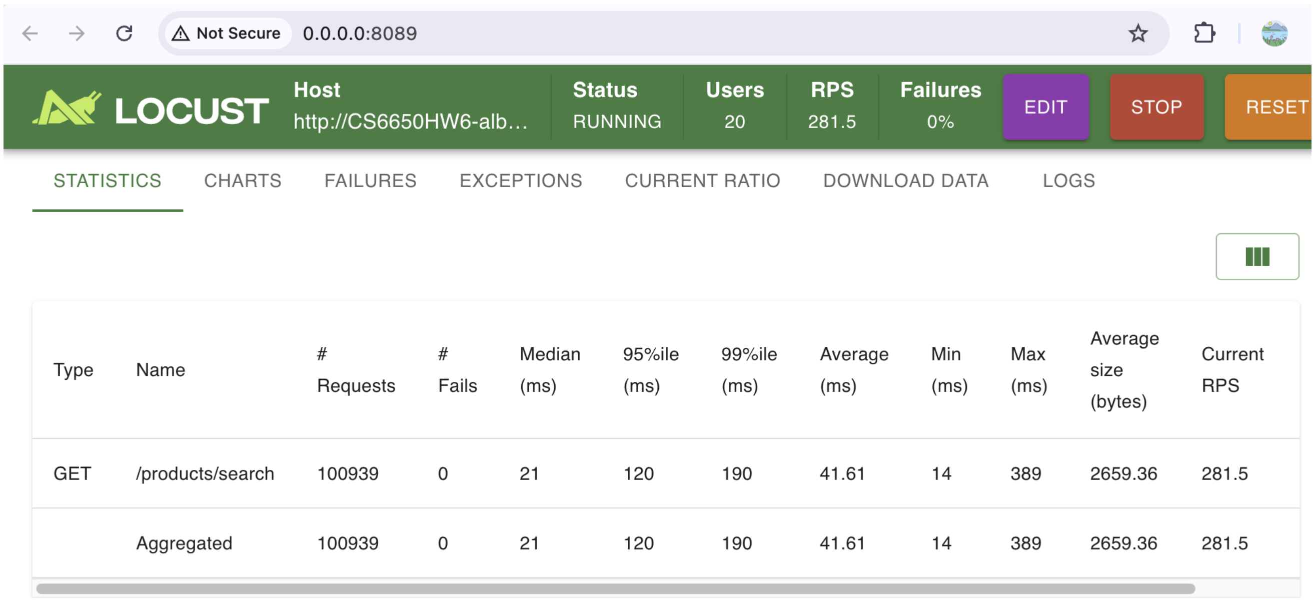
Part3:

Test: 20 users, ramp 20, 2000 products

RPS: 281.5

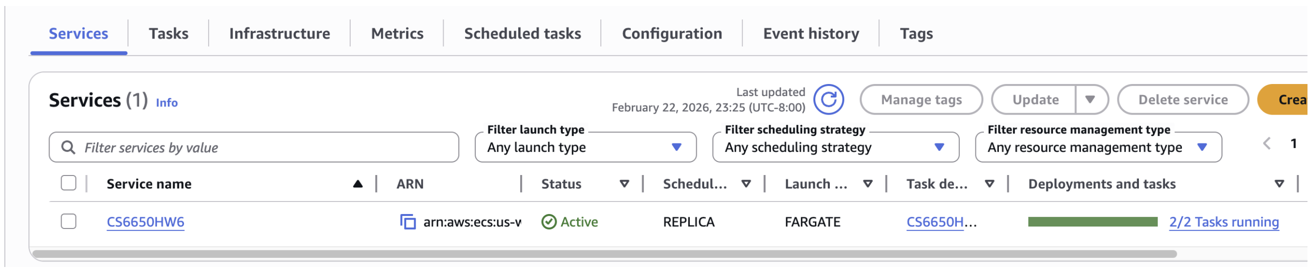


⚠ Not Secure 0.0.0.0:8089

LOCUST | Host: http://CS6650HW6-alb… | Status: RUNNING | Users: 20 | RPS: 281.5 | Failures: 0%

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/products/search	100939	0	21	120	190	41.61	14	389	2659.36	281.5
	Aggregated	100939	0	21	120	190	41.61	14	389	2659.36	281.5

2 Tasks:



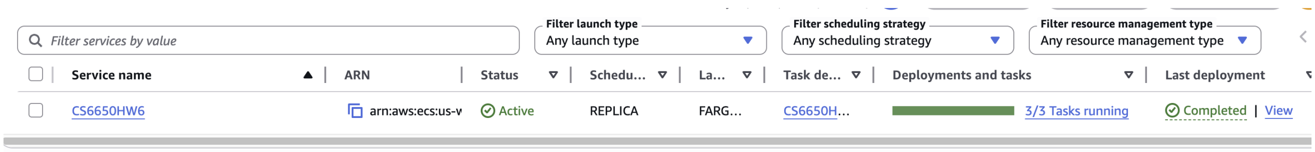
Services (1) Info

Last updated February 22, 2026, 23:25 (UTC-8:00)

