National College of Ireland

Configuration Manual

MSc Research Project
MSc Cloud Computing

Ritesh Panaganti
Student ID: X23344563

School of Computing
National College of Ireland

Supervisor: Jorge Mario Cortes Mendoza

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……Ritesh Panaganti……

Student ID: ……X23344563……

Programme:…… MSc Cloud Computing…… **Year:** ……2025……..

Module: …… Research Project……

Supervisor: ……Jorge Mario Cortes Mendoza……

Submission Due Date: ……11/08/2025……

Project Title: ……Configuration Manual……

Word Count: ……1198…… **Page Count**……6……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……Ritesh Panaganti……

Date: ……11/08/2025……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Ritesh Panaganti
X23344563

1. ECS Cluster Setup

This research project runs four microservices on Amazon Elastic Container Service (Amazon ECS). To create an ECS cluster:

1. Login to Amazon Web Services.
2. Navigate to Amazon ECS and click on Create Cluster.
3. In Cluster configurations enter the desired cluster name and in infrastructure select AWS Fargate (serverless). Then click on Create at the bottom right of the window.

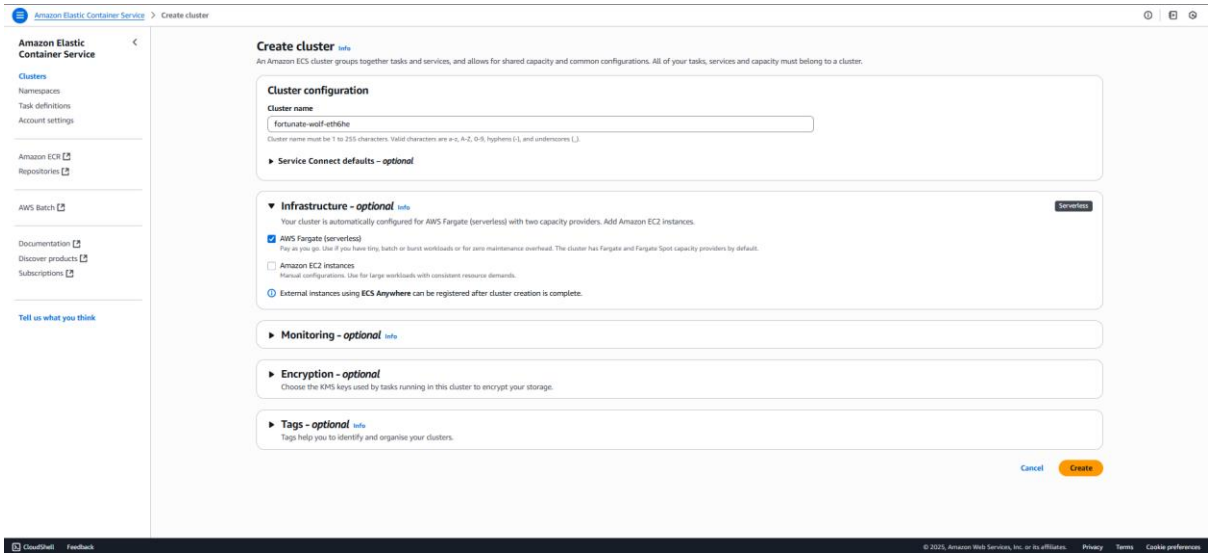

Fig 1: Screenshot showing creation of ECS Cluster

2. Amazon Elastic Container Registry (Amazon ECR)

Amazon ECR is a repository used to store the docker images. To create a repository, navigate to *Elastic Container Registry* → *Create a private repository (Create)*.

Give the *repository name* and then click on *Create*.
Repository will be created and a URI will be assigned to the repository which is used to push the docker images and also to deploy to Amazon ECS.

Example URI: 705331611976.dkr.ecr.us-east-1.amazonaws.com/<repository name>

Create four different repositories for four microservices.

