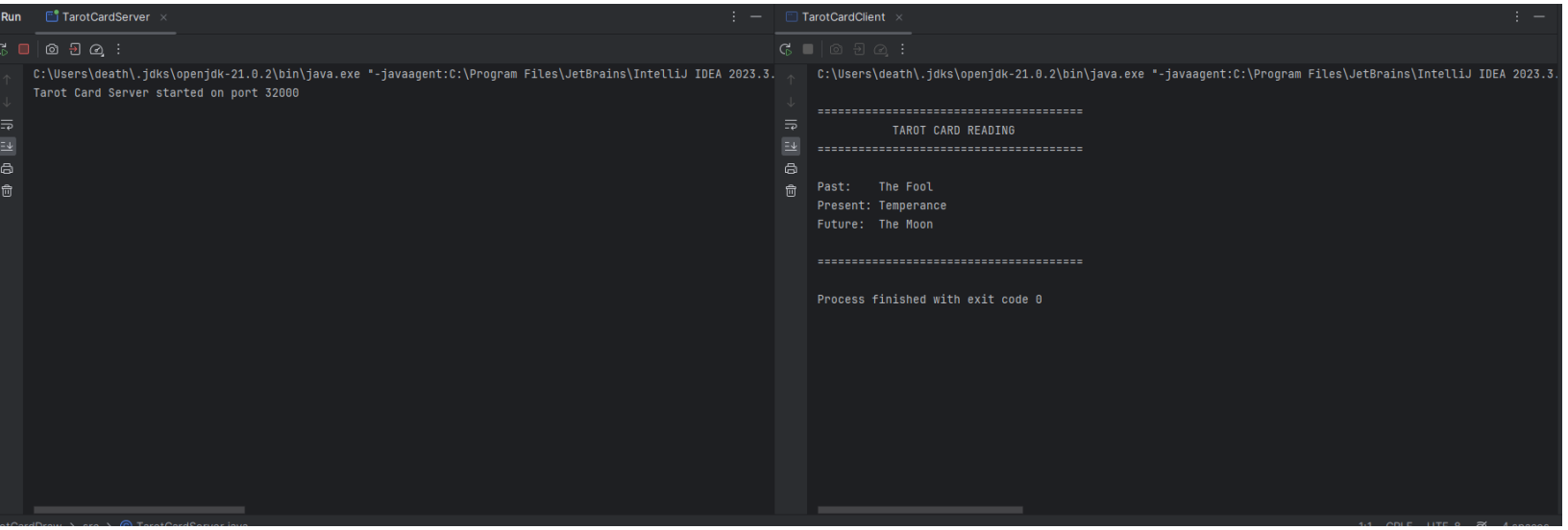
Part 1

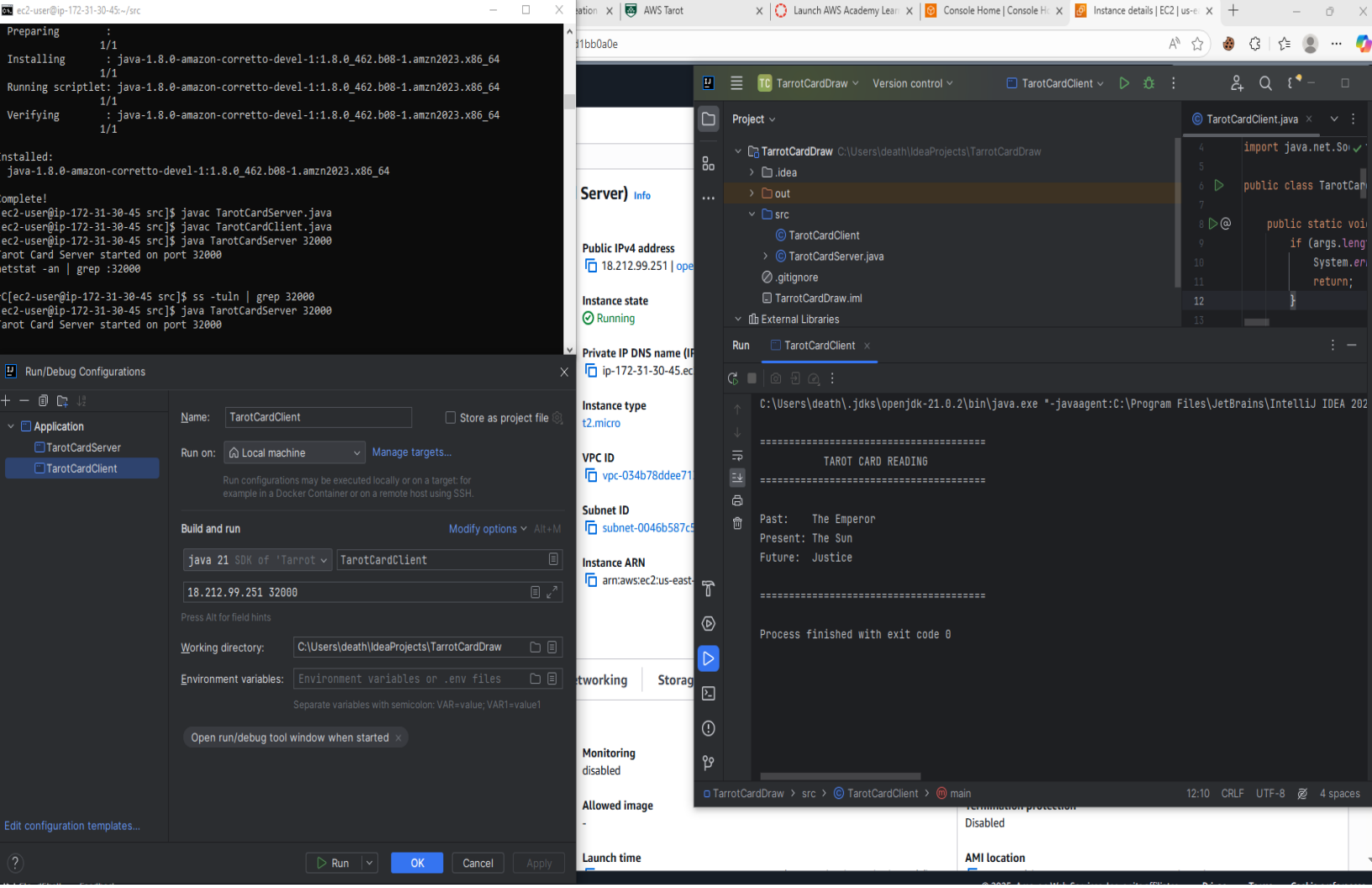
Screenshot 1:

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.



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.



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

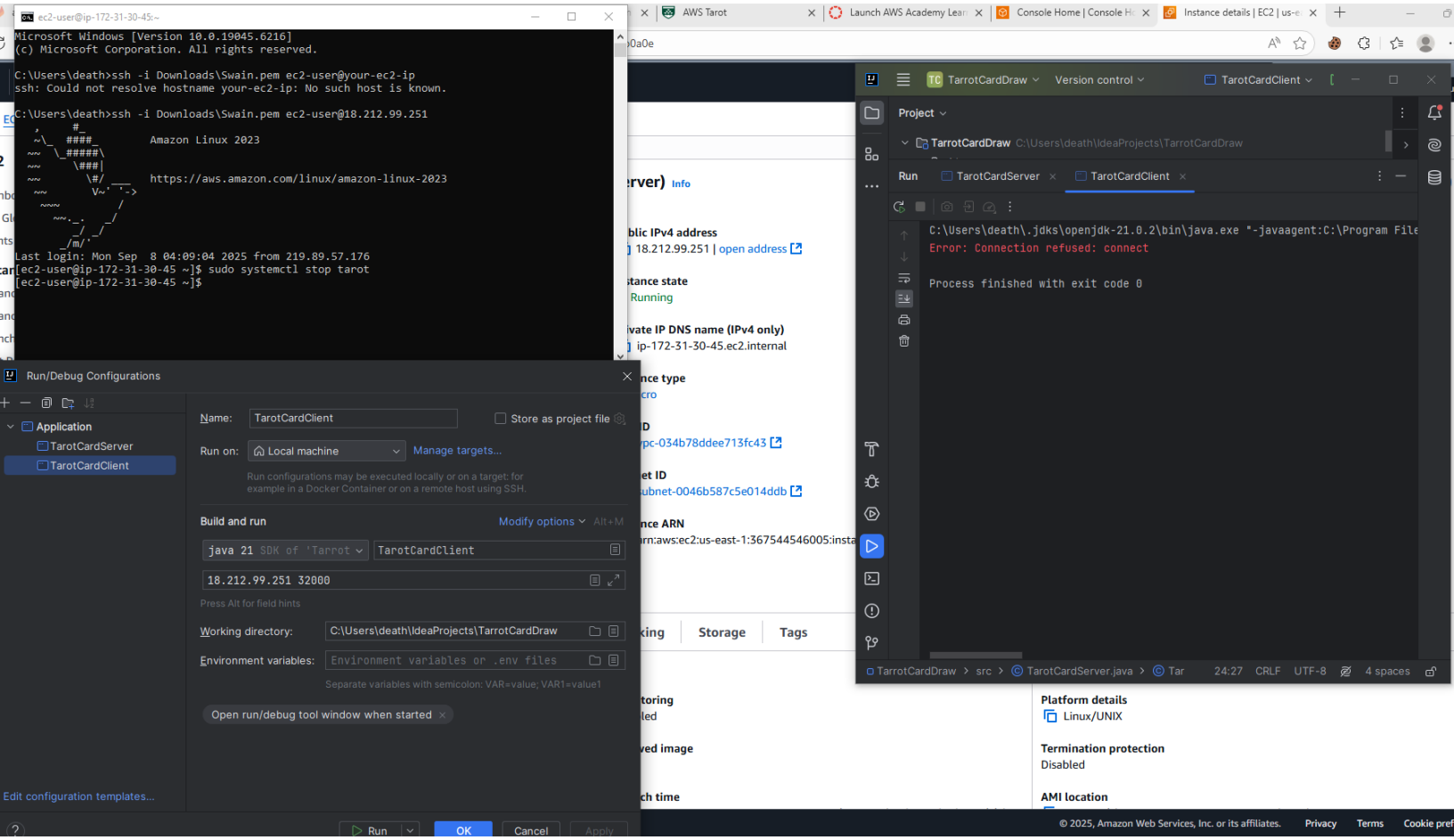
Screenshot 4:

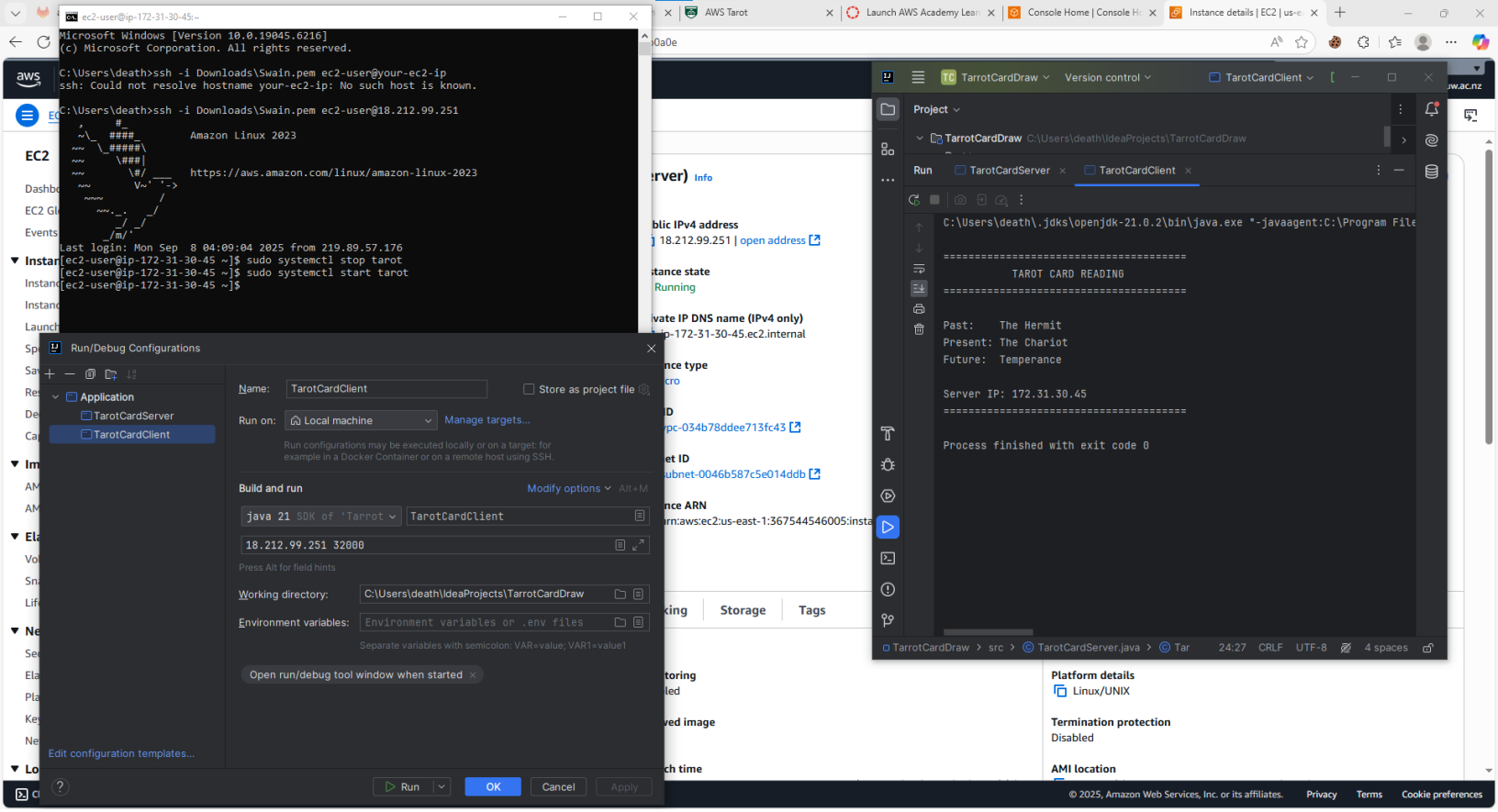
These screenshots demonstrate successful automatic startup after reboot, with Image 1 showing the systemctl status revealing the tarot service is active (running) since 06:24:28 UTC (rebooted 29 seconds ago) and system logs confirming automatic startup without manual intervention, while Images 2 and 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
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/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 ~]$
```

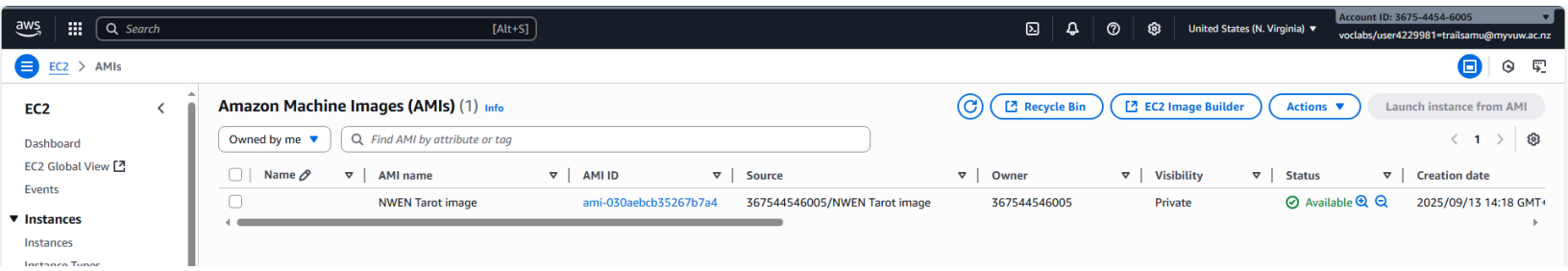
.

