📦 Amazon ECS (Elastic Container Service)

Amazon ECS

Amazon ECS is a fully managed container orchestration service 🧠🚢. It helps you **run and manage Docker containers** on a cluster of EC2 instances or using serverless (Fargate).

- No need to install your own container orchestration software.
- Integrates with **Elastic Load Balancing**, **IAM**, **CloudWatch**, and more.
- Two launch types: **EC2** and **Fargate**.

📌 **Exam Tip:** ECS is AWS's **proprietary** alternative to Kubernetes (which is EKS). Know when to use ECS vs EKS.

🧩 What are ECS Tasks?

A **Task** is the **basic unit of work** in ECS 🛠️.

- A task is launched from a **Task Definition** (like a recipe 📄 for containers).
- A task can run one or more containers.
- You define CPU, memory, networking, and IAM roles in the task definition.

📌 **Exam Tip:** Tasks run your containers. They're often asked about in context of Task Definitions and Service Auto Scaling.

⚙️ ECS - EC2 Launch Type (Features in short)

- You manage the EC2 instances yourself 🧑‍💻.
- You pay for the EC2 instances even if they're underutilized.
- Best when you need **deep control** over the infrastructure (e.g., custom AMIs, logging agents).
- Use **Auto Scaling Groups** to scale ECS capacity.

📌 **Exam Tip:** Choose **EC2 launch type** when you want **OS-level control** or have **steady workloads**.

☁️ ECS - Fargate Launch Type (Features in short)

- **Serverless** container hosting 💨🧳 — no need to manage EC2 instances.
- You only pay for **CPU and memory** used by containers.
- Easily scalable and **great for microservices**.

📌 **Exam Tip:** Fargate = no server management. Used when you want simplicity and **cost-efficient scaling**.

🔐 IAM Roles for ECS (In Short)

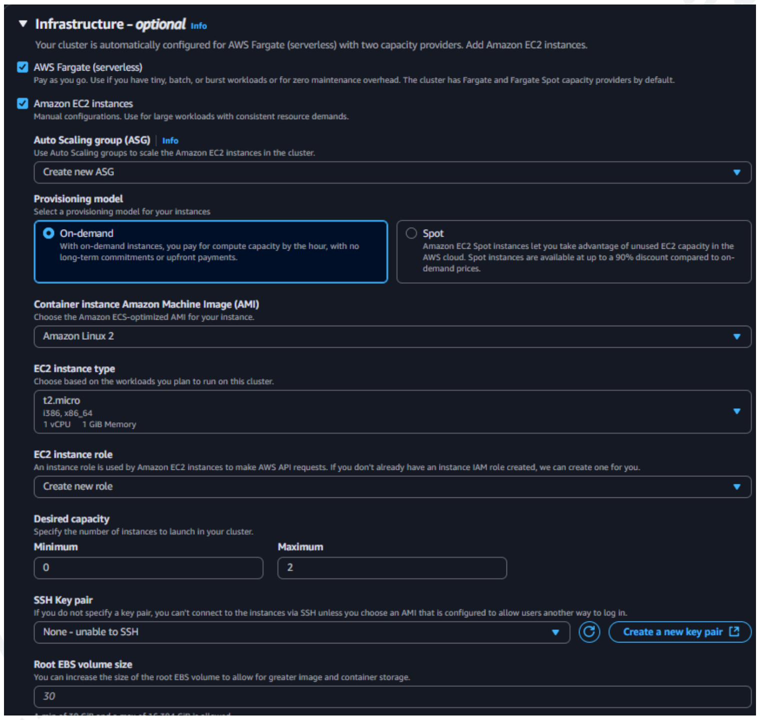
1. **Task Role** 🧾: Assigned to the **ECS task** so containers can access AWS services (e.g., S3, DynamoDB).
2. **Execution Role** 🚀: Used by ECS to **pull container images** from ECR and **write logs to CloudWatch**.
3. **Container Instance Role** (for EC2 launch type) 🖥️: Lets EC2 instances register to ECS and talk to other AWS services.

📌 **Exam Tip:** Understand the difference between **Task Role vs Execution Role**.

🚀 Create ECS Cluster (Fargate + EC2)

1. **Go to ECS > Clusters > Create Cluster**
2. **Name** your cluster: `DemoCluster` 🏷️
3. **Select Capacity Providers**:
   - ✅ AWS Fargate (serverless)
   - ✅ Amazon EC2 -Create Auto Scaling Group

   Infrastructure :-



Infrastructure - *optional* Info

Your cluster is automatically configured for AWS Fargate (serverless) with two capacity providers. Add Amazon EC2 instances.

[✓] AWS Fargate (serverless)
Pay as you go. Use if you have tiny, batch, or burst workloads or for zero maintenance overhead. The cluster has Fargate and Fargate Spot capacity providers by default.

[✓] Amazon EC2 instances
Manual configurations. Use for large workloads with consistent resource demands.