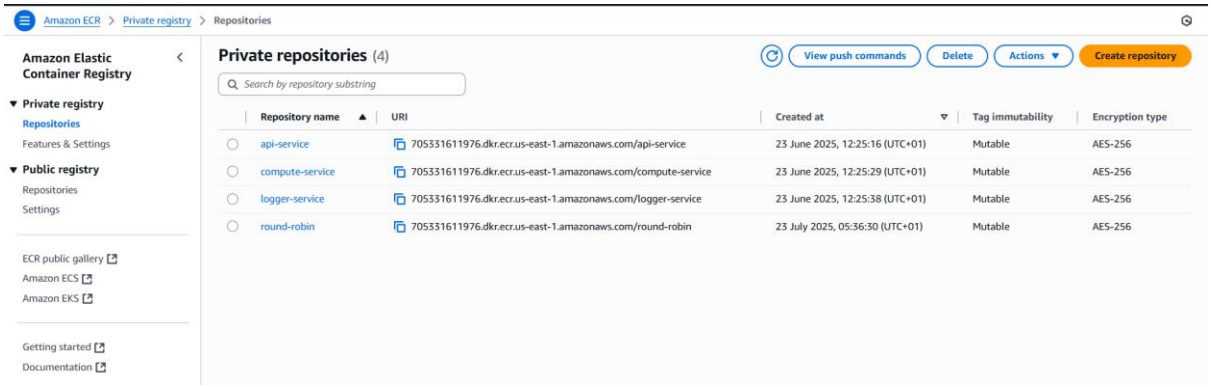

Fig 2: Screenshot of Amazon Elastic Container Registry

3. Task Definitions

Need to create four task definitions because we have four microservices, each microservice needs a task definition. To create task definitions:

1. Navigate to *Amazon ECS* → *Task definition* → *Create new task definition*
2. In the task definition configurations fill all the details:
   Task definition family: < task definition name >
   Launch type: AWS Fargate
   Operation system/Architecture: Linux/X86_64
   CPU: .25 vCPU
   Memory: .5 GB
   Task Role: LabRole
   Task execution role: LabRole
   Container: Name: <Container Name>
   Image URI: < Respective ECR repository URI>
   Container port: 5000 (for api services)
   5001 (For Compute service)
   5002 (For Logger service)
   Protocol: TCP
   Port name: <name>
   App protocol: HTTP
   Leave other configurations as default and click on *Create* in the bottom right.
3. Similarly create four task definitions ensuring Image URI and Container port are matching with respect to the microservice.

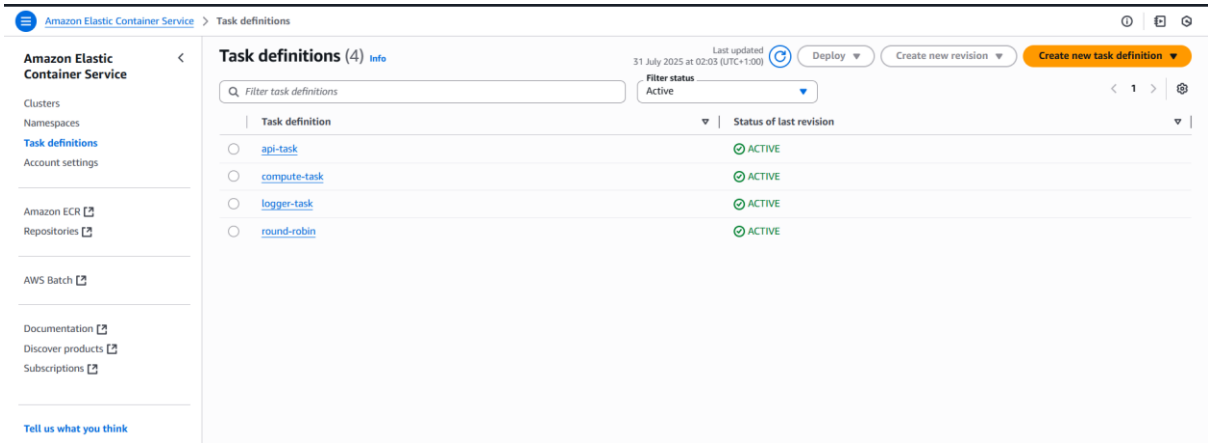

Fig 3: Screenshot showing Task Definitions

4. Target Groups

Need to Create four target groups for all the four microservices. To create target groups navigate to *EC2 → Target groups → Create target group*.
Target group details:

Choose a target type: IP addresses
Target group name: <Name>
IP address type: IPv4
VPC: default
Protocol version: HTTP1
Health check protocol: HTTP
Health check path: / (For both api services)
/health (for compute and logger services)

Then click on Create target group.
Similarly create all four target groups ensuring correct health check path for the correct microservice.