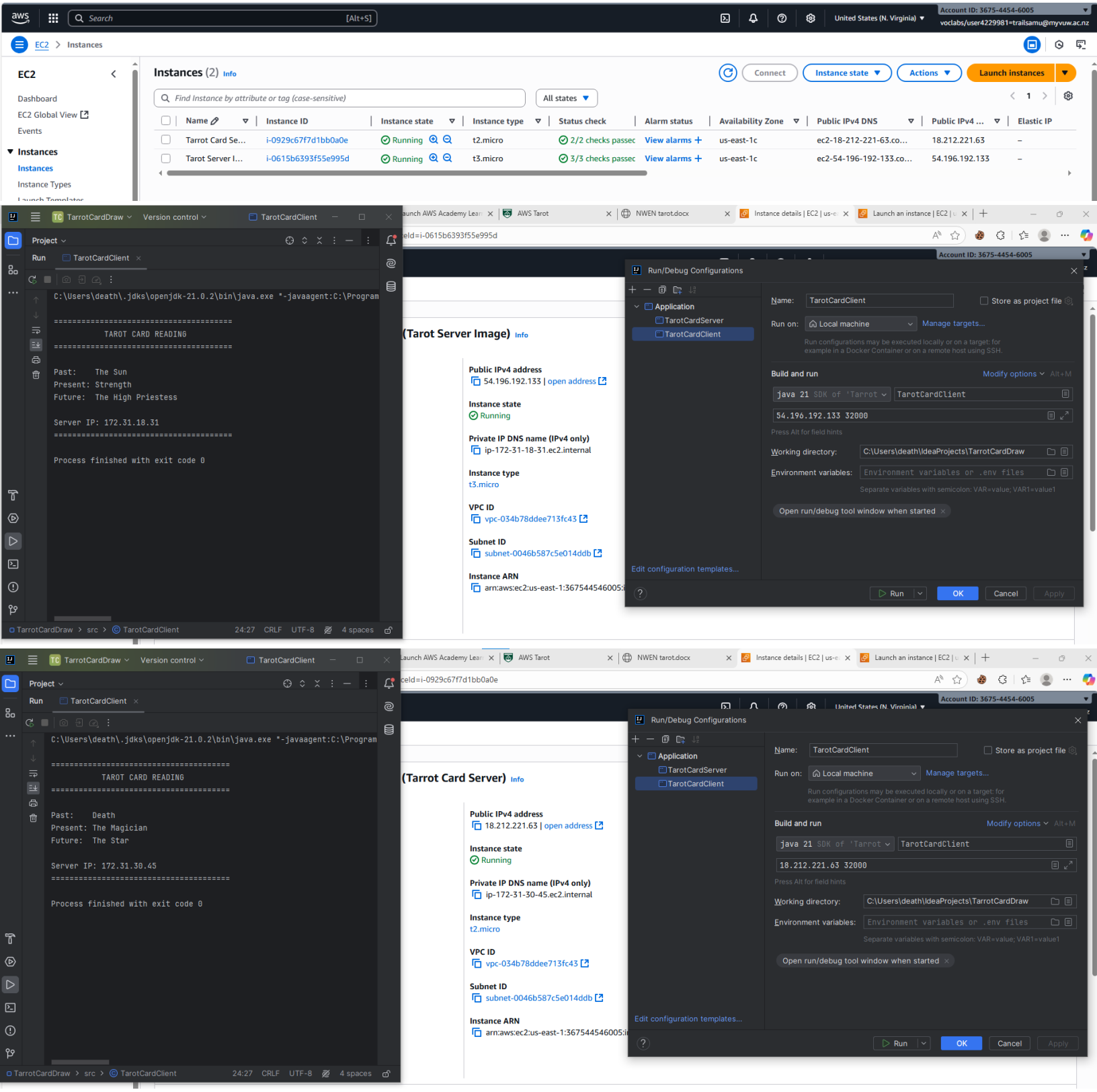




Screenshot 5:

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.

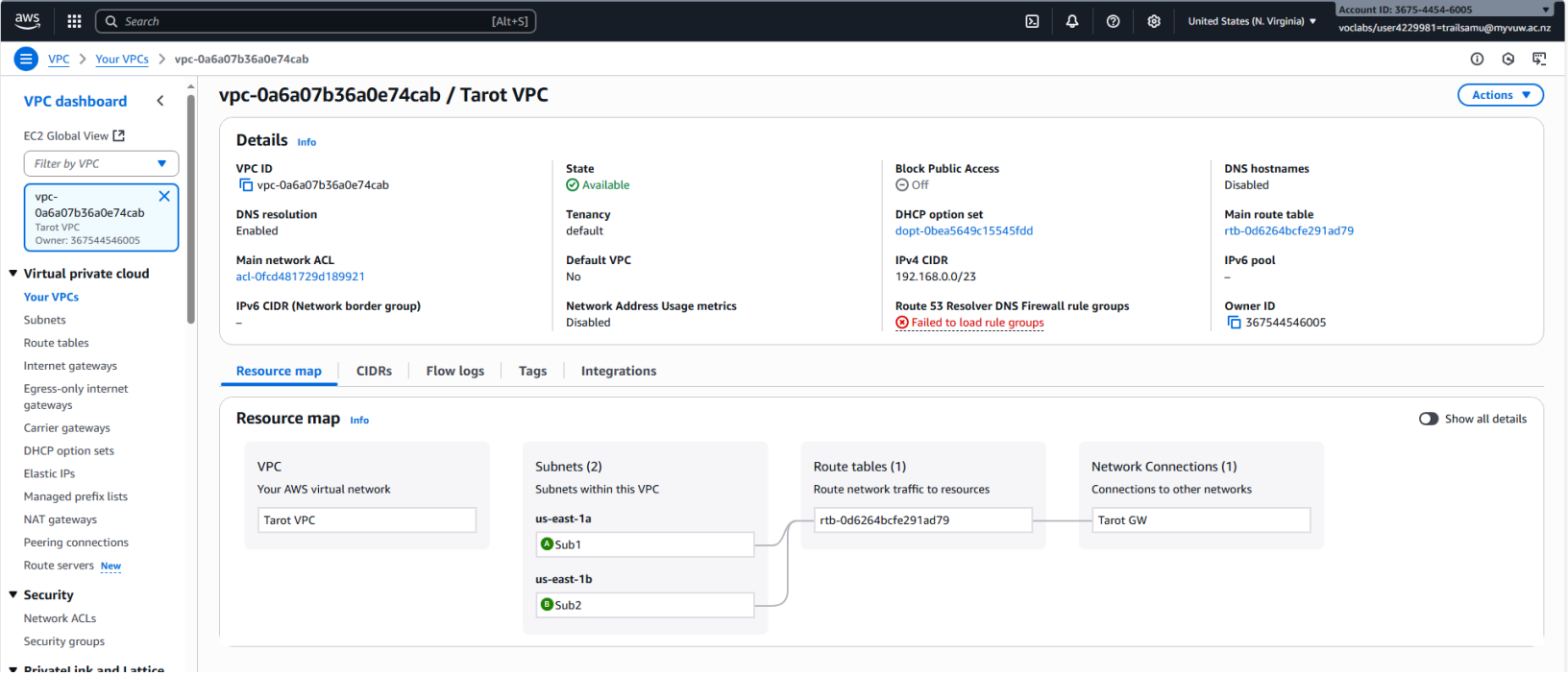




Part 2

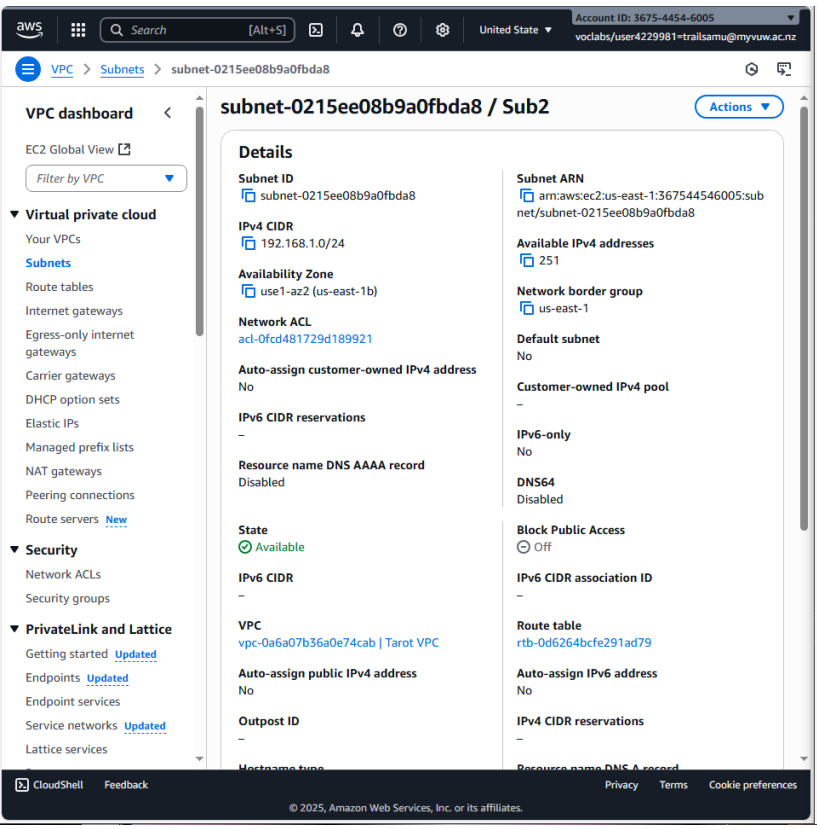
Screenshot 1

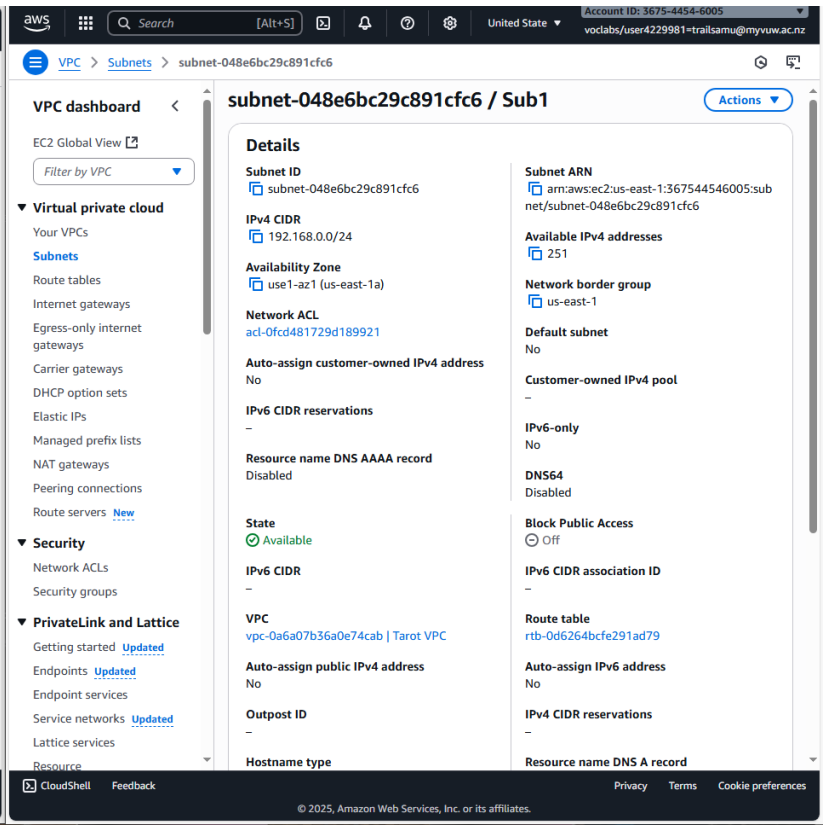
The dashboard shows the custom VPC that we made (screenshot was taken with the following steps completed)



Screenshot 2

This screenshot shows the AWS 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.