Auto Scaling group (ASG) | Info
Use Auto Scaling groups to scale the Amazon EC2 instances in the cluster.
Create new ASG

Provisioning model
Select a provisioning model for your instances
[✓ On-demand] With on-demand instances, you pay for compute capacity by the hour, with no long-term commitments or upfront payments.
Spot: Amazon EC2 Spot instances let you take advantage of unused EC2 capacity in the AWS cloud. Spot instances are available at up to a 90% discount compared to on-demand prices.

Container instance Amazon Machine Image (AMI)
Choose the Amazon ECS-optimized AMI for your instance.
Amazon Linux 2

EC2 instance type
Choose based on the workloads you plan to run on this cluster.
t2.micro
i386, x86_64
1 vCPU 1 GiB Memory

EC2 instance role
An instance role is used by Amazon EC2 instances to make AWS API requests. If you don't already have an instance IAM role created, we can create one for you.
Create new role

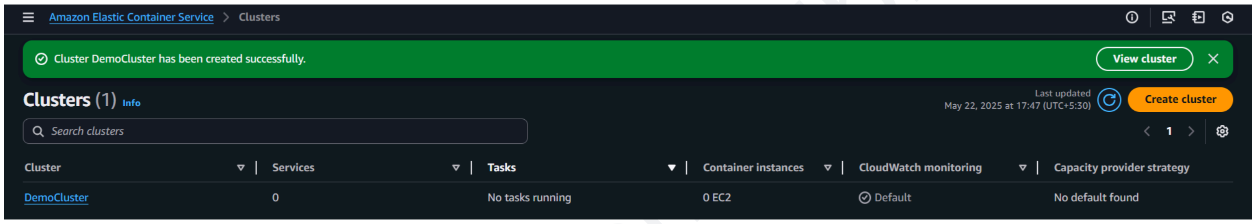
Desired capacity
Specify the number of instances to launch in your cluster.
Minimum: 0
Maximum: 2

SSH Key pair
If you do not specify a key pair, you can't connect to the instances via SSH unless you choose an AMI that is configured to allow users another way to log in.
None - unable to SSH

Root EBS volume size
You can increase the size of the root EBS volume to allow for greater image and container storage.
30

4. **Keep Network settings default** 🌐
5. **Provisioning**:
   - Choose **On-demand** or **Spot** ⚡

6. **EC2 Settings**:
   - AMI: Amazon Linux 2 🐧
   - Instance type: `t2.micro` 💻
   - IAM role: Create one if needed 🔐
   - Capacity: Min `0`, Max `2`
7. **Key Pair**: Create one if you want SSH access 🔑
8. **EBS Volume**: Default 30 GB 💽
9. Click **Create** ✅



Amazon Elastic Container Service > Clusters

Cluster DemoCluster has been created successfully.

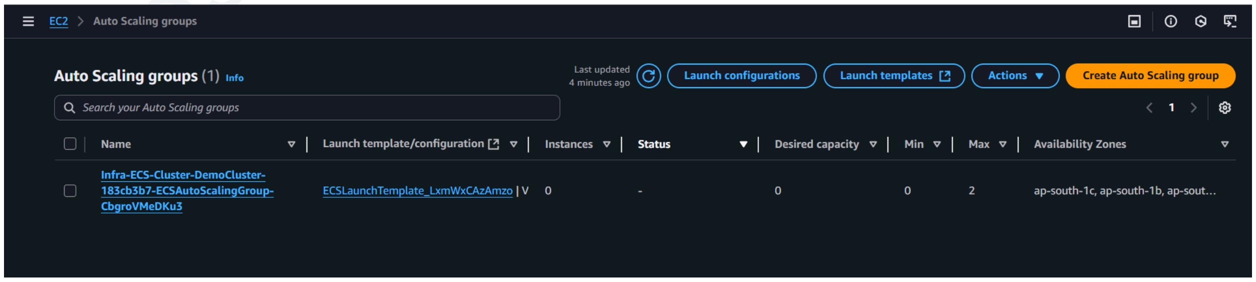
Clusters (1) Info

Last updated May 22, 2025 at 17:47 (UTC+5:30)

Cluster	Services	Tasks	Container instances	CloudWatch monitoring	Capacity provider strategy
DemoCluster	0	No tasks running	0 EC2	Default	No default found

🔍 Check ASG for ECS Cluster

1. Go to **EC2 Console → Auto Scaling Groups** 📊
2. Look for ASG named like `ECS-Cluster-DemoCluster-...`
3. Verify:
   - Instance type ✅
   - Capacity (e.g., 0–2) 📈
   VPC/Subnet 🌐

