## Task:

For this task, you're going to be part of the deployment team of a tech company. Our client is a start-up that wants to deploy a simple web application and ensure its availability to accommodate variable traffic volumes. You're asked to:

1. Use Docker to run a simple web application on AWS utilizing:

- EC2 Auto-scaling group(or)

- ECS Fargate(or)

2. Direct incoming traffic to the application through a load balancer.

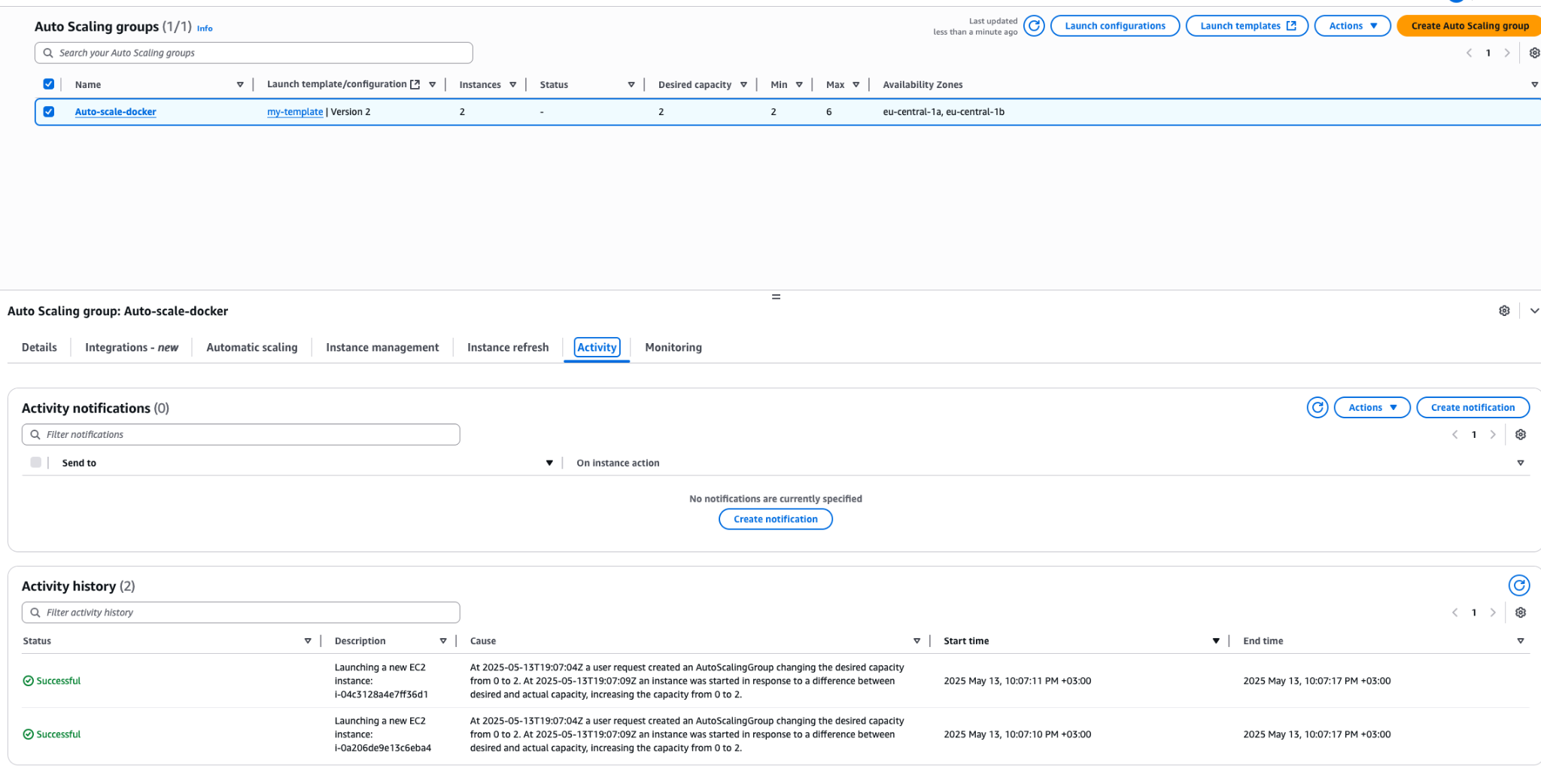
3. Use a load testing tool (eg [Locust](https://locust.io/)) to simulate high traffic load on the endpoint, triggering resource scaling and observing the process. Add a simulation screenshot to the report.

4. In conclusion, prepare a report detailing the pros and cons of each deployment method in terms of ease of setup, cost, and scalability. Attach the created report to Trello task.