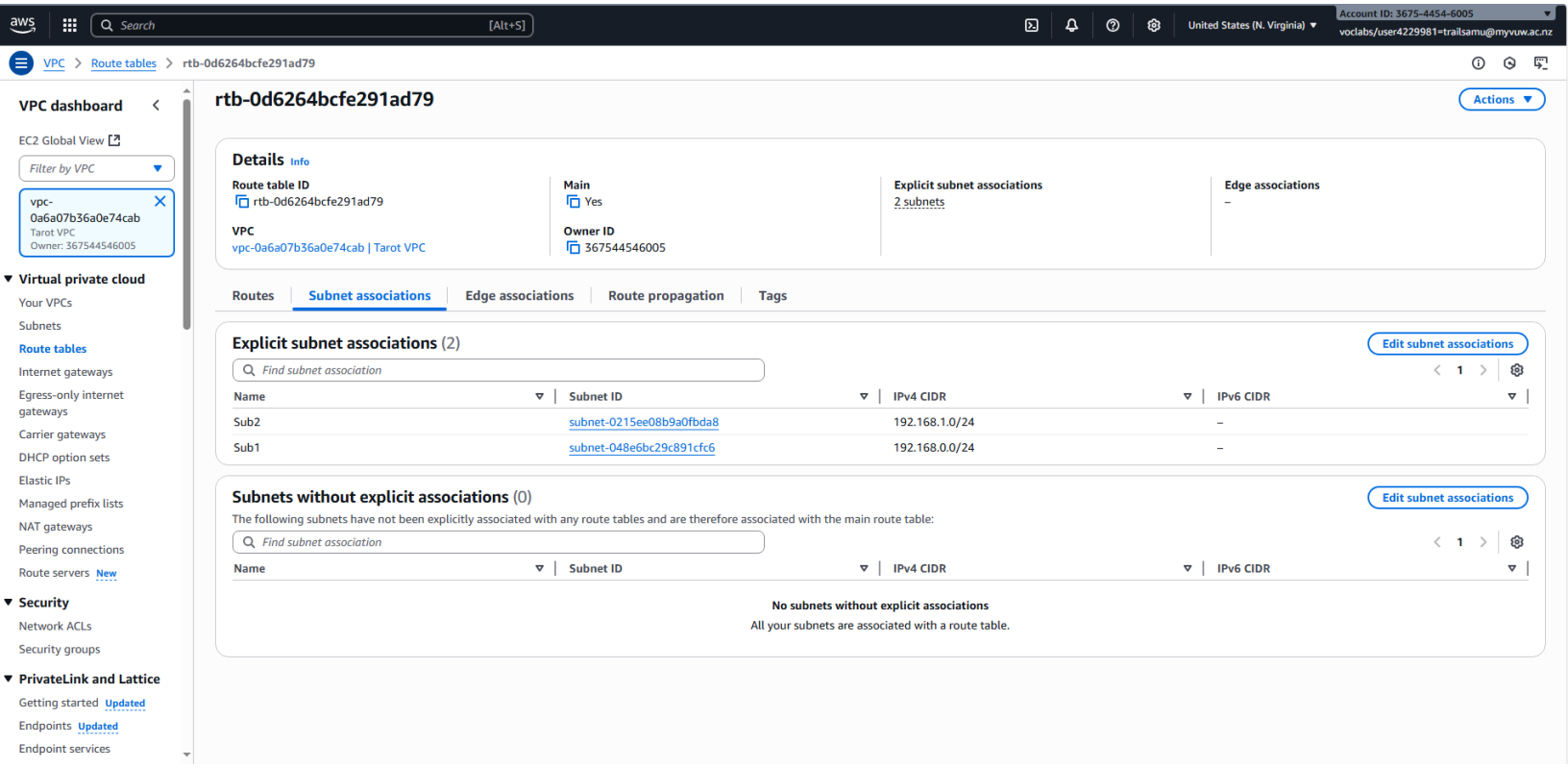


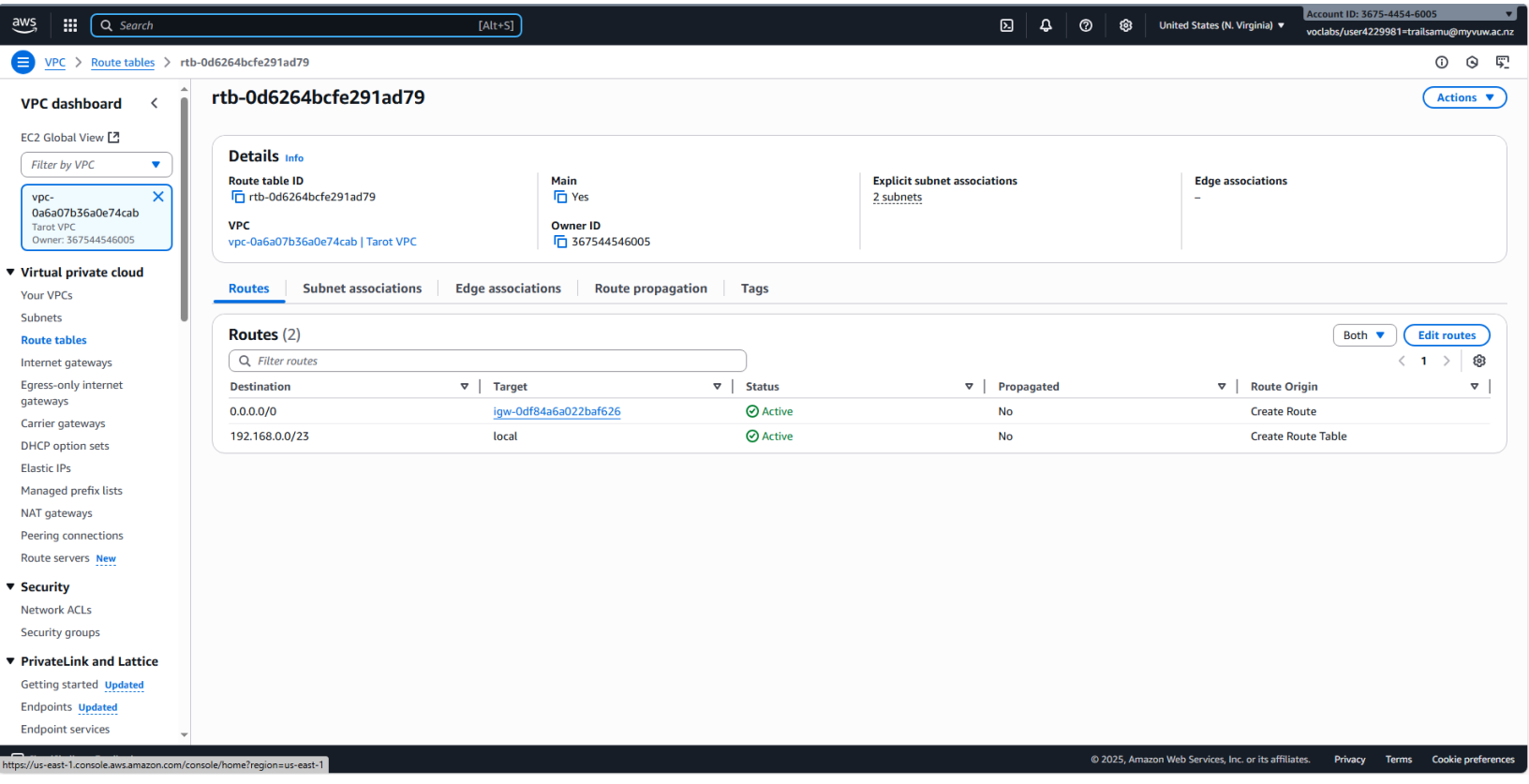


Screenshot 3,4

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.

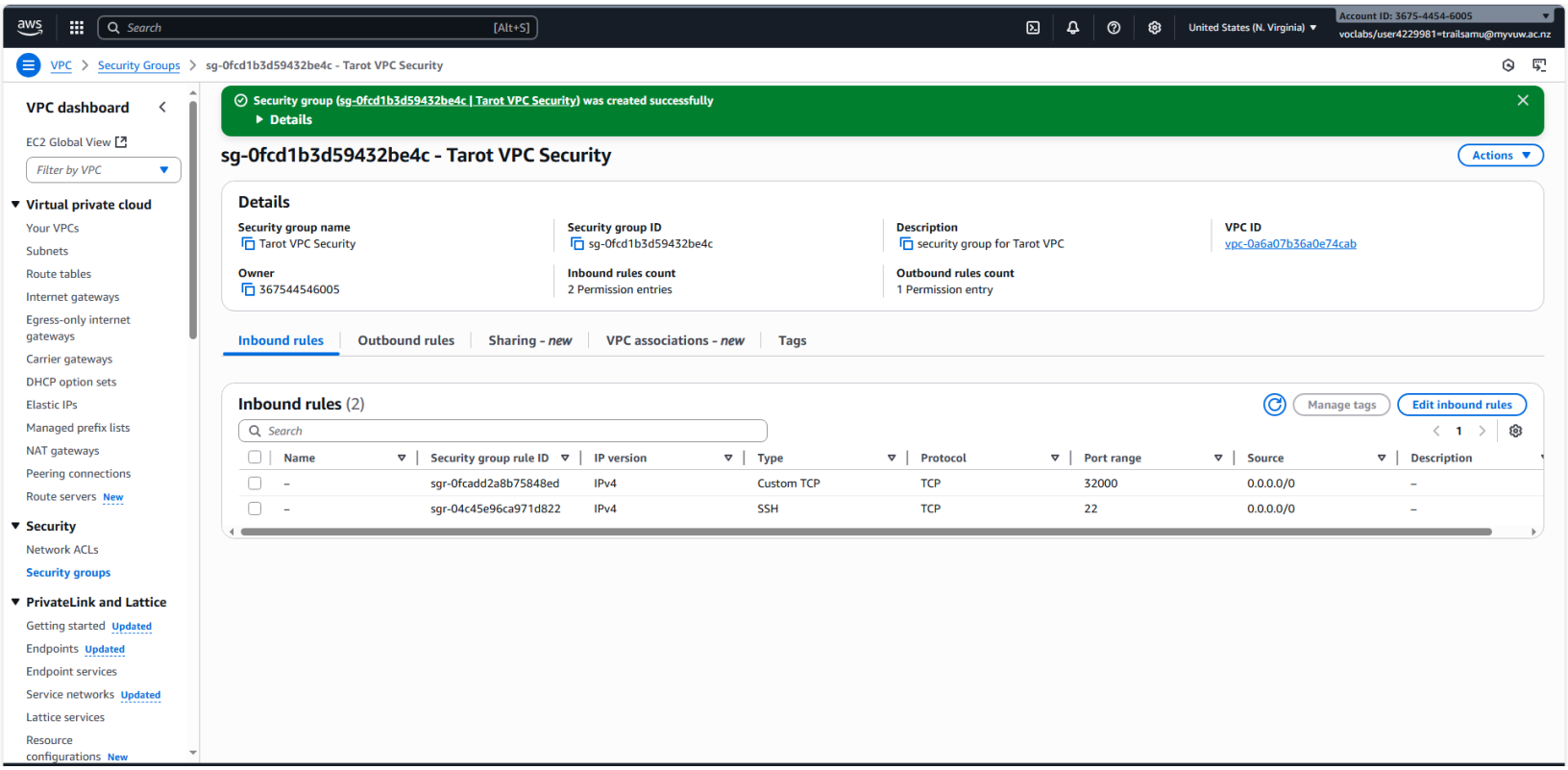
Image 4: This screenshot shows the route table's routes configuration with two entries the local route for internal VPC traffic (192.168.0.0/23) and the internet gateway route (0.0.0.0/0) that enables external internet access, completing the routing setup for the VPC.





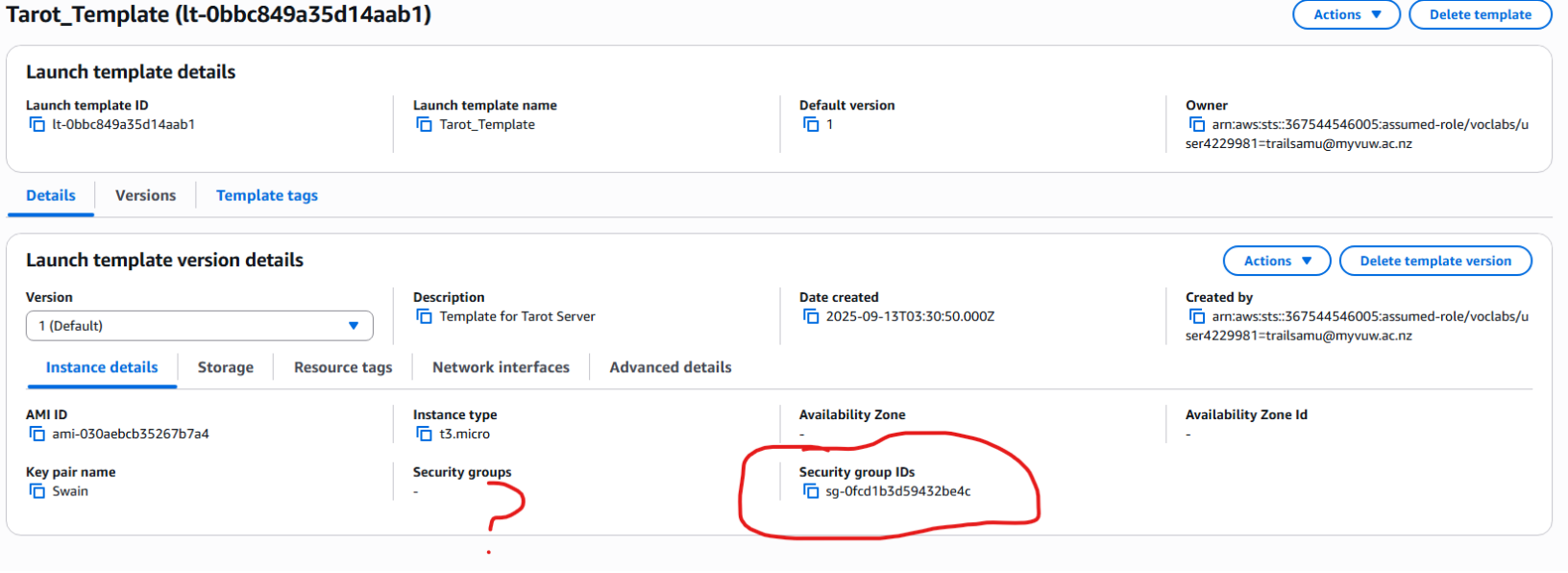
Screenshot 5

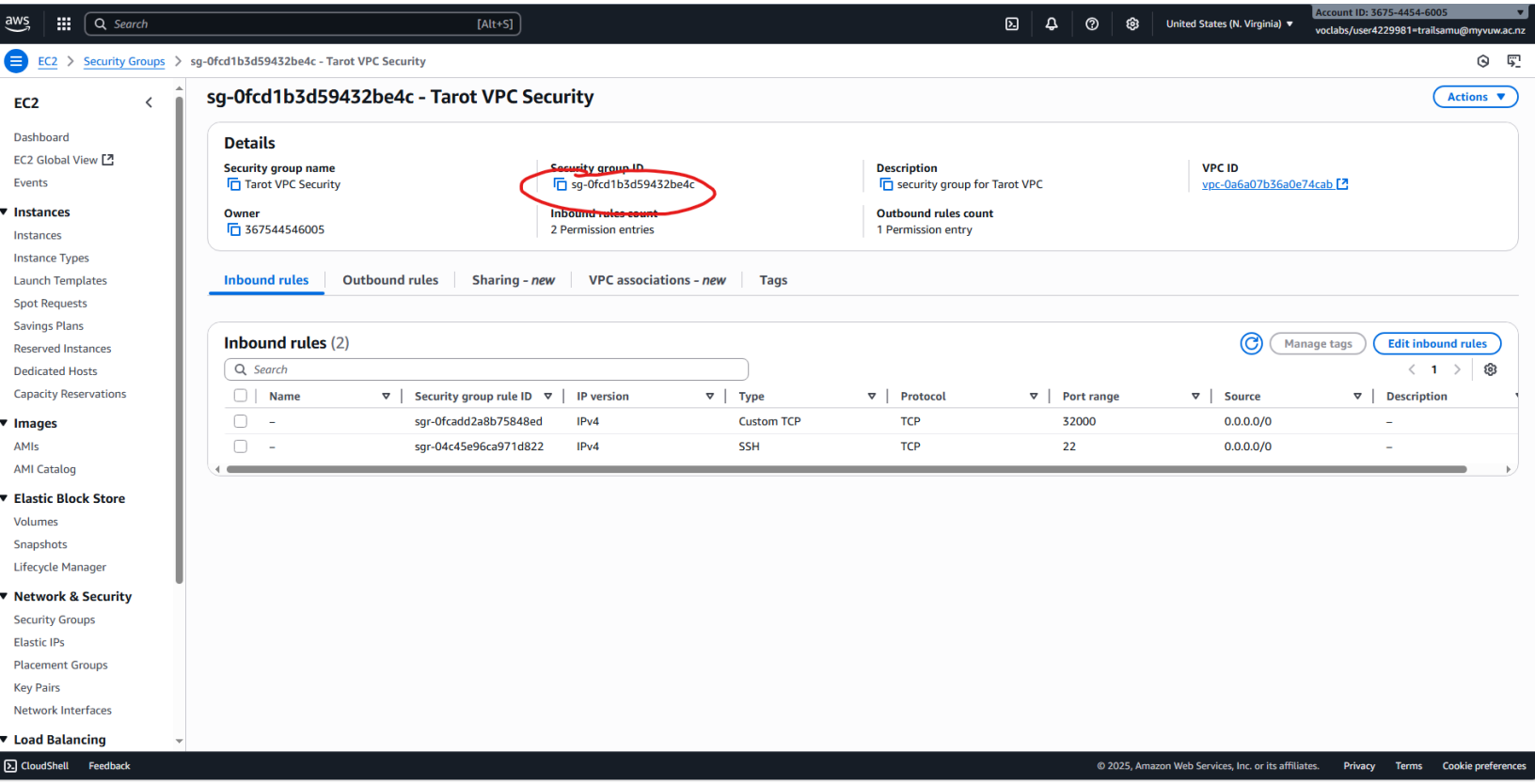
This screenshot shows the security group Tarot VPC Security with the required inbound rules configured one for SSH (port 22) and one for Custom TCP (port 32000), both allowing traffic from anywhere (0.0.0.0/0). The security group is properly associated with the custom Tarot VPC and will control access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



Screenshot 6,7

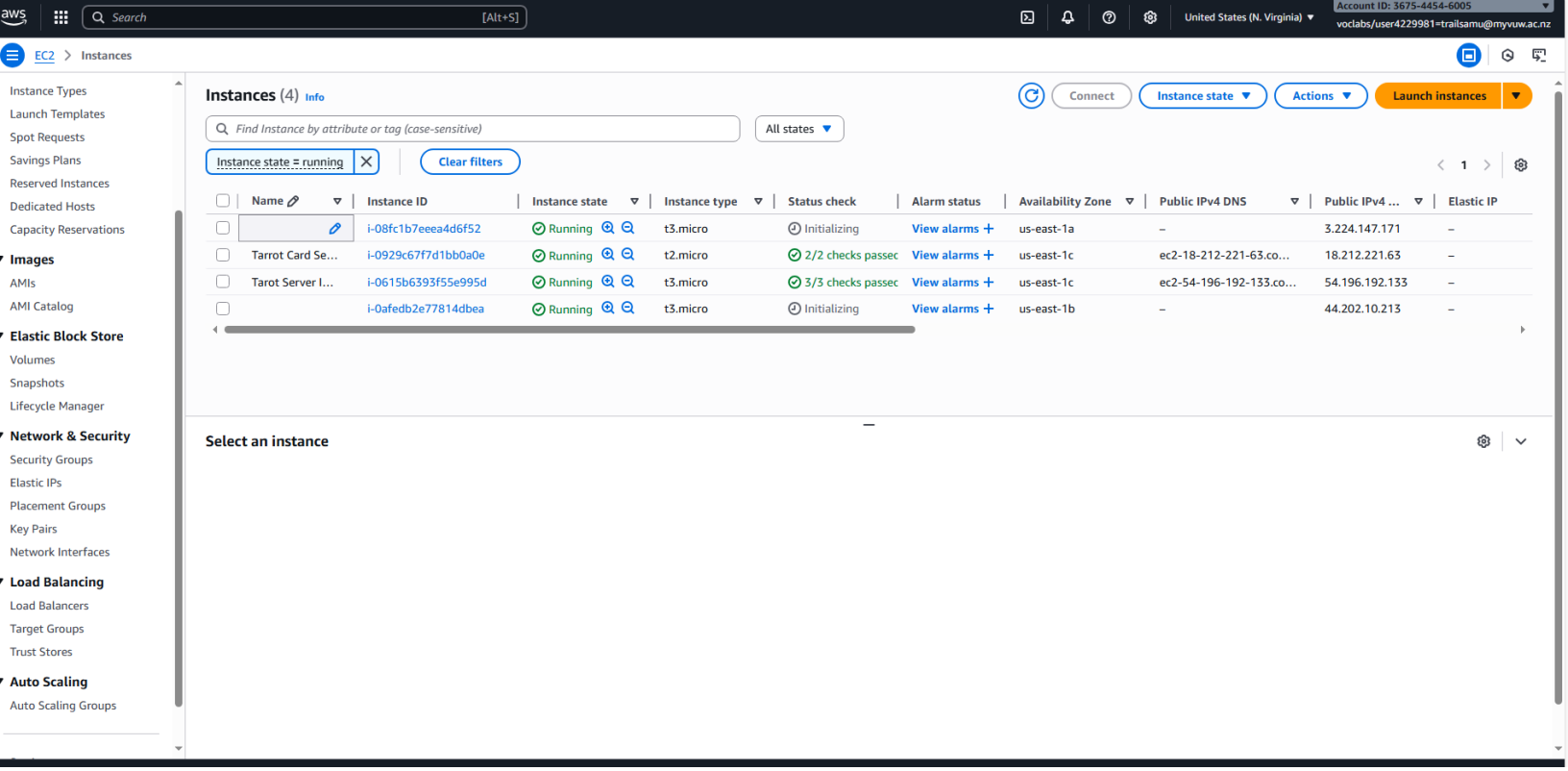
These two screenshots show the successful creation of the Tarot_Template launch template configured with the custom AMI, t3.micro instance type, and the Tarot VPC Security group. The security group ID (sg-0fcd1b3d59432be4c) matches between both images, confirming that the launch template is properly linked to the correct security group even though the name doesn't display in the template details view.

