

HW4 Report

Rong Huang

CS6650 25Fall

This week I finally got a full feel for the ECR/ECS workflow. It's a lot of clicking the first time, but once the pieces connect (push image to ECR, define a task, run it on Fargate), it starts to make sense. I can see why people automate this with tools like Terraform—we'll probably do that next.

Part II: Infrastructure set up!

Result

Cluster hello-cluster has been created successfully.

View cluster

Task definition successfully created
hello-task-1 has been successfully created. You can use this task definition to deploy a service or run a task.

View task definition

Tasks launched
arn:aws:ecs:us-west-2:381492270964:task/hello-cluster/523ed9c246f248afa8ed41ec31704343

hello-cluster

Last updated
September 29, 2025, 03:40 (UTC-4:00)

Update clusterDelete clusterLaunch

Cluster overview

ARN
arn:aws:ecs:us-west-2:381492270964:cluster/hello-cluster

Status
Active

CloudWatch monitoring
Default

Registered container instances
-

Services

Draining
-

Active
-

Pending
-

Running
1

ServicesTasksInfrastructureMetricsScheduled tasksConfigurationTags

Tasks (1)

Last updated
September 29, 2025, 03:40 (UTC-4:00)

Manage tagsStopRun new task

Filter tasks by property or value

Filter desired status
Any desired status

Filter launch type
Any launch type

Task

Last status

Desired st...

Task de...

Health sta...

Created at

Started by

Started at

Group

Container instan...

Launch type

Platf

523ed9c246f248afa8ed4...

Provisioning

Running

hello-task:1

Unknown

4 minutes ago

-

-

family:hello-task

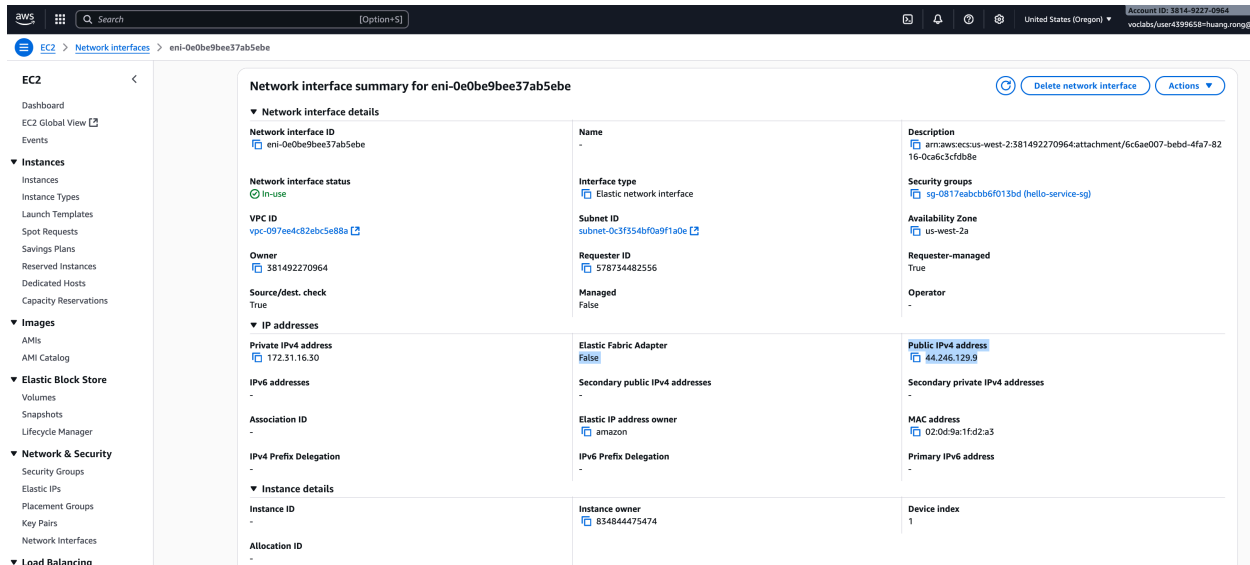
-

FARGATE

1.4.0

HW4 Report

1



```
(base) ronghuang@Reginas-macbook hello-service % curl http://44.246.129.9:8080/albums
```

```
[
  {
    "id": "1",
    "title": "Blue Train",
    "artist": "John Coltrane",
    "price": 56.99
  },
  {
    "id": "2",
    "title": "Jeru",
    "artist": "Gerry Mulligan",
    "price": 17.99
  },
  {
    "id": "3",
    "title": "Sarah Vaughan and Clifford Brown",
    "artist": "Sarah Vaughan",
    "price": 39.99
  }
]
```

```
(base) ronghuang@Reginas-macbook hello-service % curl http://44.246.129.9:8080/albums/1
```

```
{
  "id": "1",
  "title": "Blue Train",
  "artist": "John Coltrane",
  "price": 56.99
}
```

```
(base) ronghuang@Reginas-macbook hello-service % █
```

Things I explored and what I learned

1. Difference between EC2 and ECS

- EC2: I manage the virtual machines myself.