Filter services by value | Filter launch type: Any launch type | Filter scheduling strategy: Any scheduling strategy | Filter resource management type: Any resource management type

Service name	ARN	Status	Schedul…	Launch …	Task de…	Deployments and tasks
CS6650HW6	arn:aws:ecs:us-v	Active	REPLICA	FARGATE	CS6650H…	2/2 Tasks running

Then scale up to 3 tasks:



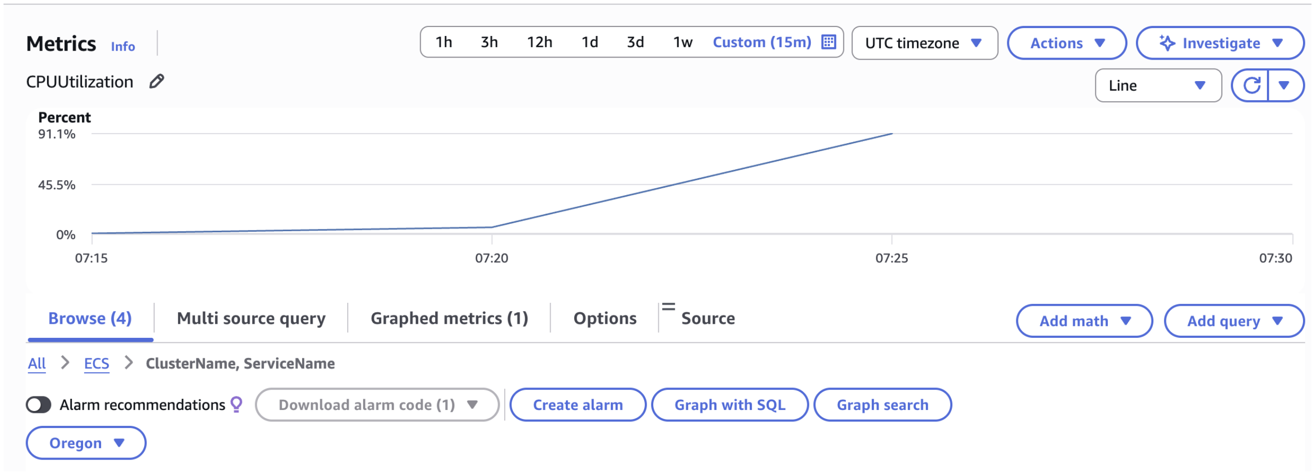
Filter services by value | Filter launch type: Any launch type | Filter scheduling strategy: Any scheduling strategy | Filter resource management type: Any resource management type

Service name	ARN	Status	Schedu…	La…	Task de…	Deployments and tasks	Last deployment
CS6650HW6	arn:aws:ecs:us-v	Active	REPLICA	FARG…	CS6650H…	3/3 Tasks running	Completed | View

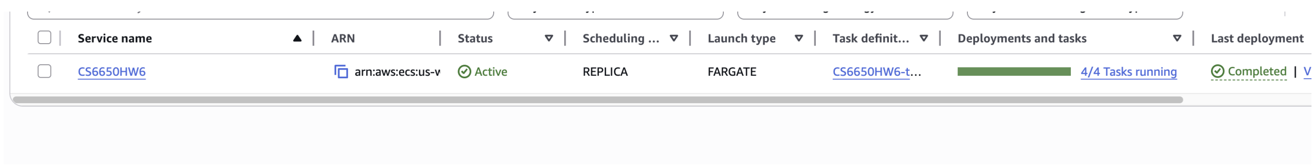
Interestlly it's still almost 100% CPU utilization, **the scaling metric is an average across all tasks, and the new task needs time to warm up.**

When the 3rd task spins up:

1. It takes ~10-20 seconds to start, pass health check, and register in the target group
2. During that time, the original 2 tasks are still handling all traffic at ~100% CPU
3. Even after the 3rd task is healthy, CloudWatch metrics are reported on a **1-minute average** — so the graph still reflects the period when only 2 tasks were working
4. Additionally, 20 users / 3 tasks (each 0.25 vCPU) is still a heavy load — 3 tasks might genuinely not be enough

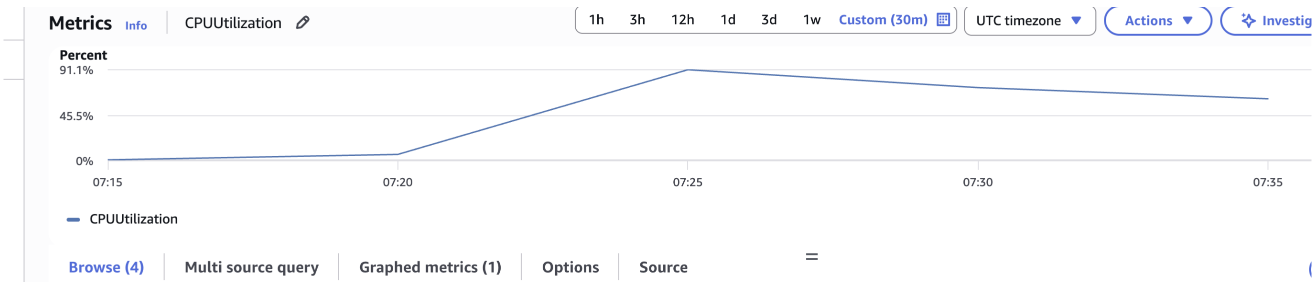


Now 4 tasks:

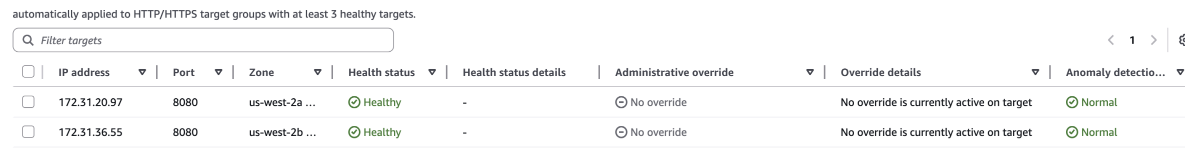


Service name	ARN	Status	Scheduling ...	Launch type	Task definit...	Deployments and tasks	Last deployment
CS6650HW6	arn:aws:ecs:us-v	Active	REPLICA	FARGATE	CS6650HW6-t...	4/4 Tasks running	Completed | V

CPU utilization goes down and become 60%, at this point there's finally enough total CPU (4 × 0.25 = 1 vCPU) to comfortably handle 281 RPS.

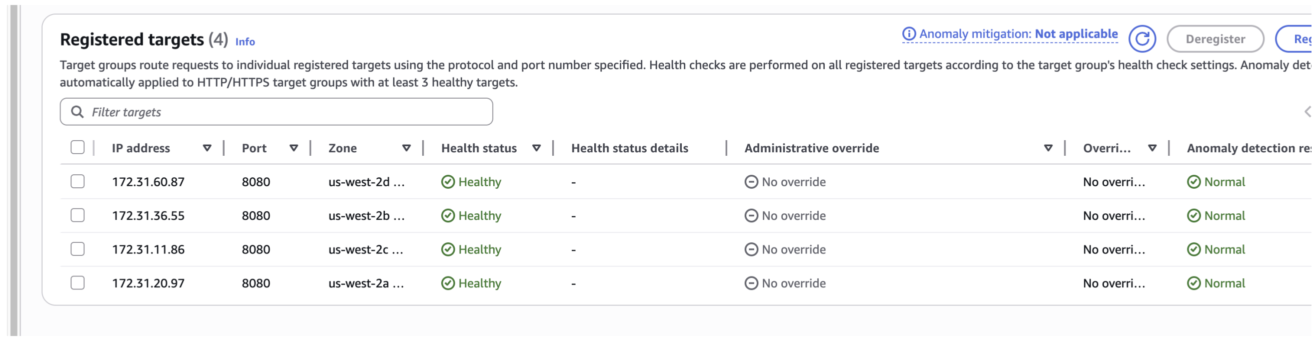


Targets are all healthy



automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

IP address	Port	Zone	Health status	Health status details	Administrative override	Override details	Anomaly detectio...
172.31.20.97	8080	us-west-2a ...	Healthy	-	No override	No override is currently active on target	Normal
172.31.36.55	8080	us-west-2b ...	Healthy	-	No override	No override is currently active on target	Normal



Registered targets (4) Info

Anomaly mitigation: Not applicable

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly det
automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

IP address	Port	Zone	Health status	Health status details	Administrative override	Overri...	Anomaly detection re
172.31.60.87	8080	us-west-2d ...	Healthy	-	No override	No overri...	Normal
172.31.36.55	8080	us-west-2b ...	Healthy	-	No override	No overri...	Normal
172.31.11.86	8080	us-west-2c ...	Healthy	-	No override	No overri...	Normal
172.31.20.97	8080	us-west-2a ...	Healthy	-	No override	No overri...	Normal

Report:

1. How the System Solved the Part II Bottleneck

In Part II, a single ECS Fargate task (0.25 vCPU, 512 MB) hit 100% CPU at 20 users. The bottleneck was CPU — each search request checks 2,000 products, which is a fixed-cost computation that can't be optimized further. Memory stayed flat because all products are loaded once at startup.

Part III solved this by **spreading the same load across multiple tasks**. Instead of one task handling 281 RPS alone, the ALB distributes requests round-robin across 2–4 tasks. Each task only handles a fraction of the total load:

	Part II (1 task)	Part III (14 tasks)
Total CPU available	0.25 vCPU	1.0 vCPU
CPU utilization	~100%	~60%
| RPS	Limited by single task	281.5 sustained
Failure risk	Single point of failture	Any task can die, others continue

The key insight: **the code didn't change at all**. Same Go service, same search logic, same Dockerfile. The only change was infrastructure — more instances behind a load balancer.