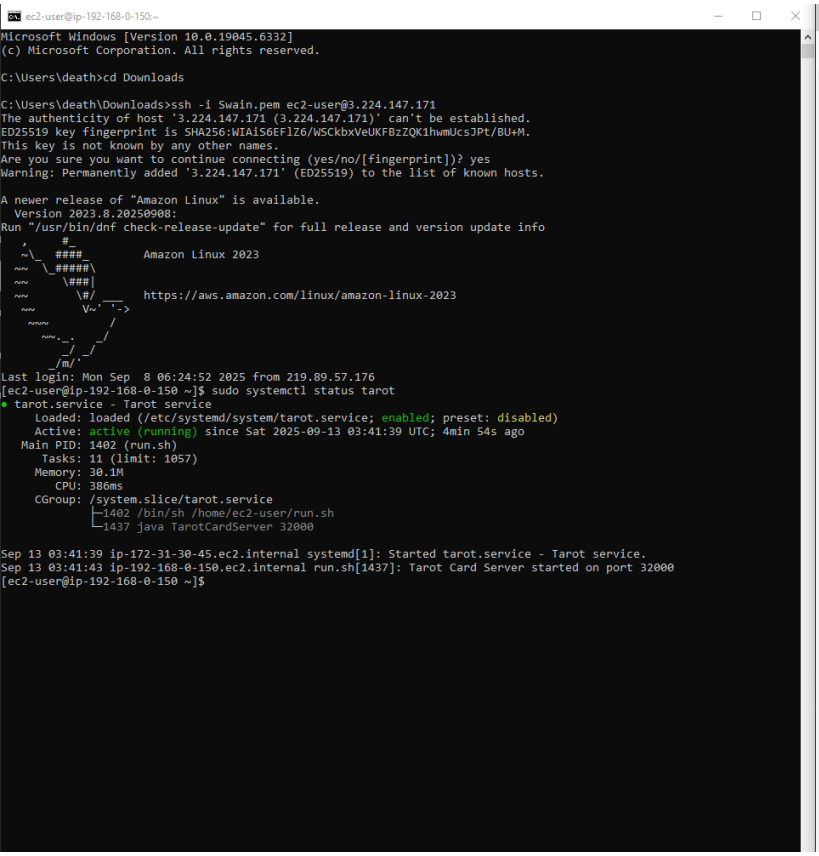


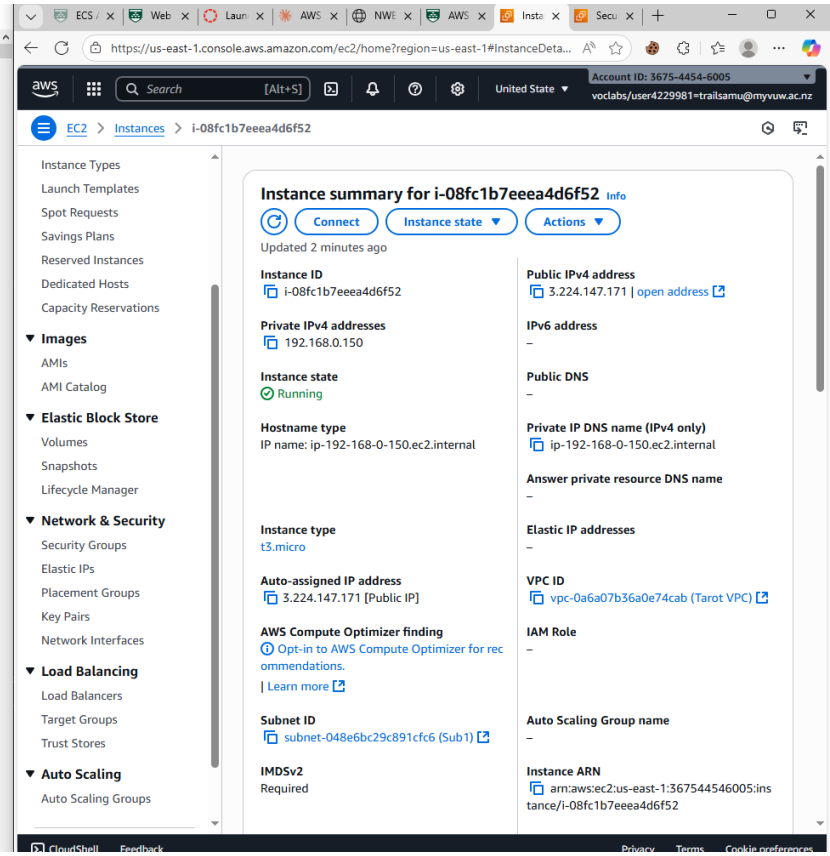


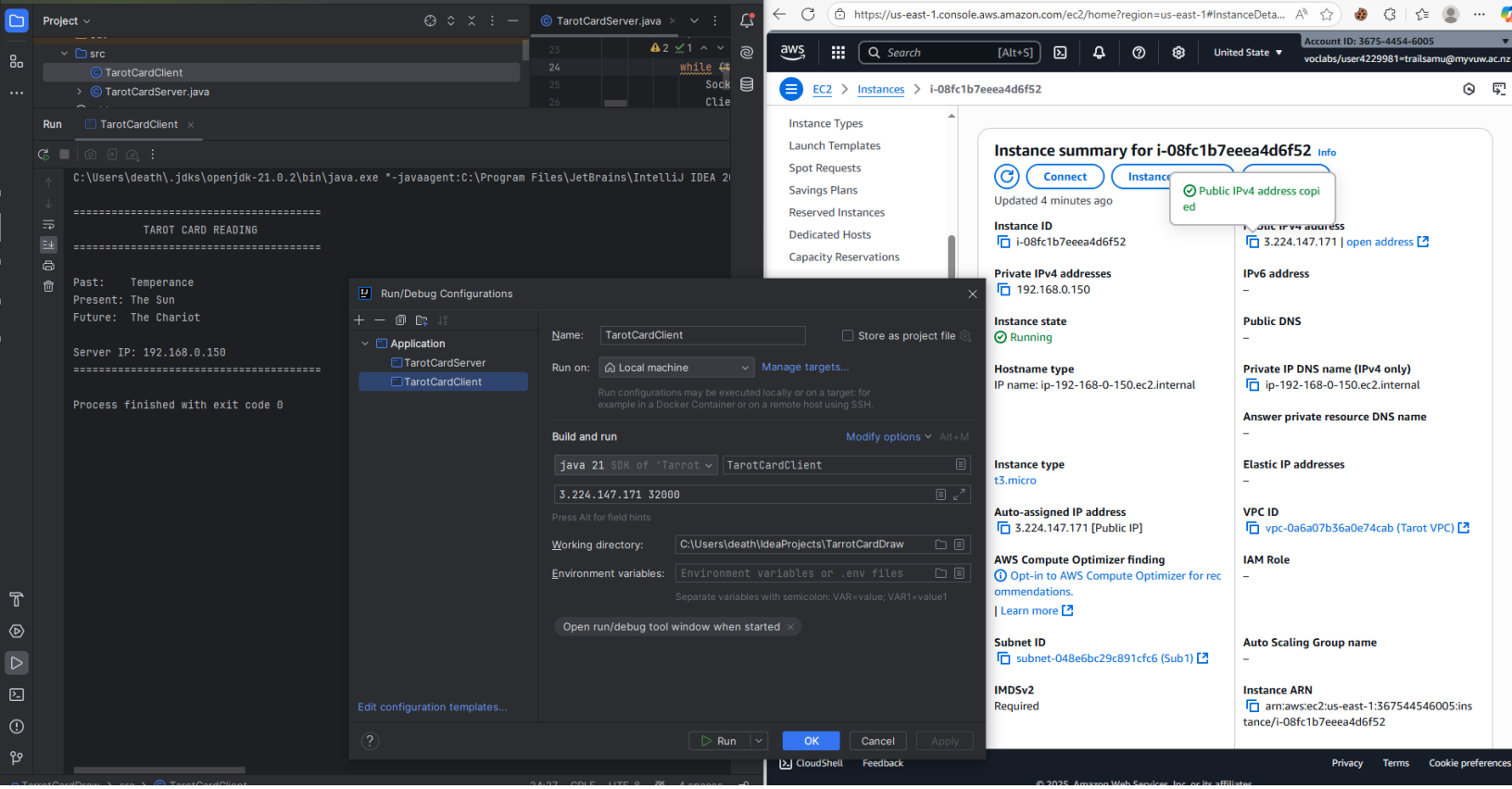
Screenshot 8,9,10,11,12

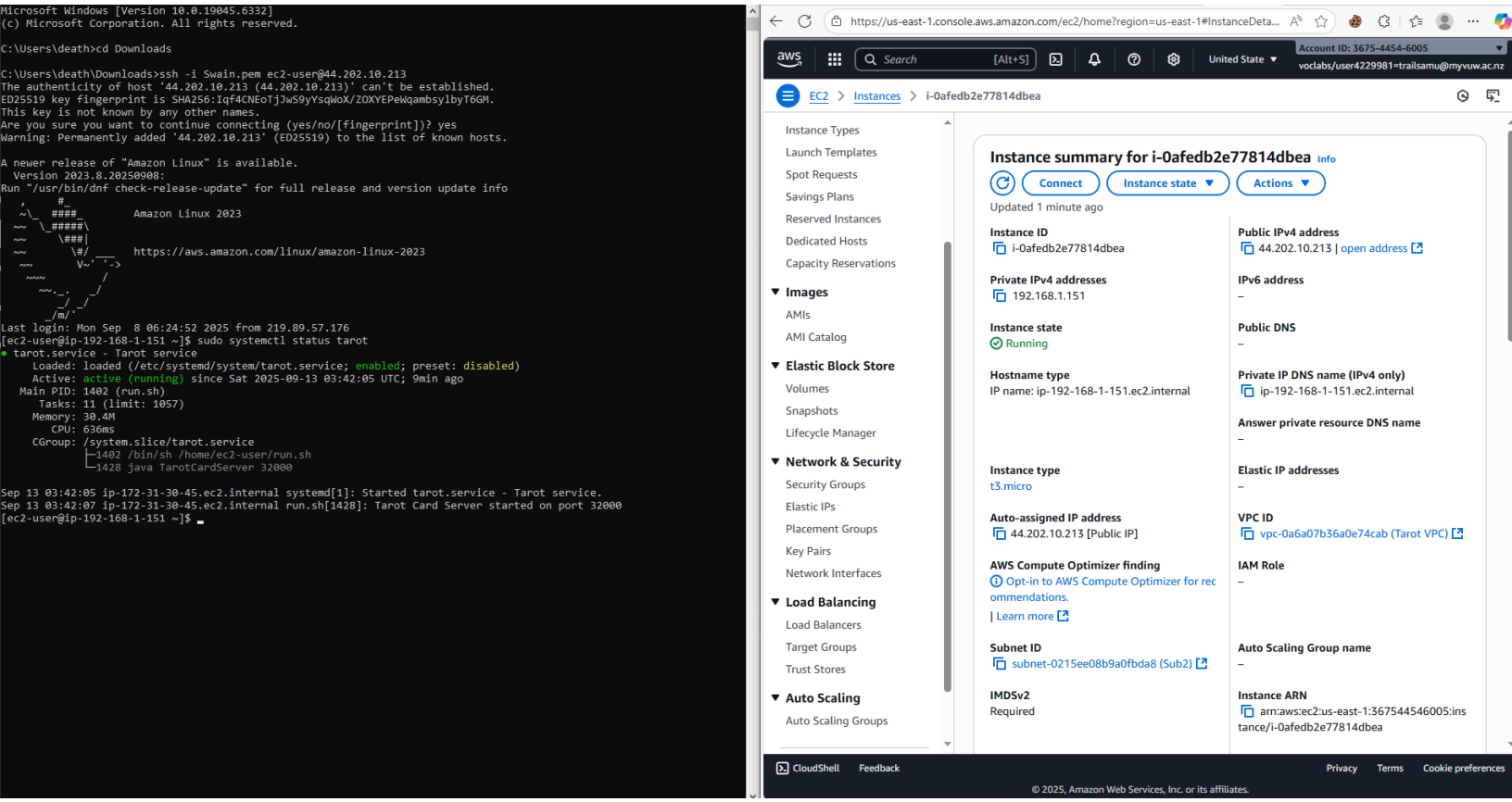
These screenshots demonstrate the successful testing phase of the launch template creation, showing the EC2 instances dashboard with four running instances (two original named instance and two unnamed instances created from the template), followed by comprehensive testing of each instance through SSH 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.