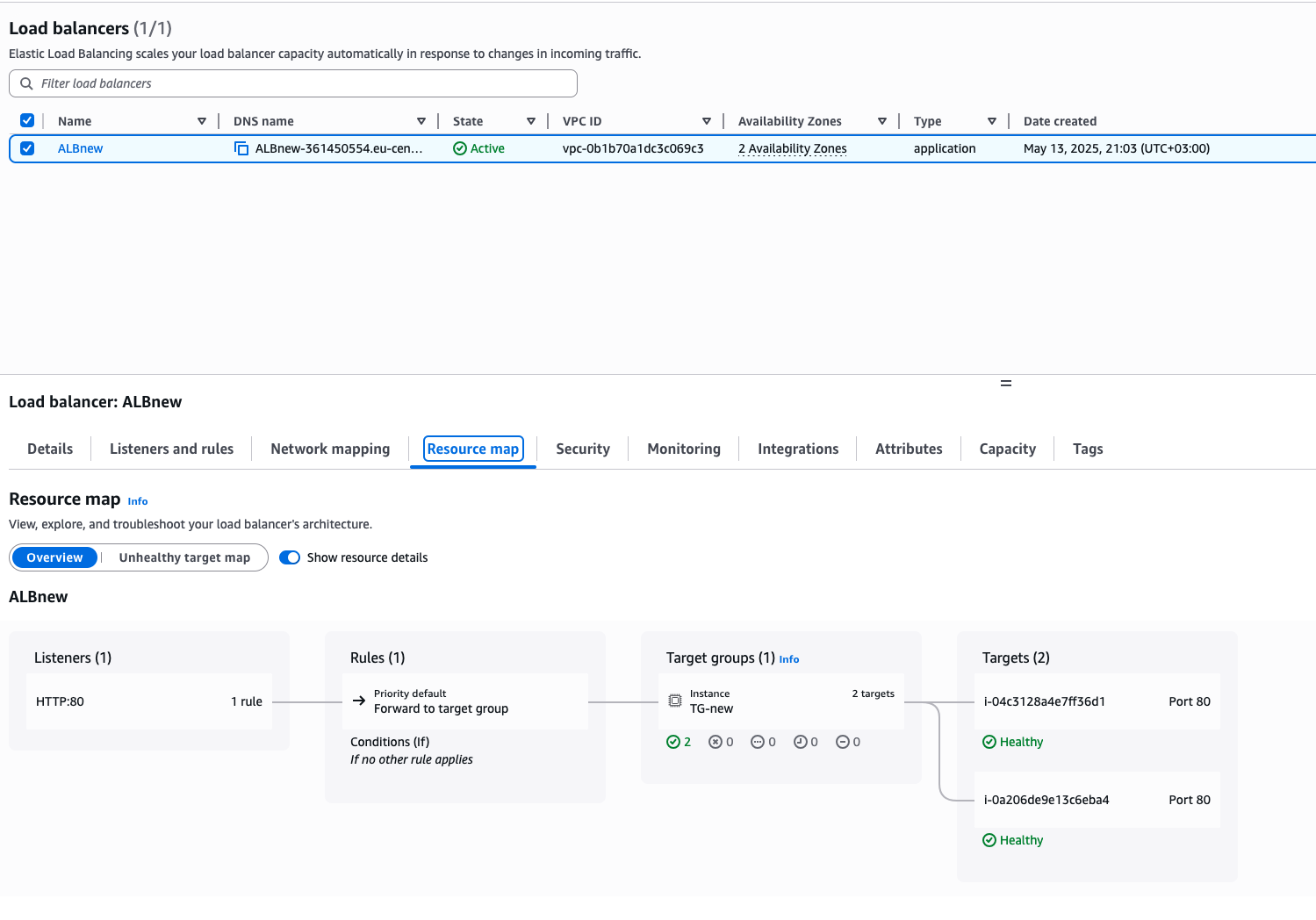
**Note:** You can use this simple application for deployment - <https://github.com/benc-uk/nodejs-demoapp> (Docker image is ready).

## Method EC2 Auto-scaling group Testing:

### Before high traffic:

[](https://github.com/benc-uk/nodejs-demoapp)

## 



### Add traffic using Locust:

### 

### After high traffic:

### 

## Conclusion

| Solution | EC2 + Auto-scaling | Fargate |
| --- | --- | --- |
| Easy of setup | Hard | Simpler |
| Customization | Full control of OS and Docker | Limited |
| Infrastructure management | You manage EC2 instances, AMIs, patching, etc. | No server management — AWS handles compute provisioning |
| Startup time | Slower | Faster |
| Pricing model | Pay per EC2 instance uptime | Pay per vCPU and memory per task second |
| Scalability | Auto Scaling group required | Task-level auto scaling |
| Security and isolation | More control over host OS, networking, and firewalls | Better isolation (tasks run on separate VMs internally) |
| Use case | Long-running containers, custom OS needs, heavy workloads | Short tasks, microservices, event-driven apps, simplicity focus |