2. Role of Each Component

ALB (Application Load Balancer)

- Sits between users and ECS tasks as the single entry point (port 80)
- Distributes incoming requests across all healthy tasks using round-robin
- Users only know one DNS name — they never connect to individual tasks directly
- Handles the HTTP → container port translation (80 → 8080)

Target Group

- Maintains a registry of all ECS task IPs and their health status
- Every 30 seconds, sends a GET request to each task's /health endpoint
- If a task fails 3 consecutive health checks, it's marked unhealthy and removed from rotation — no traffic is sent to it
- When ECS launches a new task, it auto-registers in the target group; after 2 successful health checks, it starts receiving traffic
- As seen in the screenshots: 2 targets (172.31.20.97 and 172.31.56.55) both showing "Healthy" on port 8080 across two availability zones

Auto Scaling

- Monitors the average CPU utilization across all tasks via CloudWatch
- Policy: target 70% average CPU
- If CPU > 70% for 300 seconds → launch a new task (up to max 4)
- If CPU < 70% for 300 seconds → terminate a task (down to min 2)
- As observed: started at 2 tasks → scaled to 3 (CPU still ~91%) → scaled to 4 (CPU dropped to ~60%)

Why CPU was still ~91% at 3 tasks: Three tasks provide 0.75 vCPU total. At 281 RPS with each request checking 2,000

products, 0.75 vCPU is still not enough headroom. Additionally, CloudWatch reports 1-minute averaged metrics, so the
graph reflects the transition period. Only at 4 tasks (1.0 vCPU total) did CPU settle to a comfortable ~60%.

3. Trade-offs: Horizontal vs Vertical Scaling

Aspect: Availability

Horizontal (Part III): If one task dies, others keep serving. ALB routes around failures automatically

Vertical (e.g., 256→512 CPU units): Single instance = single point of failure. If it crashes, service is down

Aspect: Scaling ceiling

Horizontal (Part III): Can add many instances (we capped at 4, but could go to 10, 100...)

Vertical (e.g., 256→512 CPU units): Fargate maxes out at 4 vCPU / 30 GB. Physical hardware has hard limits

Aspect: Cost efficiency

Horizontal (Part III): Pay for what you use — auto scaling removes tasks when load drops

Vertical (e.g., 256→512 CPU units): Paying for peak capacity 24/7, even at 3 AM when nobody is searching

Aspect: Complexity

Horizontal (Part III): More infrastructure: ALB, target group, security groups, scaling policies

Vertical (e.g., 256→512 CPU units): Simple: change one number (CPU: 256 → 512). No new components

Aspect: State management

Horizontal (Part III): Each task has its own in-memory products (100k × 4 = 400k copies in memory)

Vertical (e.g., 256→512 CPU units): One copy of everything. Simpler memory footprint

Aspect: Scaling speed

Horizontal (Part III): New Fargate task takes 30-60 seconds to start + health check. Not instant

Vertical (e.g., 256→512 CPU units): Requires redeployment. Can't scale without downtime

Aspect: Cost

Horizontal (Part III): ALB costs ~$16/month baseline + per-request charges. Multiple tasks = more Fargate cost

Vertical (e.g., 256→512 CPU units): Just pay the difference in CPU units. No ALB cost

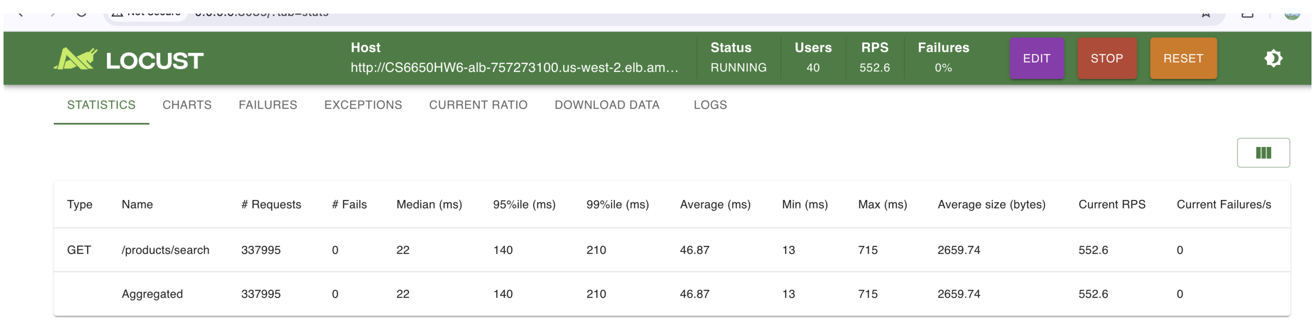
For this workload (CPU-bound fixed-cost computation), horizontal scaling is the right choice because:

- The bottleneck is CPU, which divides perfectly across instances
- There's no shared state — each task loads its own products at startup
- The workload is stateless — any task can handle any request
- Auto scaling matches cost to demand automatically

Vertical scaling would work for a quick fix (doubling CPU from 0.25 to 0.5 vCPU would handle 20 users), but it doesn't provide fault tolerance and eventually hits a ceiling.