Search [Alt+S]
United States (N. Virginia)
Account ID: 3675-4454-6005
voclabs/user4229981=trailsamu@myvuw.ac.nz
EC2 > 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
Load Balancers
Target Groups
Trust Stores
Auto Scaling
Auto Scaling Groups
Instances (4) Info
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
Instance state = running
Clear filters
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
Public IPv4 DNS
Public IPv4 ...
Elastic IP
i-08fc1b7eeea4d6f52 Running t3.micro Initializing View alarms us-east-1a – 3.224.147.171 –
Tarrot Card Se... i-0929c67f7d1bb0a0e Running t2.micro 2/2 checks passec View alarms us-east-1c ec2-18-212-221-63.co... 18.212.221.63 –
Tarot Server I... i-0615b6393f55e995d Running t3.micro 3/3 checks passec View alarms us-east-1c ec2-54-196-192-133.co... 54.196.192.133 –
i-0afedb2e77814dbea Running t3.micro Initializing View alarms us-east-1b – 44.202.10.213 –
Select an instance

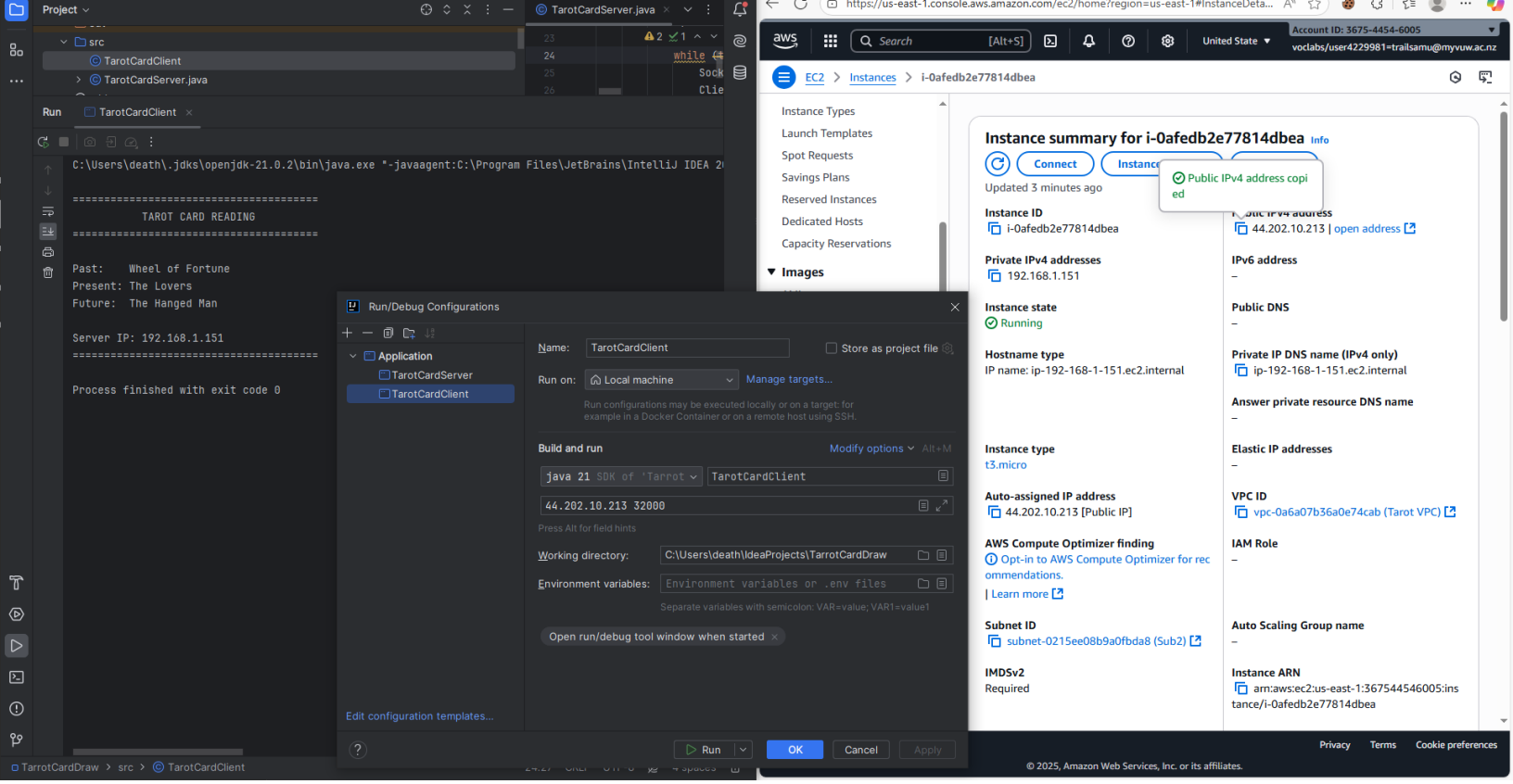

ec2-user@ip-192-168-0-150:~
Microsoft Windows [Version 10.0.19045.6332]
(c) Microsoft Corporation. All rights reserved.
C:\Users\death>cd Downloads
C:\Users\death\Downloads>ssh -i Swain.pem ec2-user@3.224.147.171
The authenticity of host '3.224.147.171 (3.224.147.171)' can't be established.
ED25519 key fingerprint is SHA256:WIAiS6EFlZ6/WSCkbxVeUKFBzZQK1hwmUcsJPt/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 - Tarot service.
Sep 13 03:41:43 ip-192-168-0-150.ec2.internal run.sh[1437]: Tarot Card Server started on port 32000
[ec2-user@ip-192-168-0-150 ~]$


https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#InstanceDeta...
Search [Alt+S]
United State
Account ID: 3675-4454-6005
voclabs/user4229981=trailsamu@myvuw.ac.nz
EC2 > Instances > i-08fc1b7eeea4d6f52
Instance summary for i-08fc1b7eeea4d6f52 Info
Connect
Instance state
Actions
Updated 2 minutes ago
Instance ID
i-08fc1b7eeea4d6f52
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
Answer private resource DNS name
–
Instance type
t3.micro
Elastic IP addresses
–
Auto-assigned IP address
3.224.147.171 [Public IP]
VPC ID
vpc-0a6a07b36a0e74cab (Tarot VPC)
AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations.
Learn more
IAM Role
–
Subnet ID
subnet-048e6bc29c891cfc6 (Sub1)
Auto Scaling Group name
–
IMDSv2
Required
Instance ARN
arn:aws:ec2:us-east-1:367544546005:instance/i-08fc1b7eeea4d6f52
CloudShell
Feedback
Privacy
Terms
Cookie preferences


Project
src
TarotCardClient
TarotCardServer.java
Run
TarotCardClient
C:\Users\death\.jdks\openjdk-21.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2
TAROT CARD READING
Past: Temperance
Present: The Sun
Future: The Chariot
Server IP: 192.168.0.150
Process finished with exit code 0
Run/Debug Configurations
Application
TarotCardServer
TarotCardClient
Name: TarotCardClient
Store as project file
Run on: Local machine
Manage targets...
Build and run
Modify options
java 21 SDK of 'Tarrot
TarotCardClient
3.224.147.171 32000
Working directory: C:\Users\death\IdeaProjects\TarrotCardDraw
Environment variables:
Open run/debug tool window when started
Edit configuration templates...
Run
OK
Cancel
Apply
Instance summary for i-08fc1b7eeea4d6f52
Connect
Public IPv4 address copied
Updated 4 minutes ago
Instance ID
i-08fc1b7eeea4d6f52
Private IPv4 addresses
192.168.0.150
Instance state
Running
Hostname type
IP name: ip-192-168-0-150.ec2.internal
Instance type
t3.micro
Auto-assigned IP address
3.224.147.171 [Public IP]
Subnet ID
subnet-048e6bc29c891cfc6 (Sub1)
IMDSv2
Required
3.224.147.171 | open address
IPv6 address
Public DNS
Private IP DNS name (IPv4 only)
ip-192-168-0-150.ec2.internal
Answer private resource DNS name
Elastic IP addresses
VPC ID
vpc-0a6a07b36a0e74cab (Tarot VPC)
IAM Role
Auto Scaling Group name
Instance ARN
arn:aws:ec2:us-east-1:367544546005:instance/i-08fc1b7eeea4d6f52
AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations.
Learn more
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
CloudShell
Feedback
Privacy
Terms
Cookie preferences

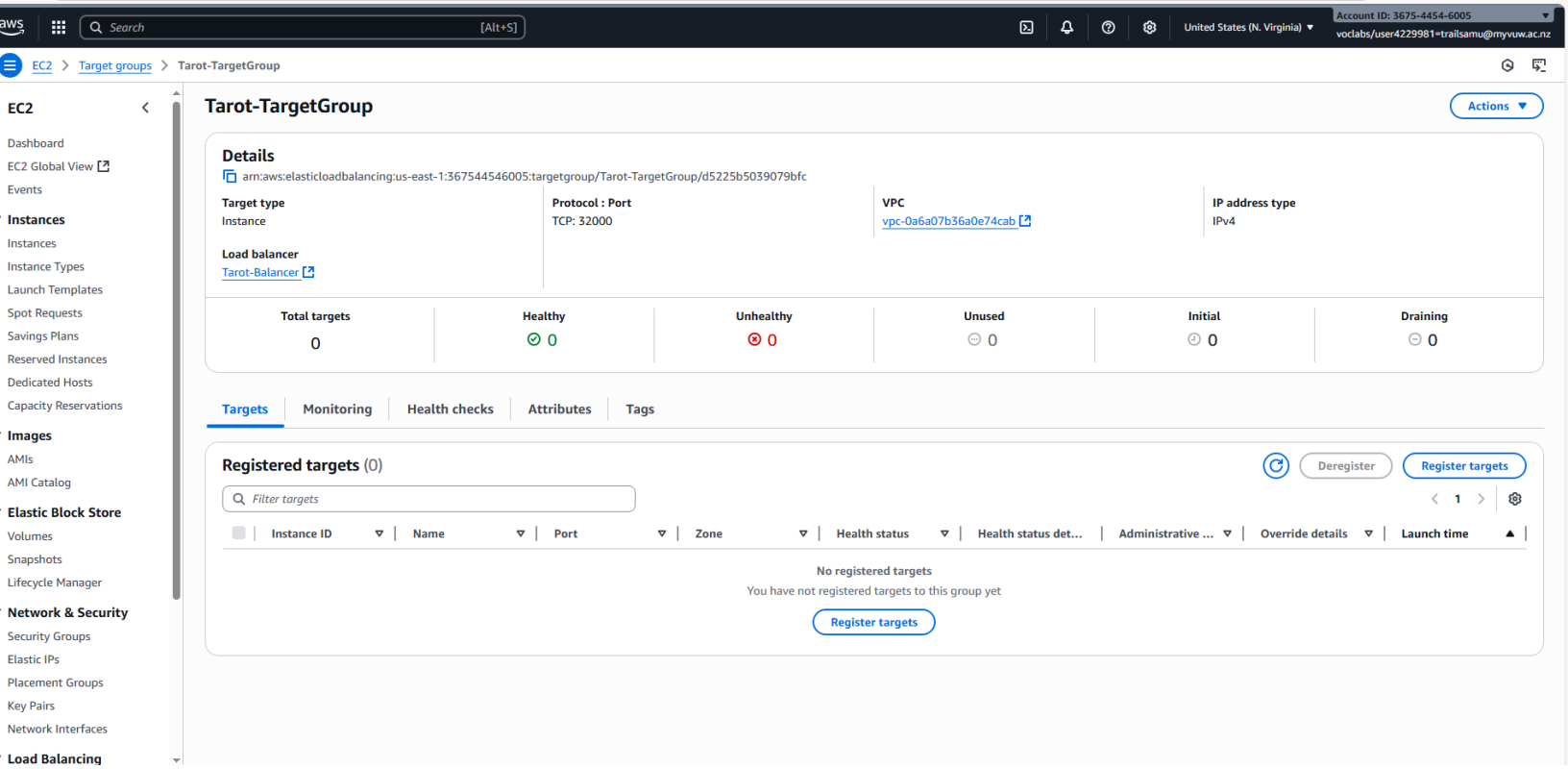

Microsoft Windows [Version 10.0.19045.6332]
(c) Microsoft Corporation. All rights reserved.
C:\Users\death>cd Downloads
C:\Users\death\Downloads>ssh -i Swain.pem ec2-user@44.202.10.213
The authenticity of host '44.202.10.213 (44.202.10.213)' can't be established.
ED25519 key fingerprint is SHA256:Iqf4CNEoTjJwS9yYsqWoX/ZOXYEPeWqambsylbyT6GM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '44.202.10.213' (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-192-168-1-151 ~]$ 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 03:42:05 UTC; 9min ago
Main PID: 1402 (run.sh)
Tasks: 11 (limit: 1057)
Memory: 30.4M
CPU: 636ms
CGroup: /system.slice/tarot.service
├─1402 /bin/sh /home/ec2-user/run.sh
└─1428 java TarotCardServer 32000
Sep 13 03:42:05 ip-172-31-30-45.ec2.internal systemd[1]: Started tarot.service - Tarot service.