I have to think about installing stuff, scaling up, patching, etc.

It's flexible but more work.

- ECS (with Fargate): I just tell AWS what container to run and how much CPU/memory it needs. No servers to manage. This was easier for my small tasks.
2. What is a VPC and a subnet? How did I access the default VPC?
 - VPC is like my private network in AWS. A subnet is a slice of that network in a specific availability zone.
 - I used the default VPC and a public subnet. I enabled a public IP on my task and opened port 8080 in the security group, so I could curl the service from my laptop.
 3. What is TCP and how is it different from UDP?
 - TCP is a reliable connection (it makes sure all data arrives and in order). It's good for HTTP and my JSON requests.
 - UDP is faster but doesn't guarantee delivery or order. It's more for streaming or custom cases where I handle reliability myself.
 4. How do I control resources for a task?
 - In the task definition, I set CPU and memory (for example, 0.25 vCPU and 0.5 GB). Fargate enforces that. If I need more power, I can bump these numbers or run more tasks at once.

Part III: Map Reduce

Result

Amazon Elastic Container Service > Clusters > mapreduce-cluster > Tasks

mapreduce-cluster

Cluster overview

ARN: arn:aws:ecs:us-west-2:381492270964:cluster/mapreduce-cluster

Status: Active

CloudWatch monitoring: Default

Registered container instances: 5

Services: Draining

Tasks: Pending

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

Tasks (6)

Filter tasks by property or value

Filter desired status: Any desired status

Filter launch type: Any launch type

| Task | Last status | Desired status | Task definition | Health status | Created at | Started by | Started at | Group | Container info |
|----------------------------|-------------|----------------|-----------------|---------------|----------------|------------|----------------|----------------------|----------------|
| 3e48ffa5b8f34ebdafc8dcf... | Running | Running | mapper-task:3 | Unknown | 22 minutes ago | - | 22 minutes ago | family:mapper-task | - |
| 6e7716b879c84c95b2152... | Running | Running | reducer-task:2 | Unknown | 20 minutes ago | - | 20 minutes ago | family:reducer-task | - |
| 9221fd7e8c044338b90e7... | Running | Running | mapper-task:3 | Unknown | 22 minutes ago | - | 22 minutes ago | family:mapper-task | - |
| de417c8a88cc4253a8352... | Running | Running | splitter-task:4 | Unknown | 1 minute ago | - | 1 minute ago | family:splitter-task | - |
| e31d560968d64737a36e... | Running | Running | mapper-task:3 | Unknown | 22 minutes ago | - | 22 minutes ago | family:mapper-task | - |

```
(base) ronghuang@Reginas-macbook reducer % aws s3 ls s3://mapreduce-regina-1759134570
/ --region us-west-2 | cat
aws s3 ls s3://mapreduce-regina-1759134570/input/ --region us-west-2 | cat
2025-09-29 04:32:11      123 test.txt
(base) ronghuang@Reginas-macbook reducer % aws ec2 describe-network-interfaces --network-interface-ids eni-064995fef87407b71 --region us-west-2 --query 'NetworkInterfaces[0].Association.PublicIp' --output text | cat
aws ec2 describe-network-interfaces --network-interface-ids eni-064995fef87407b71 --region us-west-2 --query 'NetworkInterfaces[0].Association.PublicIp' --output text | cat
35.90.79.175
<TTER $MAPPER1 $MAPPER2 $MAPPER3 $REDUCER $BUCKET_URL
http://35.88.206.27:8080 http://44.244.94.248:8080 http://35.86.204.227:8080 http://35.86.204.227:8080 http://35.90.79.175:8080 s3://mapreduce-regina-1759134570/input/test.txt
(base) ronghuang@Reginas-macbook reducer % curl -s -X POST "$SPLITTER/split-s3" -H "Content-Type: application/json" -d '{"s3_url":"$BUCKET_URL","curl -s -X POST "$SPLITTER/split-s3" -H "Content-Type: application/json" -d '{"s3_url":"$BUCKET_URL","chunks":3}' | tee /tmp/split_resp.json | cat
{"chunk_urls":["s3://mapreduce-regina-1759134570/chunks/1759151384/chunk_0.txt","s3://mapreduce-regina-1759134570/chunks/1759151384/chunk_1.txt","s3://mapreduce-regina-1759134570/chunks/1759151384/chunk_2.txt"],"total_chunks":3}
(base) ronghuang@Reginas-macbook reducer %
```

```
1  {
2    "final_count": {
3      "awesome": 2,
4      "hello": 4,
5      "is": 1,
6      "mapreduce": 3,
7      "test": 4,
8      "world": 5
9    },
10   "total_words": 19.
11 }
12
13 TAB to Dockerfile →
14
15 ... SPLITTER=http://35.88.206.27:8080; MAPPER1=http://44.244.94.248:8080; MAPPER2=http://35.86.204.227:8080; MAPPER3=http://35.8
16
17 OK
18 http://35.86.204.227:8080/health
19 OK
20 http://35.86.204.227:8080/health
21 OK
22 http://35.90.79.175:8080/health
23 OK
24 --- SPLIT ---
25 {"chunk_urls":["s3://mapreduce-regina-1759134570/chunks/1759151520/chunk_0.txt","s3://mapreduce-regina-175913
26 4570/chunks/1759151520/chunk_1.txt","s3://mapreduce-regina-1759134570/chunks/1759151520/chunk_2.txt"],"total_
27 chunks":3}
28 Chunks:
29 s3://mapreduce-regina-1759134570/chunks/1759151520/chunk_0.txt
30 s3://mapreduce-regina-1759134570/chunks/1759151520/chunk_1.txt
31 s3://mapreduce-regina-1759134570/chunks/1759151520/chunk_2.txt
32 --- MAP ---
33 {"result_url":"s3://mapreduce-regina-1759134570/map-results/1759151521/chunk_0_result.json"}
34 {"result_url":"s3://mapreduce-regina-1759134570/map-results/1759151521/chunk_1_result.json"}
35 {"result_url":"s3://mapreduce-regina-1759134570/map-results/1759151521/chunk_2_result.json"}
36 Results:
37 s3://mapreduce-regina-1759134570/map-results/1759151521/chunk_0_result.json
38 s3://mapreduce-regina-1759134570/map-results/1759151521/chunk_1_result.json
39 s3://mapreduce-regina-1759134570/map-results/1759151521/chunk_2_result.json
40 --- REDUCE ---
41 {"final_result_url":"s3://mapreduce-regina-1759134570/final-results/1759151522/final_word_count.json"}
42 Final: s3://mapreduce-regina-1759134570/final-results/1759151522/final_word_count.json
43 download: s3://mapreduce-regina-1759134570/final-results/1759151522/final_word_count.json to ./final_word_cou
44 nt.json
45 {"final_count":{"awesome":2,"hello":4,"is":1,"mapreduce":3,"test":4,"world":5},"total_words":19,"unique_words
46 ":6,"top_10_words":[{"word":"world","count":5}, {"word":"hello","count":4}, {"word":"test","count":4}, {"word":
47 "mapreduce","count":3}, {"word":"awesome","count":2}, {"word":"is","count":1}]}
48
49 (base) ronghuang@Reginas-macbook cs6650 %
```

