbalancing:us-east-1:367544546005:targetgroup/Tarot-TargetGroup/d5225b5039079bfcc

Target type Instance	Protocol : Port TCP: 32000	VPC <a href="#">vpc-0a6a07b36a0e74cab</a>	IP address type IPv4
Load balancer <a href="#">Tarot-Balancer</a>			
Total targets 2	Healthy 1	Unhealthy 0	Unused 0
	Initial 1		Draining 0

**Distribution of targets by Availability Zone (AZ)**  
Select values in this table to see corresponding filters applied to the Registered targets table below.

**Targets** | Monitoring | Health checks | Attributes | Tags

**Registered targets (2)**

<input type="checkbox"/> Instance ID	Name	Port	Zone	Health status	Health status det...	Administrative ...	Override details	Launch time
<a href="#">i-082e82e33b58d...</a>		32000	us-east-1a (use1-...)	<span>Initial</span>	Initial health che...	<input type="checkbox"/> No override	No override is curr...	September 16, 20...
<a href="#">i-057fc97024305...</a>		32000	us-east-1b (use1-...)	<span>Healthy</span>		<input type="checkbox"/> No override	No override is curr...	September 16, 20...

**Actions**

When I reduced the desired capacity from 2 to 1 and max/min to 1, auto scaling immediately began terminating one instance to match the new desired capacity. The load balancer put the instance into draining status, allowing existing connections to finish before fully terminating the instance. Unlike Question 1, this instance will not be replaced since the reduced capacity is intentional.

AWS EC2 Auto Scaling groups

**Tarot Scaling Group**

**Tarot Scaling Group Capacity overview**

arn:aws:autoscaling:us-east-1:367544546005:autoScalingGroup:34cd4d12-f353-40d2-8b75-1d8dbce12480:autoScalingGroupName/Tarot Scaling Group

Desired capacity 2	Scaling limits (Min - Max) 2 - 3	Required capacity (Max) 2	Status
Date created Tue Sep 16 2025 14:53:19 GMT+1200 (New Zealand Standard Time)			

**Details** | Integrations - new | Automatic scaling | Instances

**Launch template**

Launch template  
[lt-0bbc849a35d14aab1](#)  
Tarot\_Template

AMI ID  
[ami-030ae](#)

Version  
2

Description  
-

**Group size**  
Specify the size of the Auto Scaling group by changing the desired capacity. You can also specify minimum and maximum scaling limits.

**Desired capacity type**  
Choose the unit of measurement for the desired capacity value. vCPUs and Memory(GiB) are only supported for mixed instances groups configured with a set of instance attributes.

**Desired capacity**  
Specify your group size.

**Scaling limits**  
Set limits on how much your desired capacity can be increased or decreased.  
Min desired capacity  
  
Max desired capacity  
  
Equal or less than desired capacity | Equal or greater than desired capacity

**Owner**  
arn:aws:sts::367544546005:assumed-role/voclabs/user4229981-trailsamu@myvuw.ac.nz

**Create time**  
Sat Sep 13 2025 15:33:33 GMT+1200 (New Zealand Standard Time)

**Request Spot Instances**  
No

**Network**

Availability Zones  
use1-az2 (us-east-1b)  
use1-az1 (us-east-1a)

Subnet ID  
[subnet-0215ee08b9a0fbda8](#)

Availability Zone distribution  
Balanced best effort

**Actions**

EC2 > Target groups > Tarot-TargetGroup

## Tarot-TargetGroup

**Details**

arn:aws:elasticloadbalancing:us-east-1:367544546005:targetgroup/Tarot-TargetGroup/d5225b5039079bfc

Target type Instance	Protocol : Port TCP: 32000	VPC vpc-0a6a07b56a0e74cab	IP address type IPv4
Load balancer Tarot-Balancer			
Total targets 2	Healthy 1	Unhealthy 0	Unused 0
Initial 0			Draining 1

► Distribution of targets by Availability Zone (AZ)  
Select values in this table to see corresponding filters applied to the Registered targets table below.

**Targets** Monitoring Health checks Attributes Tags

### Registered targets (2)

<input type="checkbox"/> Instance ID	Name	Port	Zone	Health status	Health status det...	Administrative ...	Override details	Launch time
<input type="checkbox"/> i-082e82e33b58d...		32000	us-east-1a (use1-...)	<span>Healthy</span>		<input type="radio"/> No override	No override is curr...	September 16, 20...
<input checked="" type="checkbox"/> i-057fc97024305...		32000	us-east-1b (use1-...)	<span>Draining</span>	Target deregistrat...	<input type="radio"/> No override	No override is curr...	September 16, 20...

Filter targets Deregister Register targets