Sep 13 03:42:07 ip-172-31-30-45.ec2.internal run.sh[1428]: Tarot Card Server started on port 32000
[ec2-user@ip-192-168-1-151 ~]$

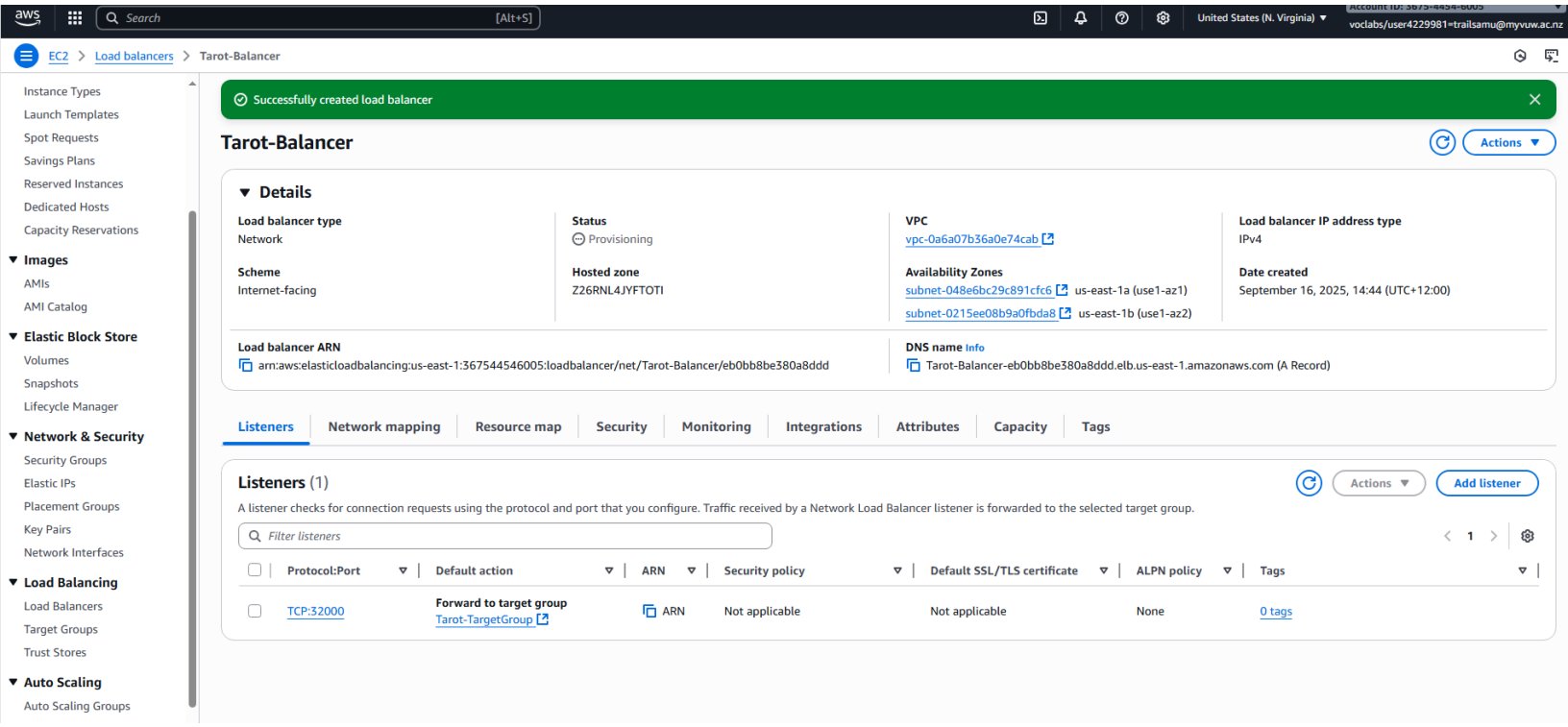
Instance summary for i-0afedb2e77814dbea
Connect
Instance state
Actions
Updated 1 minute ago
Instance ID
i-0afedb2e77814dbea
Private IPv4 addresses
192.168.1.151
Instance state
Running
Hostname type
IP name: ip-192-168-1-151.ec2.internal
Instance type
t3.micro
Auto-assigned IP address
44.202.10.213 [Public IP]
Subnet ID
subnet-0215ee08b9a0fbda8 (Sub2)
IMDSv2
Required
Public IPv4 address
44.202.10.213 | open address
IPv6 address
Public DNS
Private IP DNS name (IPv4 only)
ip-192-168-1-151.ec2.internal
Answer private resource DNS name
Elastic IP addresses
VPC ID
vpc-0a6a07b36a0e74cab (Tarot VPC)
IAM Role
Auto Scaling Group name
Instance ARN
arn:aws:ec2:us-east-1:367544546005:instance/i-0afedb2e77814dbea
AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations.
Learn more
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
CloudShell
Feedback
Privacy
Terms
Cookie preferences
© 2025, Amazon Web Services, Inc. or its affiliates.



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.





Screenshot 15,16,17,18

Auto scaling group settings

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

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 — Traffic to your Auto Scaling group will not be fronted by a load balancer.
- Attach to an existing load balancer — Choose from your existing load balancers.
- 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 not 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

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

Elastic Load Balancing monitors whether instances are available to handle requests. When it reports an unhealthy instance, EC2 Auto Scaling can replace it on its next periodic check.

Step 1
Choose launch template
Step 2
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 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