Mini MapReduce (word count) — what I built

- One Splitter task: reads a text file from S3, splits it into 3 chunks, and writes the chunks back to S3.
- Three Mapper tasks: each reads one chunk, counts words, and writes a small JSON to S3.
- One Reducer task: combines the three mapper JSON files, sums the counts, and writes the final result to S3.

I used simple HTTP endpoints on port 8080 for each. All tasks ran on ECS Fargate. IAM role had S3 read/write and ECR pull.

My results (end-to-end time)

- Data: Shakespeare's Hamlet plain text, about 159 KB, stored in S3. Source: shakespeare-hamlet.txt.
- Because the file is small, total time is dominated by startup and network/S3 overhead. The actual counting is essentially instant.
- With such a small input, using 1, 2, or 3 mappers showed no meaningful speedup; overhead hides the benefit.

Questions

- The first run was slower due to image pull ("cold start"). Warm runs were quicker but still dominated by overhead at this input size.
- To see clear parallel gains, I would use a larger input (e.g., 50–100 MB) made by concatenating the same text multiple times before uploading to S3.
- S3 I/O and network can become the bottleneck when inputs are small.

1. What if a mapper fails? How would I recover?

- Retry the failed chunk. Outputs use clear S3 keys, so re-running is safe (overwrite with the same name). The reducer can wait/retry until all expected outputs exist.

2. How to scale to 10 or 100 mappers?

- Split into more chunks (e.g., 10 or 100).
- Launch more mappers in parallel on ECS.
- Keep S3 keys organized (job ID + chunk number) for easy listing.
- For many chunks, consider a tree-style reduce to avoid a single reducer bottleneck.

3. What was the challenging part of coordinating tasks manually?

- Setting public IPs and opening the right ports initially.
- Waiting for tasks to be RUNNING and ready to accept requests.
- Cold starts added variance to timing.

- Manually tracking chunk/result URLs is clumsy; a small script or workflow tool would help.