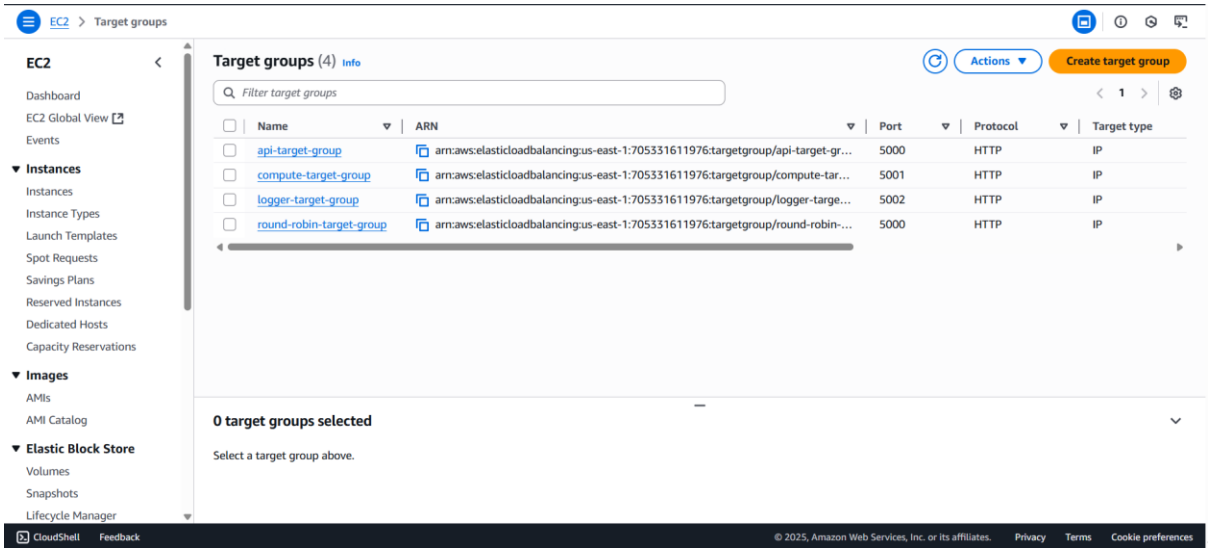

Fig 4: Screenshot showing Target groups

5. Application Load Balancer (ALB)

Once all the four target groups are created correctly then need to create an Application Load Balancer (ALB). To create ALB, navigate to *EC2 → Load balancers → Create load balancer*.
Select Application load balancer.
Load balancer name: <Name>
Scheme: Internet-facing
Load balancer IP address type: IPv4
VPC: default
Availability Zones and subnets: Select all
Protocol: HTTP
Port: 80
Default action (forward to): Select api-target-group from drop down (Select from target groups created in previous step).
Verify details in the review section and click on *Create load balancer.*
This will create the Application load balancer and take note of the DNS name.