4. Predicted Scaling Behavior for Different Load Patterns

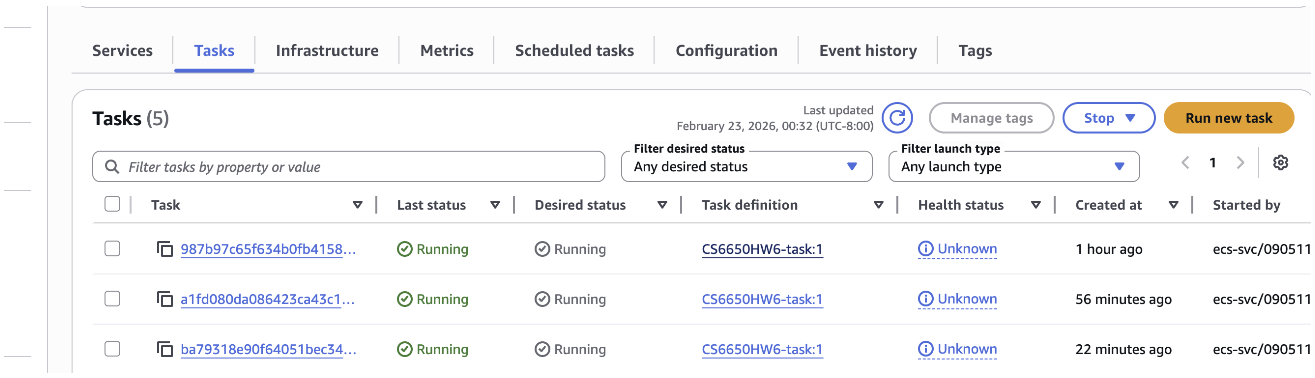
Load Pattern1: 40 users (steady)



Host: http://CS6650HW6-alb-757273100.us-west-2.elb.am…
Status: RUNNING
Users: 40
RPS: 552.6
Failures: 0%

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/products/search	337995	0	22	140	210	46.87	13	715	2659.74	552.6	0
	Aggregated	337995	0	22	140	210	46.87	13	715	2659.74	552.6	0

3 tasks



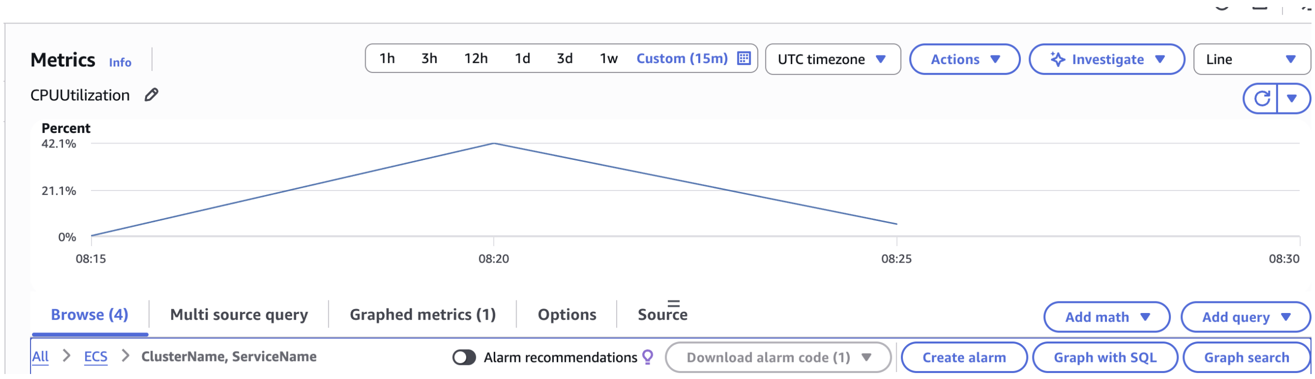
Tasks (5)

Last updated
February 23, 2026, 00:32 (UTC-8:00)