Override launch template

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_Template lt-0bbc849a35d14aab1	2	-

Instance type
t3.micro

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-0a6a07b36a0e74cab (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-048c6bc29c891cfc6 (Sub1)
192.168.0.0/24

use1-az2 (us-east-1b) | subnet-0215ee08b9a0fbda8 (Sub2)
192.168.1.0/24

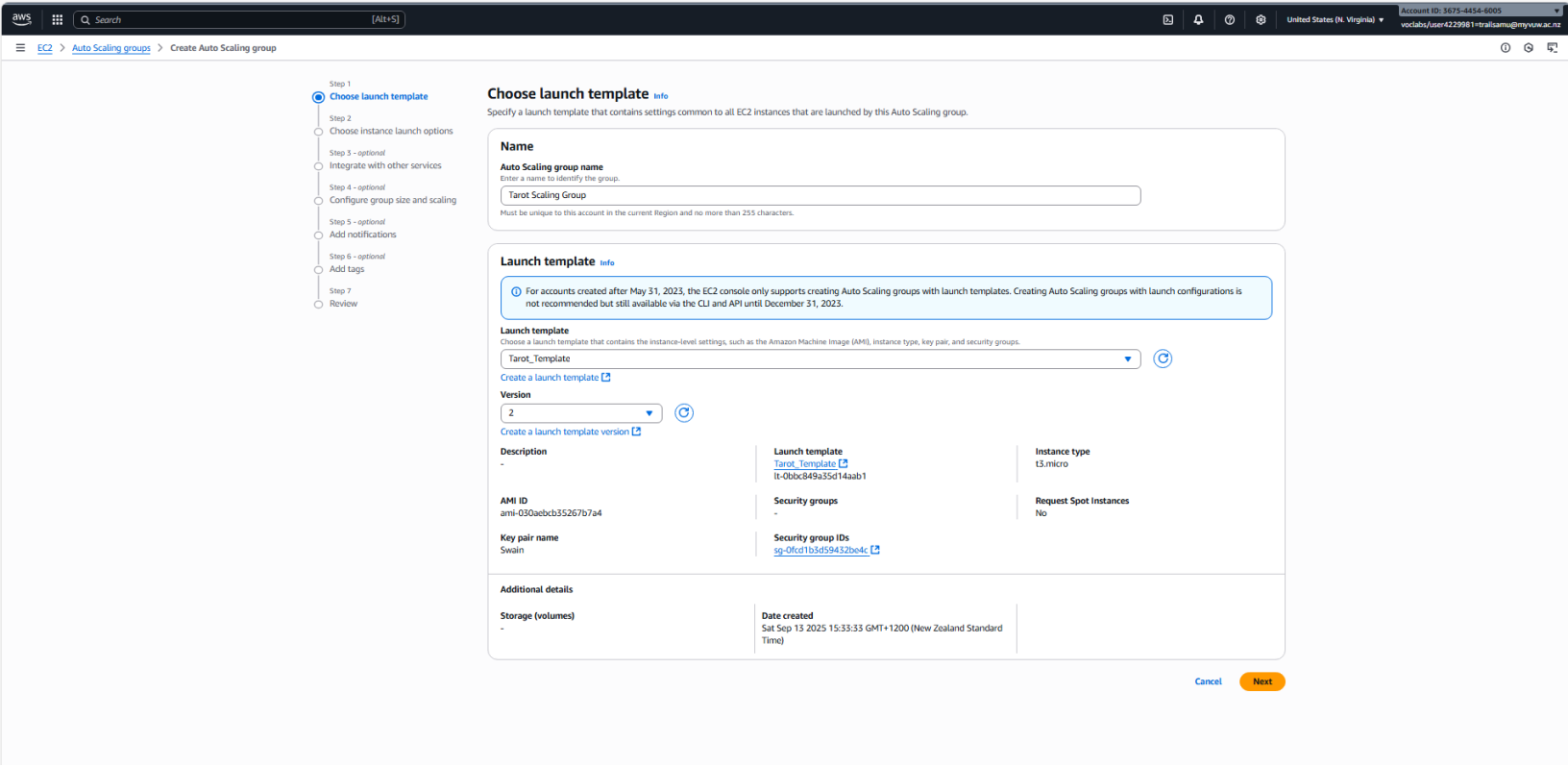
Create a subnet

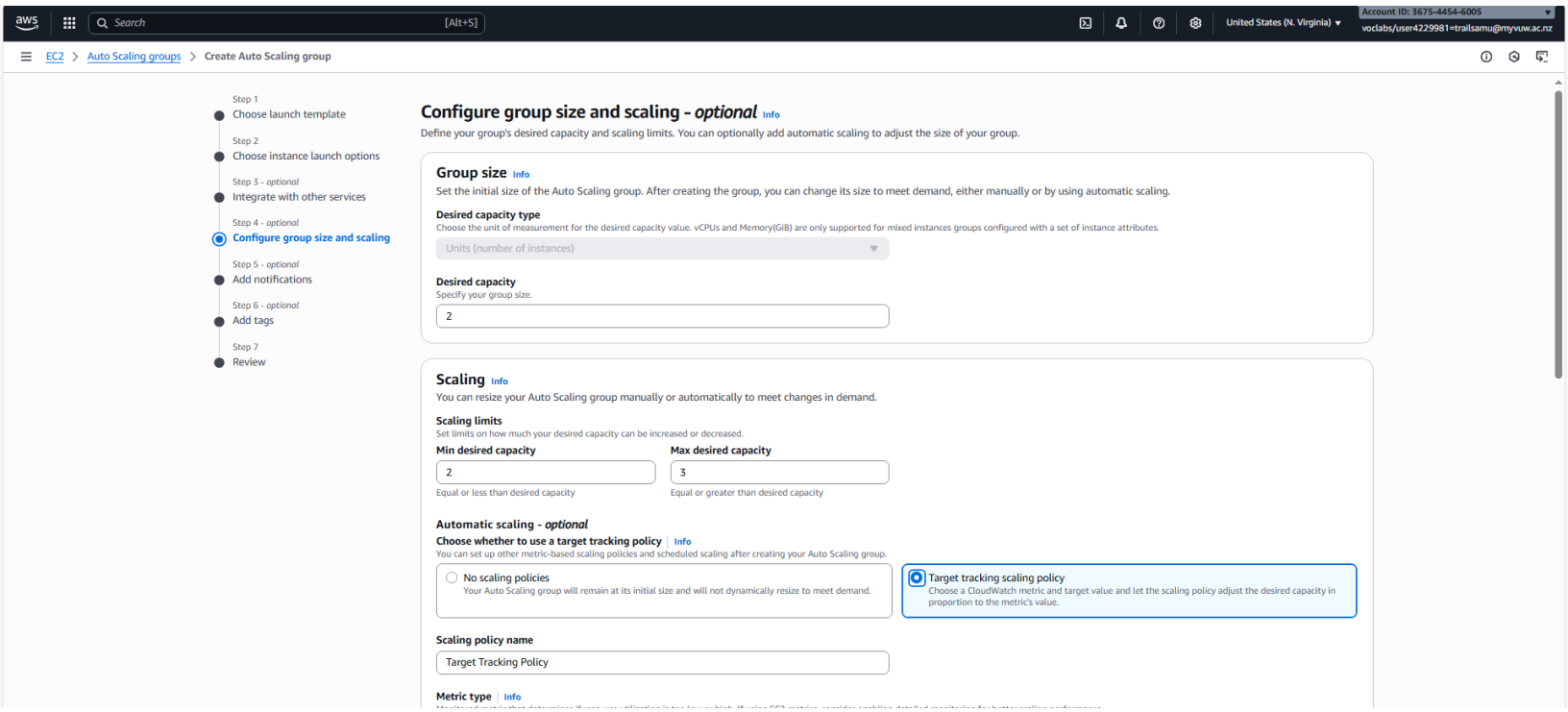
Availability Zone distribution - *new*

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

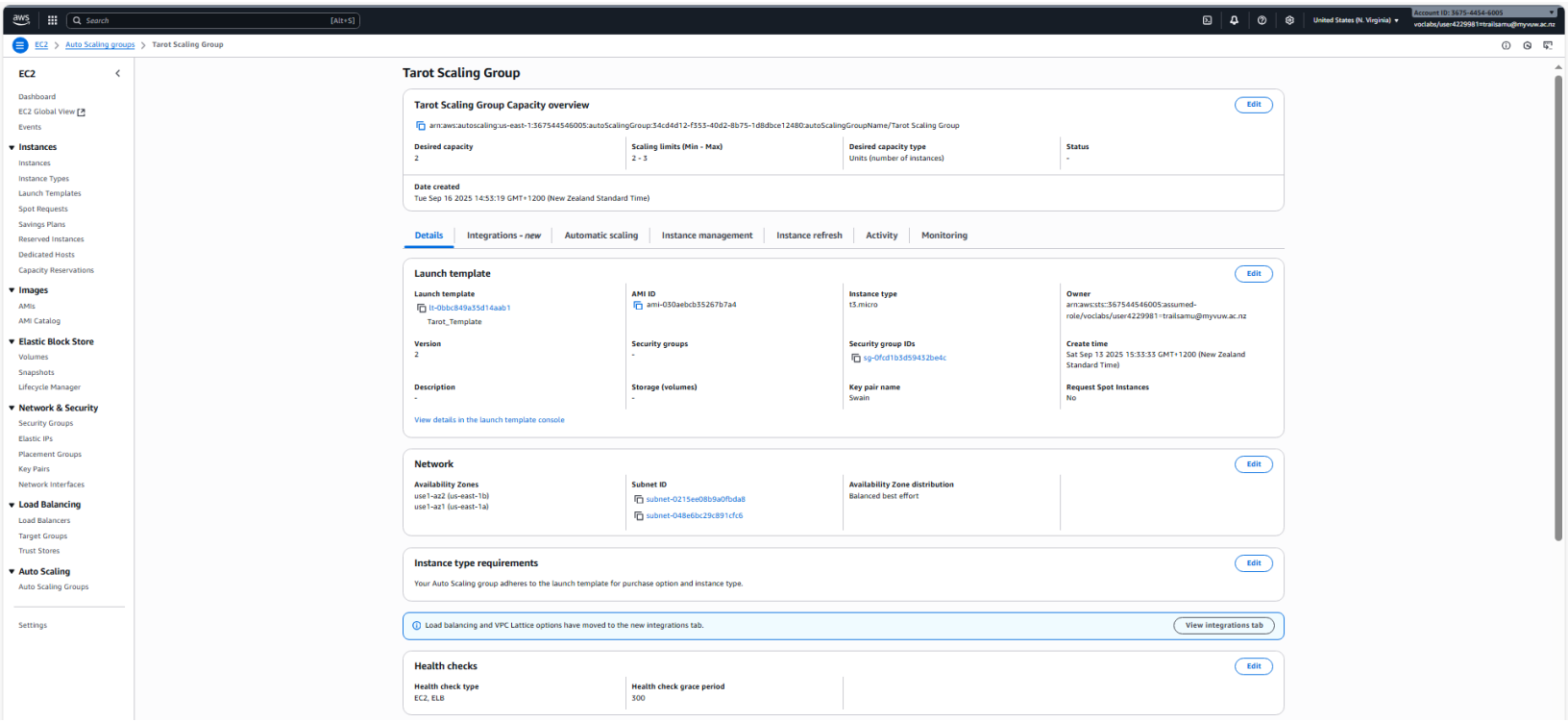




Screenshot 19

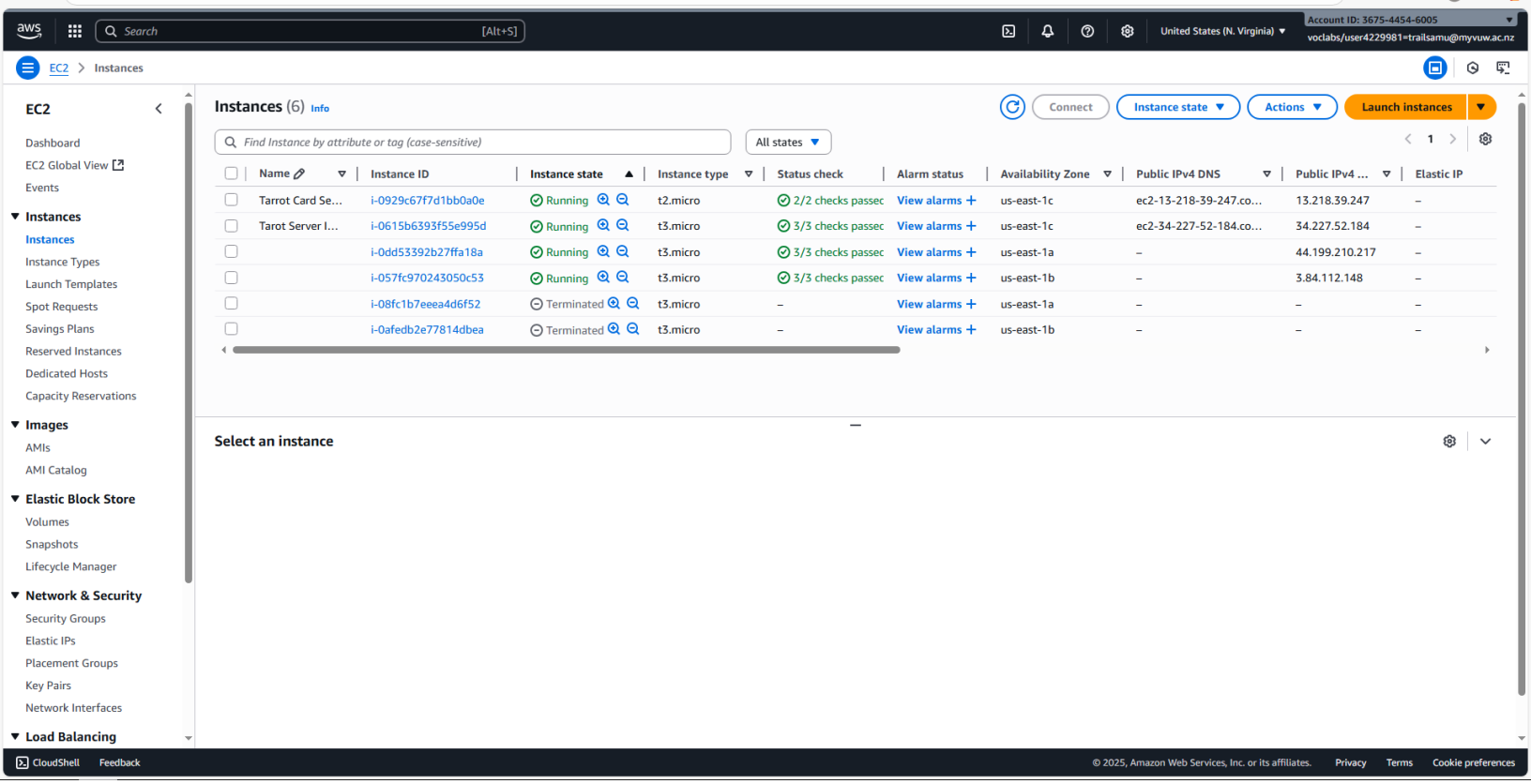
This screenshot shows the Tarot Scaling Group configuration page, displaying the setup including the launch template selection (Tarot_Template), network configuration with the custom VPC and both subnets across different availability zones, load balancer settings, instance type requirements, and health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

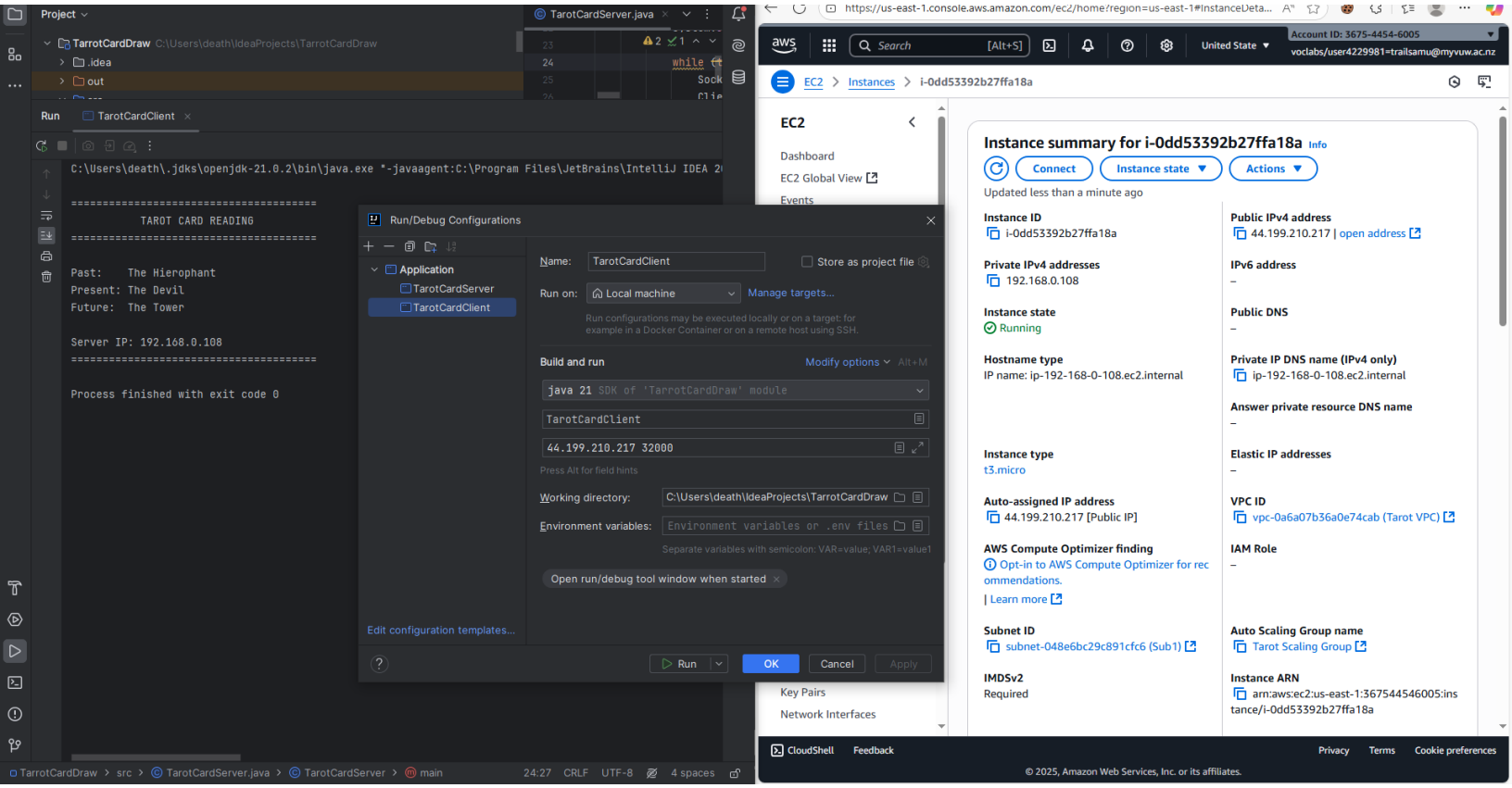
the key components properly configured for the auto scaling group that will automatically manage instance creation and integrate with the load balancer to distribute traffic across multiple availability zones.

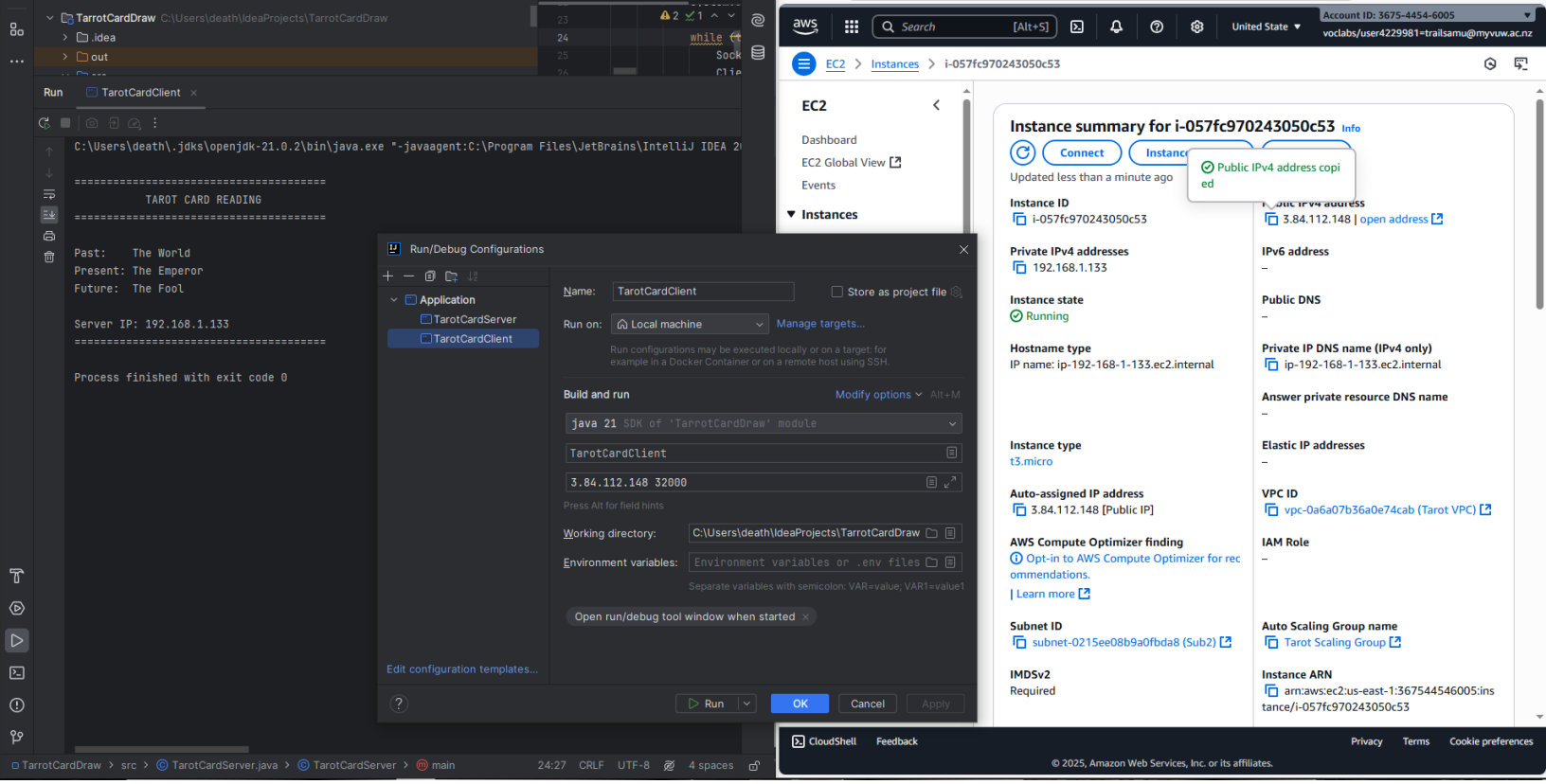


Screenshot 20,21,22

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.

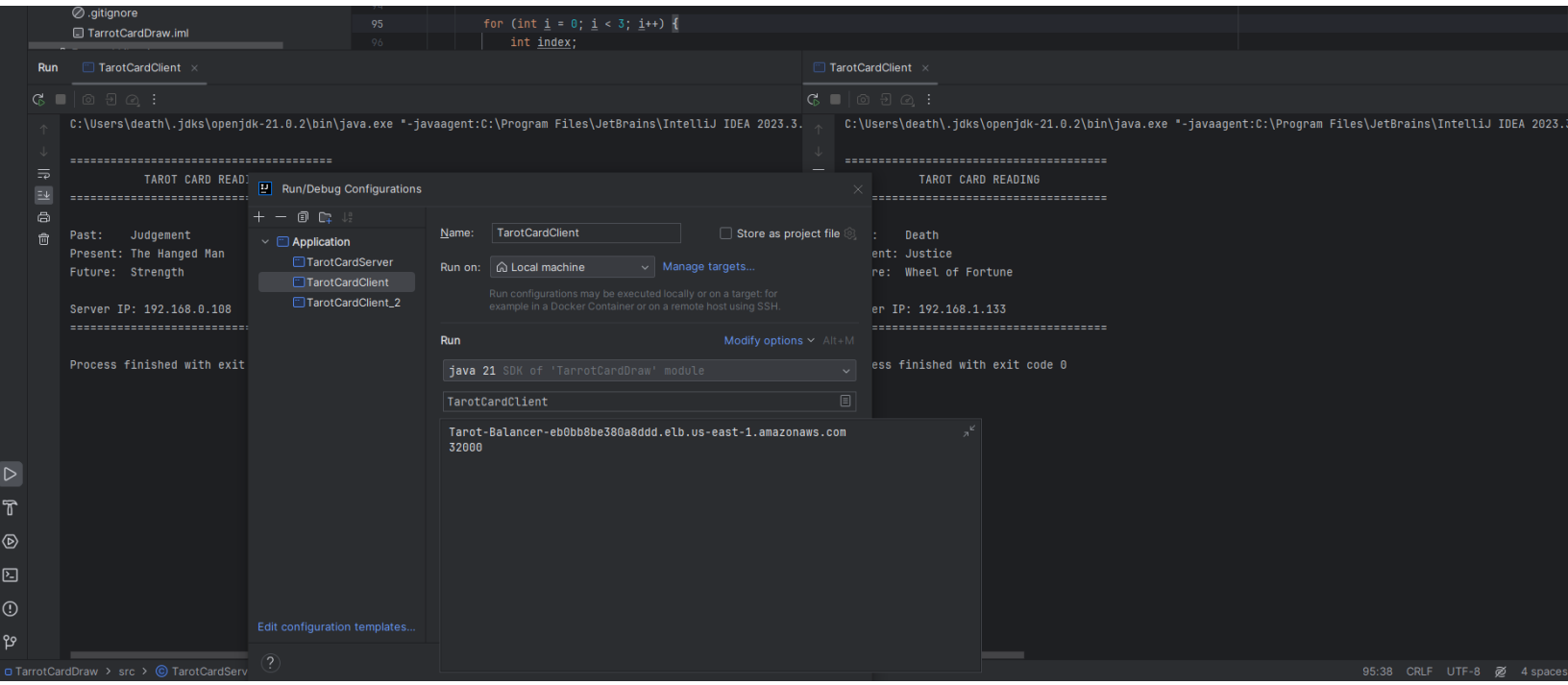

Search [Alt+S]
United States (N. Virginia)
Account ID: 3675-4454-6005
voclabs/user4229981=trailsamu@myvuw.ac.nz
EC2 > Instances
EC2
Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
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
Instances (6) Info
Connect
Instance state
Actions
Launch instances
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
Tarrot Card Se...
i-0929c67f7d1bb0a0e
Running
t2.micro
2/2 checks passed
View alarms
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
View alarms
us-east-1c
ec2-34-227-52-184.co...