Example DNS name: alb-ecs-788061870.us-east-1.elb.amazonaws.com

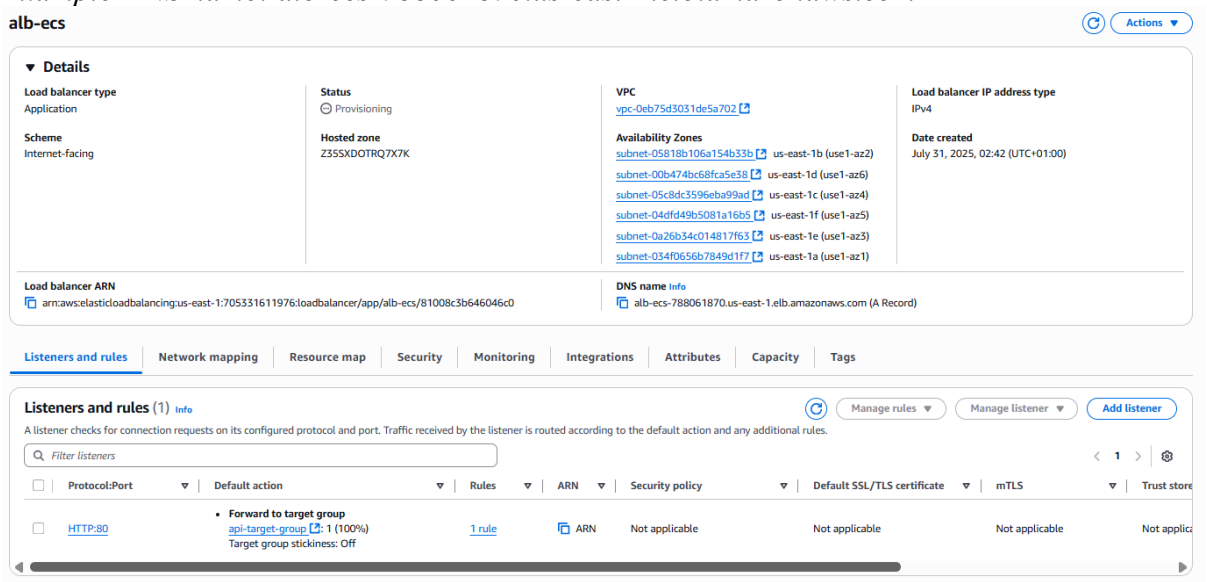

Fig 5: Screenshot of Application Load Balancer

6. Listener Rules

Once the ALB is created successfully, we need to add custom rules to route traffic to all the four Microservices. To add rules, navigate to the *ALB → Listeners and rules → rules → Add rule*

Add condition: path

Path: /round (for api-round-robin service)

/compute (For compute service)

/log (for Logger service)

Routing action: Forward to target groups

Target group: Select correct target group from the drop down with respect to path.

Click Next

Priority: 1 for api round robin service.

2 for compute service.

3 for logger service.

Click next, review the details and *Create*.

Repeat same steps to add 3 rules.

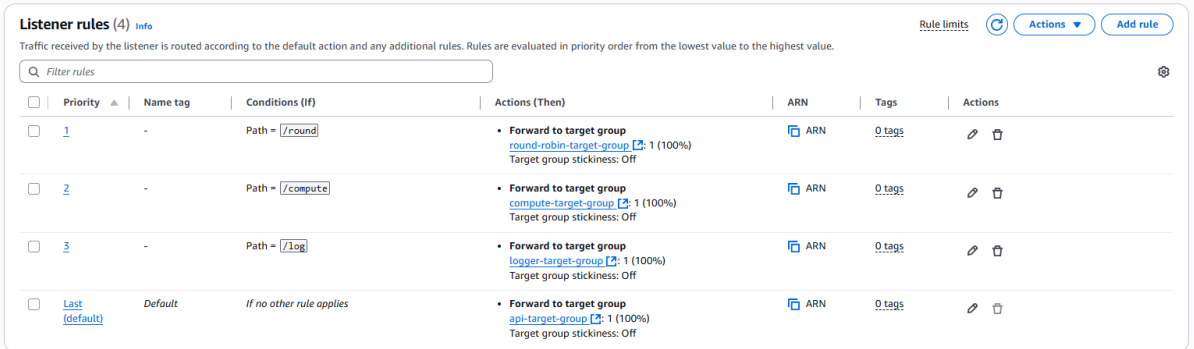

Fig 6: screenshot showing listener rules

7. Creating ECS Services

Before creating ECS services ensure ECS cluster, Task definitions, Application load balancer, Target groups and Listener rules are created and configured correctly. Without these we cannot create ECS service as they are important in configuration part. To create ECS Service navigate to *Elastic container service → api-cluster → Services → Create*
Task definition family: Select task definition from the drop down.
Service Name: <Name>
Load balancing:

Load balancer type: Application load balancer → use existing load balancer and select the load balancer which was created in the previous step.

Listener: Use existing listener and select the listener.

Target group: Use existing target group → Select target group from the from the drop down (Ensure selecting correct target group associated with the microservice).

Service auto scaling:

Use service auto scaling
Minimum number of tasks: 1
Maximum number of tasks: 2
Scaling policy type: Target tracking
Policy name: <name>
ECS service metric: ECSServiceAverageCPUUtilization
Target value: 60 or 70 or 80 accordingly

Review all the configuration and click *Create*.