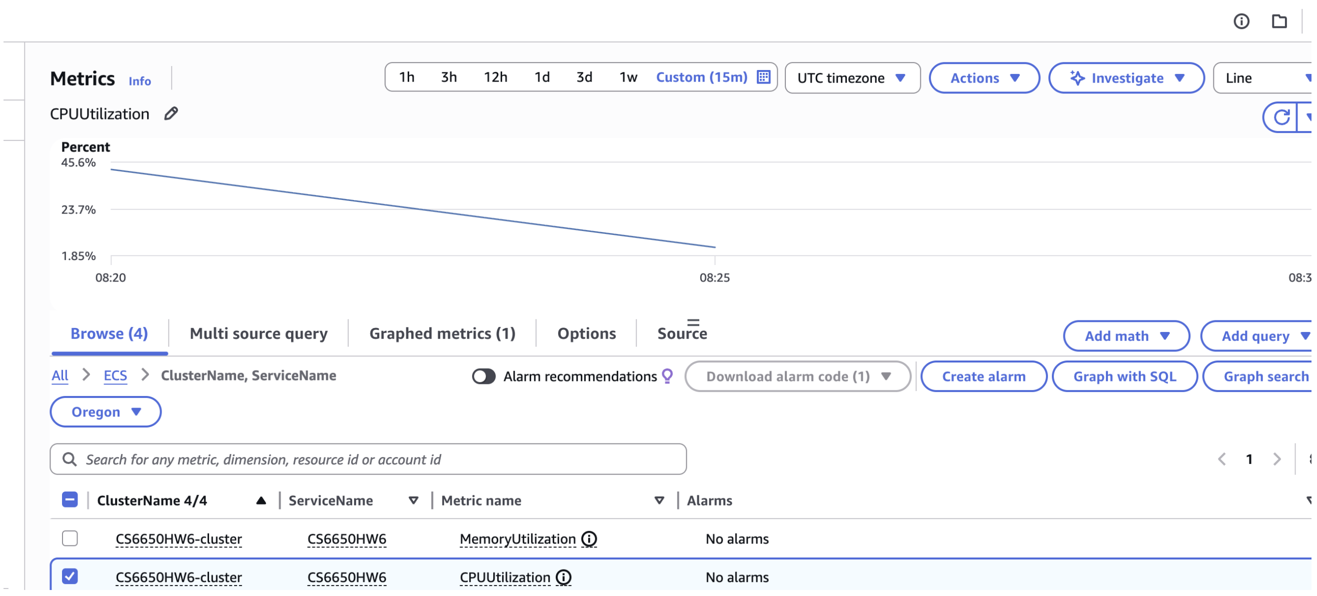
Filter tasks by property or value
Filter desired status: Any desired status
Filter launch type: Any launch type

Task	Last status	Desired status	Task definition	Health status	Created at	Started by
987b97c65f634b0fb4158…	Running	Running	CS6650HW6-task:1	Unknown	1 hour ago	ecs-svc/090511
a1fd080da086423ca43c1…	Running	Running	CS6650HW6-task:1	Unknown	56 minutes ago	ecs-svc/090511
ba79318e90f64051bec34…	Running	Running	CS6650HW6-task:1	Unknown	22 minutes ago	ecs-svc/090511

CPU utilization up to 42%. After previous 20-user test(listed in Load pattern2), auto scaling had ramped up to 4 tasks. When you stopped that test, the 300-second cooldown started. By the time you ran the 40-user test, auto scaling had already **scaled in** — removing one task, leaving 3.



Dropped sigficantlly, down to 5%. The 42% was during the **ramp-up phase** (10 users/sec climbing to 40). Once the test finished or you stopped it, CPU dropped to 5% — that's just the 3 idle tasks doing nothing.


Metrics Info

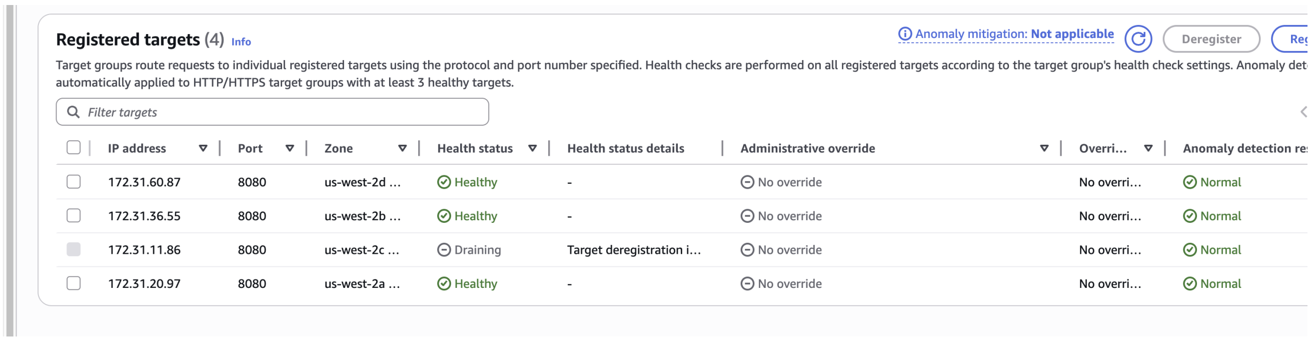
CPUUtilization

ClusterName 4/4	ServiceName	Metric name	Alarms
CS6650HW6-cluster	CS6650HW6	MemoryUtilization	No alarms
✓ CS6650HW6-cluster	CS6650HW6	CPUUtilization	No alarms

Overall:

 This seems low for 40 users. The likely explanation: **your 3 tasks had already warmed up and the ALB was distributing efficiently**. With 3 tasks × 0.25 vCPU = 0.75 vCPU, and each request only checking 2,000 products (not 20,000), the per-request CPU cost is actually small. At 553 RPS across 3 tasks, that's ~184 RPS per task — fast enough that CPU doesn't saturate.