34.227.52.184
i-0dd53392b27ffa18a
Running
t3.micro
3/3 checks passed
View alarms
us-east-1a
44.199.210.217
i-057fc970243050c53
Running
t3.micro
3/3 checks passed
View alarms
us-east-1b
3.84.112.148
i-08fc1b7eeea4d6f52
Terminated
t3.micro
View alarms
us-east-1a
i-0afedb2e77814dbea
Terminated
t3.micro
View alarms
us-east-1b
Select an instance
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Project
TarrotCardDraw C:\Users\death\IdeaProjects\TarrotCardDraw
.idea
out
TarotCardServer.java
Run
TarotCardClient
C:\Users\death\.jdks\openjdk-21.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
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
Application
TarotCardServer
TarotCardClient
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.
Build and run
Modify options Alt+M
java 21 SDK of 'TarrotCardDraw' module
TarotCardClient
44.199.210.217 32000
Press Alt for field hints
Working directory: C:\Users\death\IdeaProjects\TarrotCardDraw
Environment variables: Environment variables or .env files
Separate variables with semicolon: VAR=value; VAR1=value1
Open run/debug tool window when started
Edit configuration templates...
Run
OK
Cancel
Apply
TarrotCardDraw > src > TarotCardServer.java > TarotCardServer > main
24:27 CRLF UTF-8 4 spaces
EC2 > Instances > i-0dd53392b27ffa18a
Instance summary for i-0dd53392b27ffa18a Info
Connect
Instance state
Actions
Updated less than a minute ago
Instance ID
i-0dd53392b27ffa18a
Private IPv4 addresses
192.168.0.108
Instance state
Running
Hostname type
IP name: ip-192-168-0-108.ec2.internal
Instance type
t3.micro
Auto-assigned IP address
44.199.210.217 [Public IP]
AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations.
Learn more
Subnet ID
subnet-048e6bc29c891cfc6 (Sub1)
IMDSv2
Required
Public IPv4 address
44.199.210.217 | open address
IPv6 address
Public DNS
Private IP DNS name (IPv4 only)
ip-192-168-0-108.ec2.internal
Answer private resource DNS name
Elastic IP addresses
VPC ID
vpc-0a6a07b36a0e74cab (Tarot VPC)
IAM Role
Auto Scaling Group name
Tarot Scaling Group
Instance ARN
arn:aws:ec2:us-east-1:367544546005:instance/i-0dd53392b27ffa18a
Key Pairs
Network Interfaces
CloudShell
Feedback
Privacy
Terms
Cookie preferences
© 2025, Amazon Web Services, Inc. or its affiliates.



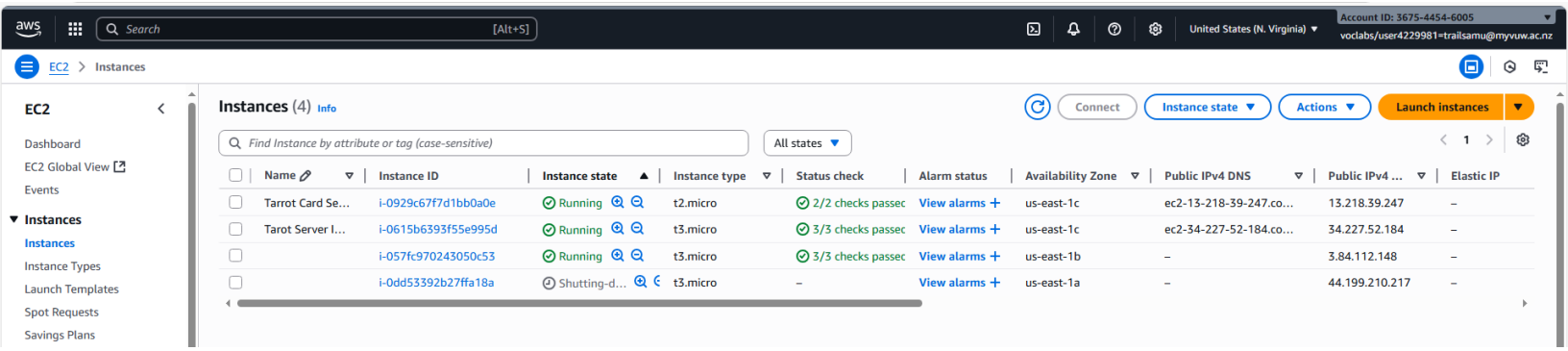
Screenshot 23

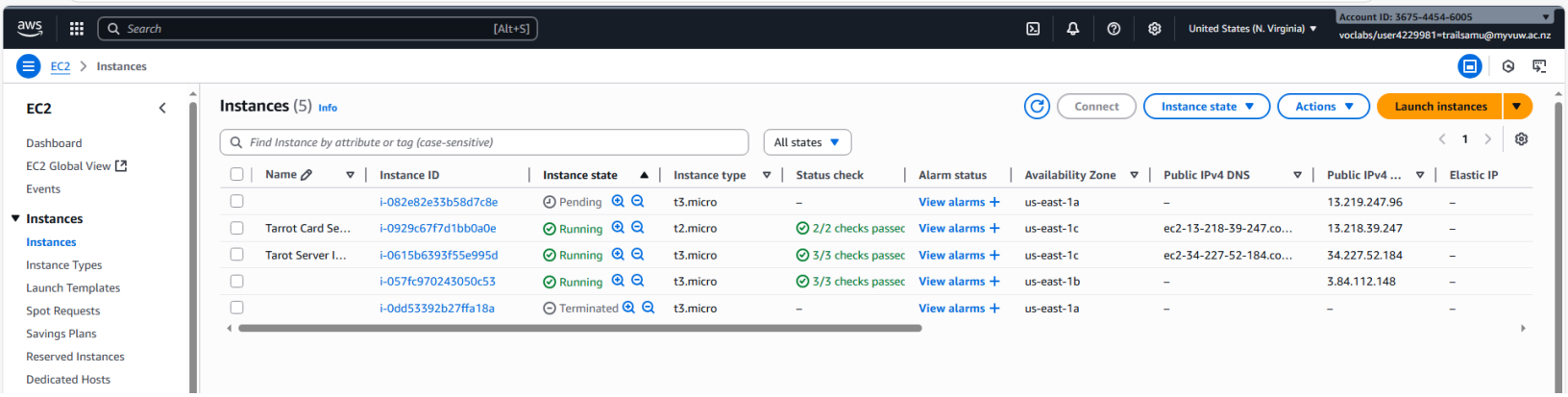
This screenshot demonstrates the successful testing of the load balancer functionality, showing the Tarot card client connecting to the load balancer's DNS name on port 32000, which transparently routes traffic to backend instances running in different availability zones (as evidenced by the different 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.

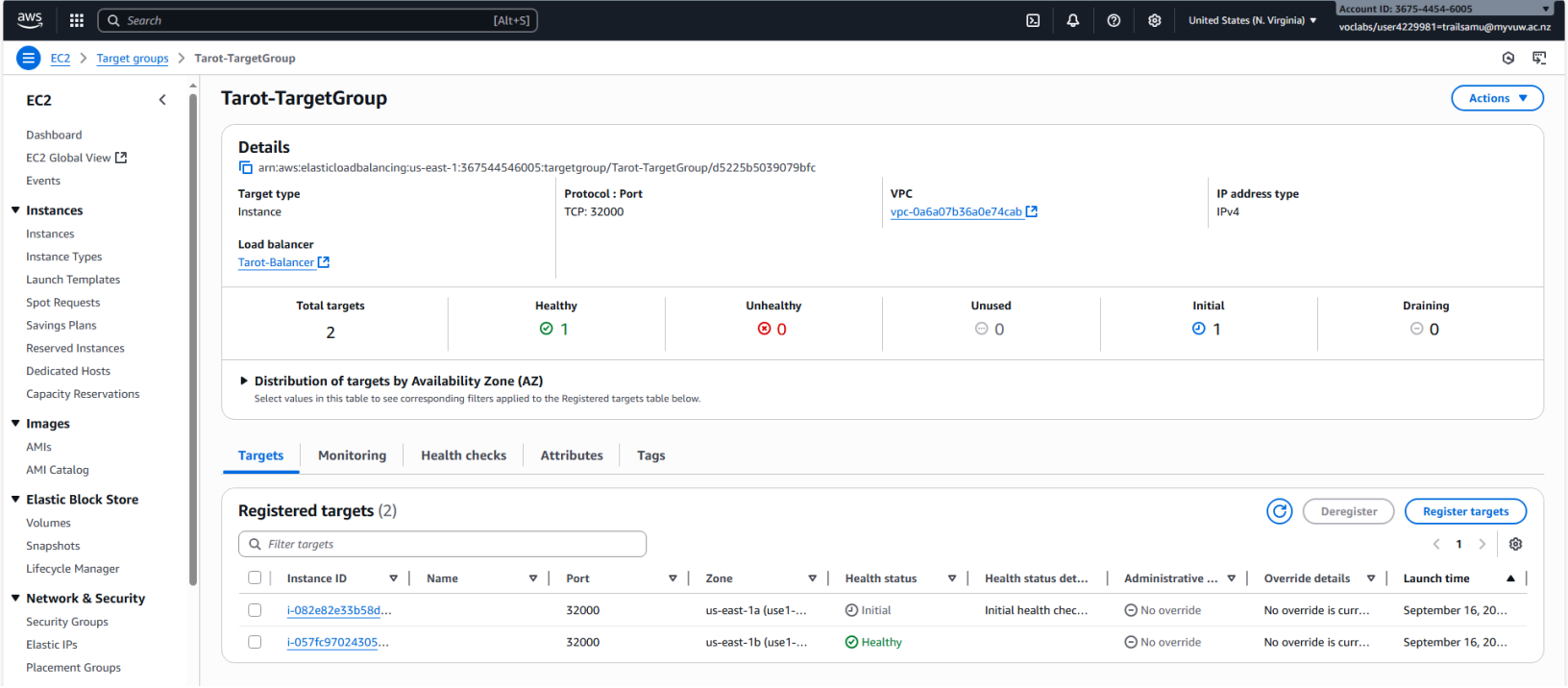


Notes:

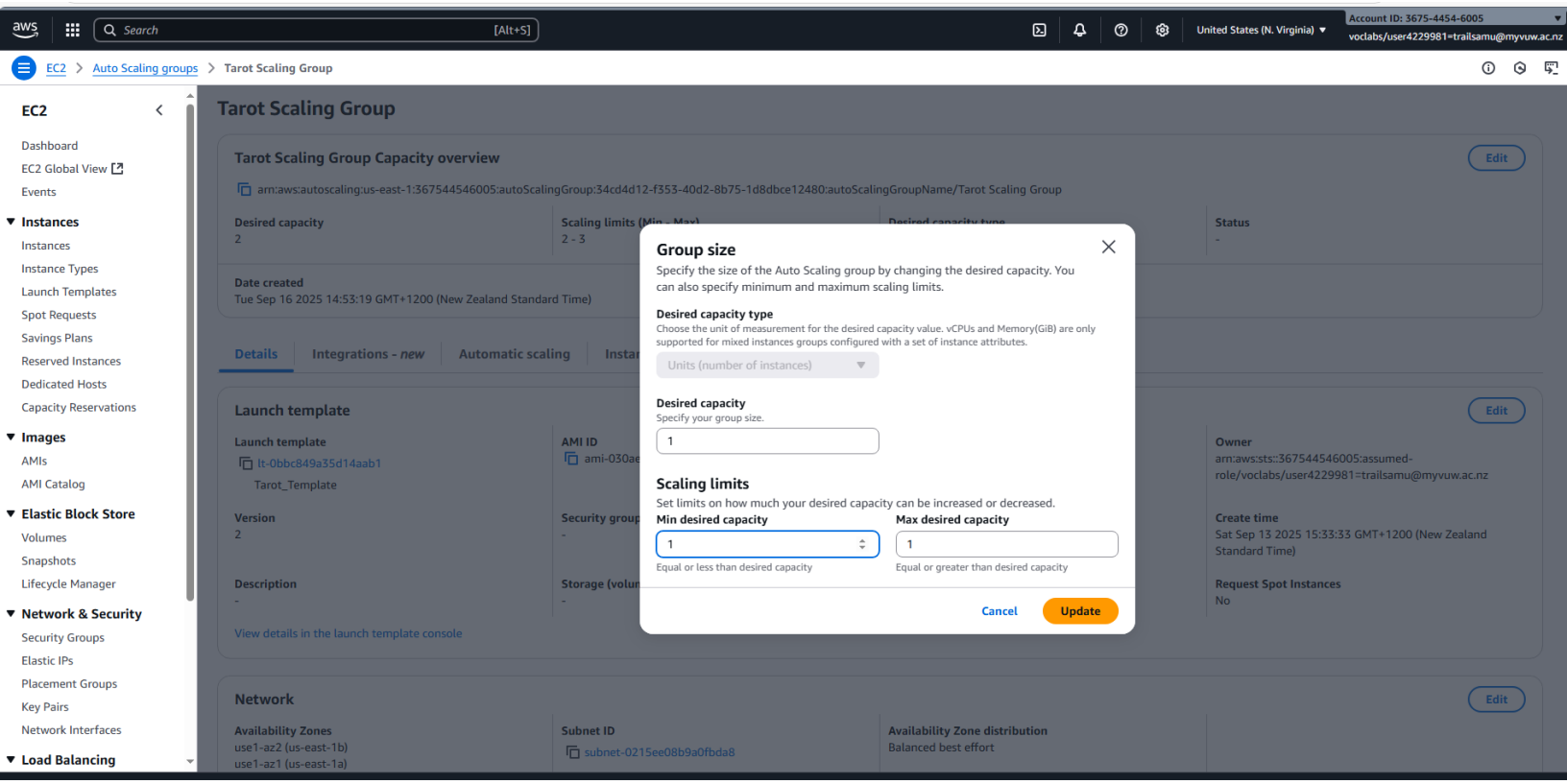
When I terminated one instance, the auto scaling group immediately detected the termination and automatically launched a new replacement instance to maintain the desired capacity. The load balancer continued routing traffic to the remaining healthy instance, ensuring no service interruption occurred.







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.



EC2

Dashboard
EC2 Global View
Events

Instances
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

Tarot-TargetGroup

Actions

Details

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

Target type	Protocol : Port	VPC	IP address type
Instance	TCP: 32000	vpc-0a6a07b36a0e74cab	IPv4

Load balancer
Tarot-Balancer

Total targets	Healthy	Unhealthy	Unused	Initial	Draining
2	1	0	0	0	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)

Deregister | Register targets

Filter targets

1

Instance ID	Name	Port	Zone	Health status	Health status det...	Administrative ...	Override details	Launch time
i-082e82e33b58d...		32000	us-east-1a (use1-...	Healthy		No override	No override is curr...	September 16, 20...
i-057fc97024305...		32000	us-east-1b (use1-...	Draining	Target deregistrat...	No override	No override is curr...	September 16, 20...