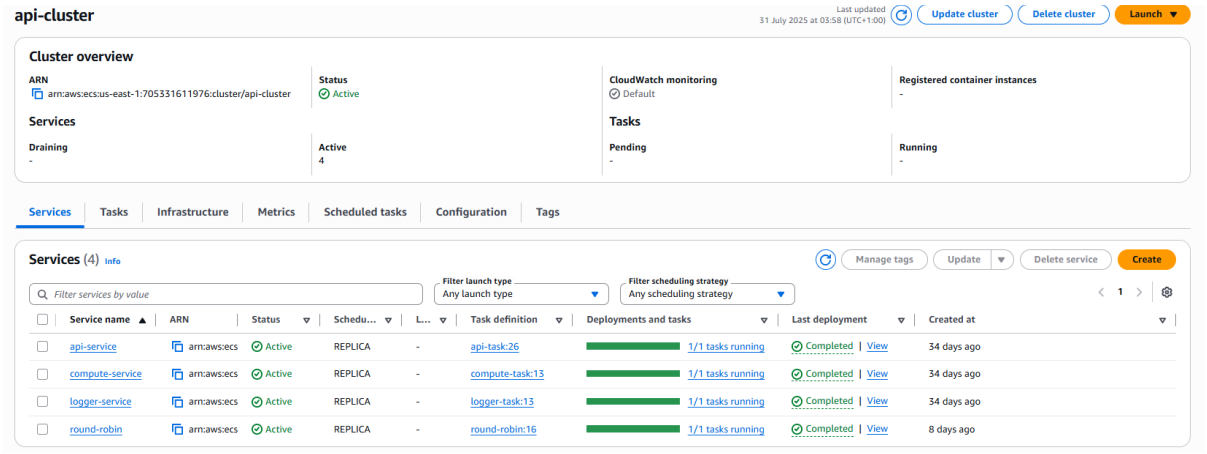

Fig 7: Screenshot showing ECS cluster with services and running tasks

8. Testing Services

Ensure all the services are up and running.
Ensure all the target groups are showing healthy status.
Copy the DNS name from the application load balancer.

Example DNS name: alb-ecs-788061870.us-east-1.elb.amazonaws.com

Try accessing the url in the browser with ALB DNS that is
http://<alb dns>/ → for API service
http://<alb dns>/compute → for compute service

http://<alb dns>/log → for logger service
http://<alb dns>/round → for round robin Load balancing
http://<alb dns>/random → for random load balancing

Example:
http:// *alb-ecs-788061870.us-east-1.elb.amazonaws.com/*
http:// *alb-ecs-788061870.us-east-1.elb.amazonaws.com/compute*
http:// *alb-ecs-788061870.us-east-1.elb.amazonaws.com/log*
http:// *alb-ecs-788061870.us-east-1.elb.amazonaws.com/round*
http:// *alb-ecs-788061870.us-east-1.elb.amazonaws.com/random*

9. References

Ahmad, H., Treude, C., Wagner, M. and Szabo, C., 2025. Towards resource-efficient reactive and proactive auto-scaling for microservice architectures. *Journal of Systems and Software* 213: 112772.

Lu, C., Zhou, J. and Zou, Q., 2025. An optimized approach for container deployment driven by a two-stage load balancing mechanism *PLOS ONE* 20(1): e0317039.

Biegel, G., 2024. Leveraging cloud compute and open source software to generate 3D models from drone photography. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLVIII-4-2024: 73–80.

Miao, J., Rajasekhar, D., Mishra, S., Nayak, S.K. and Yadav, R., 2024. A microservice-based smart agriculture system to detect animal intrusion at the edge. *Future Internet* 16(5): 196.

Saboor, A., Mahmood, A.K., Hassan, M.F., Shah, S.N.M., Hassan, F. and Siddiqui, M.A., 2021, July. Design pattern based distribution of microservices in cloud computing environment. In *2021 International Conference on Computer & Information Sciences (ICCOINS)* 1-6.

Hu, K., Wen, L., Xu, M. and Ye, K., 2024. MSARS: A Meta-Learning and Reinforcement Learning Framework for SLO Resource Allocation and Adaptive Scaling for Microservices, *Journal of Systems Architecture* 150: 103258.

Kambala, G., 2025 Integration Of Microservices And Cloud Computing: A Paradigm Shift In Enterprise Application Design.

Li, W., Li, X., Chen, L. and Wang, M., 2025. Microservice Workflow Scheduling with a Resource Configuration Model Under Deadline and Reliability Constraints. *Sensors* 25(4): 1042.

Arya, S., Chauhan, D., Anand, S. and Sharma, O., 2024, August. Beyond Monoliths: An In-Depth Analysis of Microservices Adoption in the Era of Kubernetes. In *2024 1st International Conference on Advanced Computing and Emerging Technologies (ACET)* 979–986.

Matos, G.H.M., Carvalho, M. and Macedo, D.F., 2024, November. Container-Based Microservice Scheduling Using Reinforcement Learning in Distributed Cloud Computing. In *2024 IEEE Latin-American Conference on Communications (LATINCOM)* 1-6.