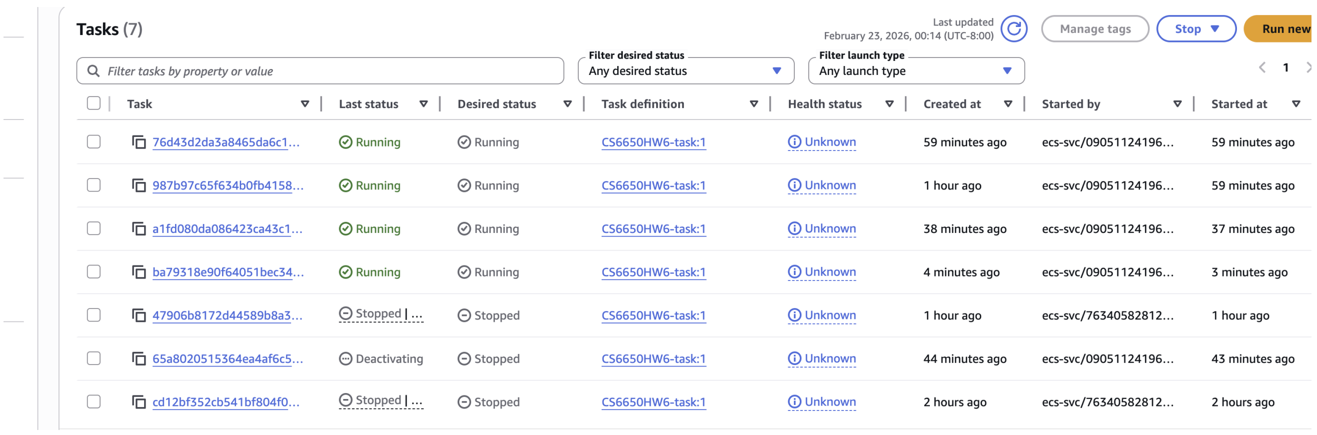
Load Pattern2: Kill a task during load


Registered targets (4) Info

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly det
automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

IP address	Port	Zone	Health status	Health status details	Administrative override	Overri...	Anomaly detection re
172.31.60.87	8080	us-west-2d ...	Healthy	-	No override	No overri...	Normal
172.31.36.55	8080	us-west-2b ...	Healthy	-	No override	No overri...	Normal
172.31.11.86	8080	us-west-2c ...	Draining	Target deregistration i...	No override	No overri...	Normal
172.31.20.97	8080	us-west-2a ...	Healthy	-	No override	No overri...	Normal

ECS auto-replaced the task — the task list shows 4 tasks still running after the kill, but one of them (ba79318e...) is new — it wasn't in the original list. ECS automatically launched a replacement to maintain the desired count.


Tasks (7)

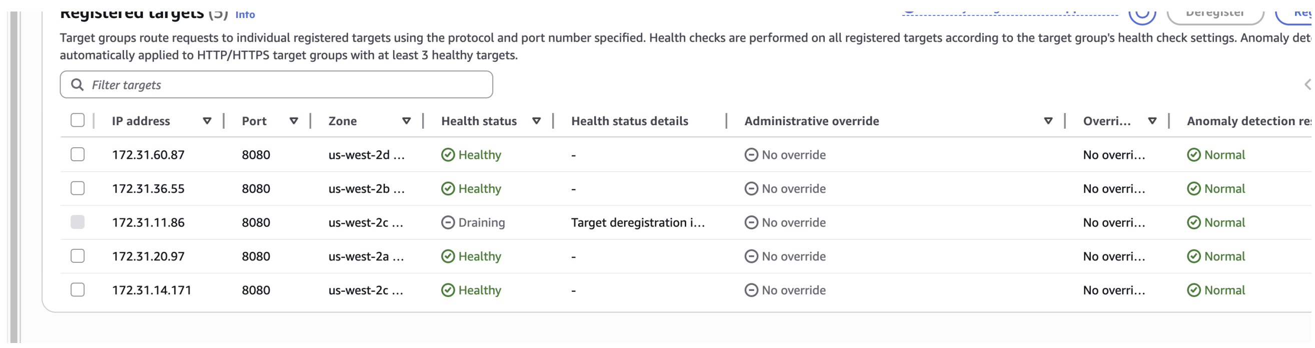
Last updated February 23, 2026, 00:14 (UTC-8:00)

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
76d43d2da3a8465da6c1...	Running	Running	CS6650HW6-task:1	Unknown	59 minutes ago	ecs-svc/09051124196...	59 minutes ago
987b97c65f634b0fb4158...	Running	Running	CS6650HW6-task:1	Unknown	1 hour ago	ecs-svc/09051124196...	59 minutes ago
a1fd080da086423ca43c1...	Running	Running	CS6650HW6-task:1	Unknown	38 minutes ago	ecs-svc/09051124196...	37 minutes ago
ba79318e90f64051bec34...	Running	Running	CS6650HW6-task:1	Unknown	4 minutes ago	ecs-svc/09051124196...	3 minutes ago
47906b8172d44589b8a3...	Stopped | ...	Stopped	CS6650HW6-task:1	Unknown	1 hour ago	ecs-svc/76340582812...	1 hour ago
65a8020515364ea4af6c5...	Deactivating	Stopped	CS6650HW6-task:1	Unknown	44 minutes ago	ecs-svc/09051124196...	43 minutes ago
cd12bf352cb541bf804f0...	Stopped | ...	Stopped	CS6650HW6-task:1	Unknown	2 hours ago	ecs-svc/76340582812...	2 hours ago

 Predicted Behavior: ALB detects unhealthy target within 90s (3 × 30s health check interval). Remaining tasks absorb load. Auto scaling launches replacement. Brief spike in response times.

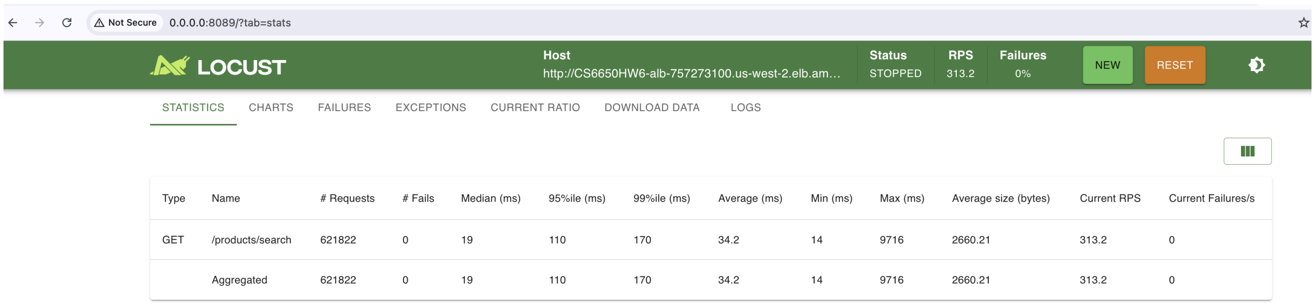


Registered targets (5) Info

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly det automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

IP address	Port	Zone	Health status	Health status details	Administrative override	Overri...	Anomaly detection re
172.31.60.87	8080	us-west-2d ...	✓ Healthy	-	No override	No overri...	✓ Normal
172.31.36.55	8080	us-west-2b ...	✓ Healthy	-	No override	No overri...	✓ Normal
172.31.11.86	8080	us-west-2c ...	Draining	Target deregistration i...	No override	No overri...	✓ Normal
172.31.20.97	8080	us-west-2a ...	✓ Healthy	-	No override	No overri...	✓ Normal
172.31.14.171	8080	us-west-2c ...	✓ Healthy	-	No override	No overri...	✓ Normal

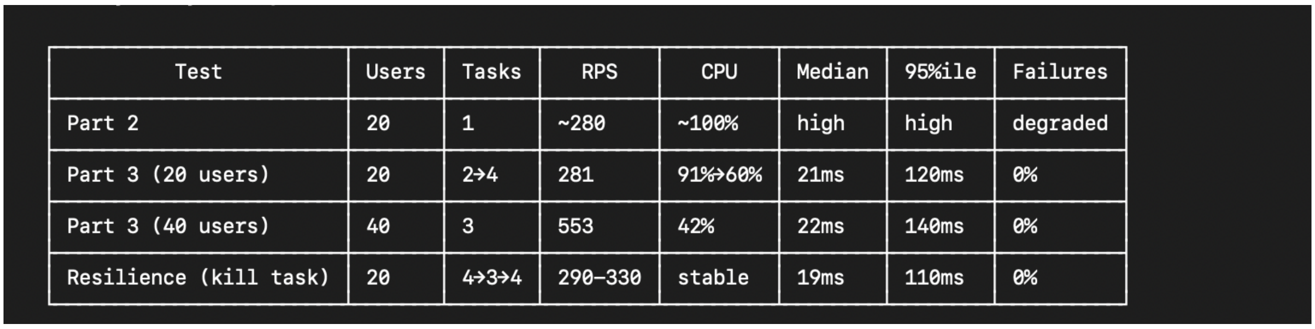
Zero failures — despite killing a task mid-test, not a single request failed. The ALB detected the dead task and stopped routing to it.



Host http://CS6650HW6-alb-757273100.us-west-2.elb.am... Status STOPPED RPS 313.2 Failures 0%

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/products/search	621822	0	19	110	170	34.2	14	9716	2660.21	313.2	0
	Aggregated	621822	0	19	110	170	34.2	14	9716	2660.21	313.2	0

Summary for the report



Test	Users	Tasks	RPS	CPU	Median	95%ile	Failures
Part 2	20	1	~280	~100%	high	high	degraded
Part 3 (20 users)	20	2→4	281	91%→60%	21ms	120ms	0%
Part 3 (40 users)	40	3	553	42%	22ms	140ms	0%
Resilience (kill task)	20	4→3→4	290-330	stable	19ms	110ms	0%

The 40-user test actually proves that **the system had plenty of headroom** — it didn't even need to scale to 4 tasks.

This is a good result: horizontal scaling handled double the load that broke Part 2, without breaking a sweat.