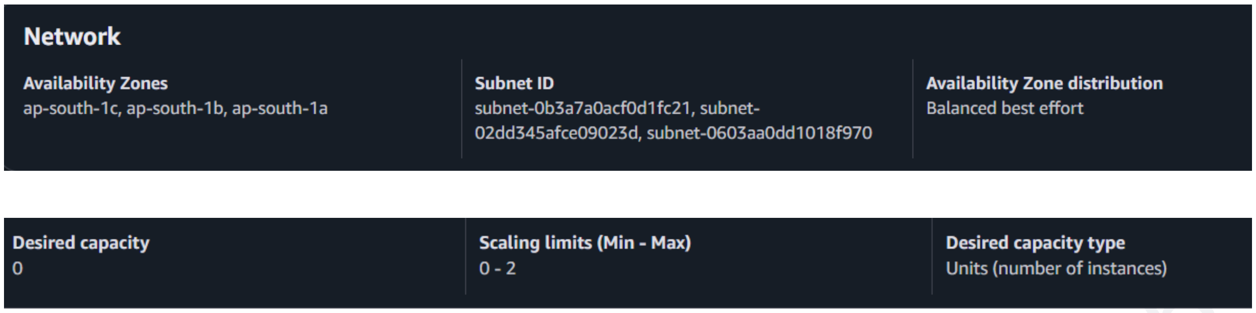


EC2 > Auto Scaling groups

Auto Scaling groups (1) Info

Last updated 4 minutes ago

Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones
Infra-ECS-Cluster-DemoCluster-183cb3b7-ECSAutoScalingGroup-CbgroVMeDKu3	ECSLaunchTemplate_LxmWxCAzAmzo | V	0	-	0	0	2	ap-south-1c, ap-south-1b, ap-sout...

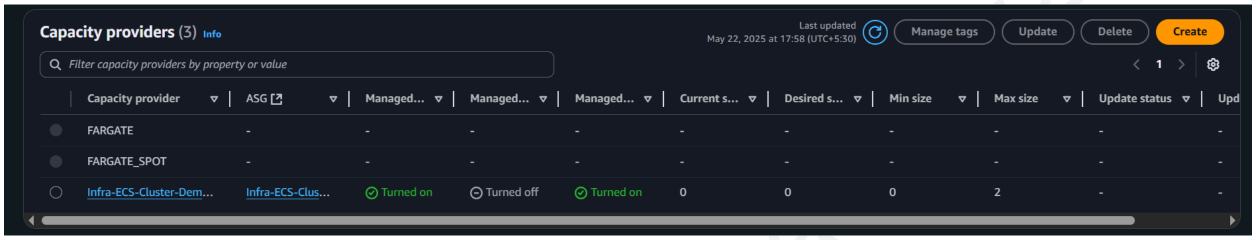


Network

Availability Zones	Subnet ID	Availability Zone distribution
ap-south-1c, ap-south-1b, ap-south-1a	subnet-0b3a7a0acf0d1fc21, subnet-02dd345afce09023d, subnet-0603aa0dd1018f970	Balanced best effort

Desired capacity	Scaling limits (Min - Max)	Desired capacity type
0	0 - 2	Units (number of instances)

Go to **ECS > Clusters > DemoCluster > Infrastructure tab** to see attached **Capacity Providers** like `FARGATE` and `EC2` ✅.



Capacity providers (3) Info

Last updated May 22, 2025 at 17:58 (UTC+5:30)

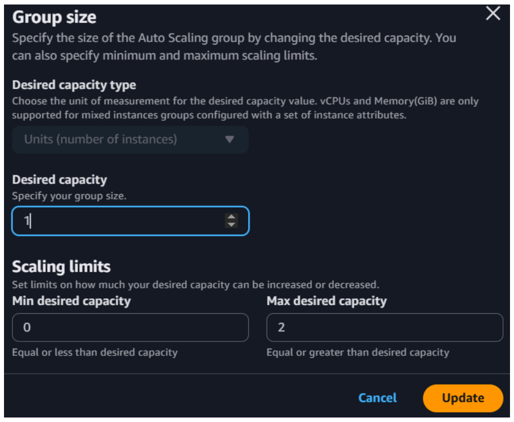
Capacity provider	ASG	Managed...	Managed...	Managed...	Current s...	Desired s...	Min size	Max size	Update status	Upd
FARGATE	-	-	-	-	-	-	-	-	-	-
FARGATE_SPOT	-	-	-	-	-	-	-	-	-	-
Infra-ECS-Cluster-Dem...	Infra-ECS-Clus...	Turned on	Turned off	Turned on	0	0	0	2	-	-

Capacity Providers in ECS define **where and how your tasks run** — on **Fargate**, **EC2**, or **Spot** 💡.

- `FARGATE` → Serverless, no EC2 to manage 🚀
- `EC2` → Use your own EC2 instances 🖥️
- `FARGATE_SPOT` → Run on spare capacity, cheaper 💰

Change ASG Desired Capacity from 0 to 1 🚀

- Set desired capacity to 1 in your Auto Scaling Group.
- AWS launches a new EC2 instance. 🖥️
- The instance automatically **registers itself** to the ECS Cluster. Check in **Container Instances** 🤝



Group size

Specify the size of the Auto Scaling group by changing the desired capacity. You can also specify minimum and maximum scaling limits.

Desired capacity type

Choose the unit of measurement for the desired capacity value. vCPUs and Memory(GiB) are only supported for mixed instances groups configured with a set of instance attributes.

Units (number of instances)

Desired capacity

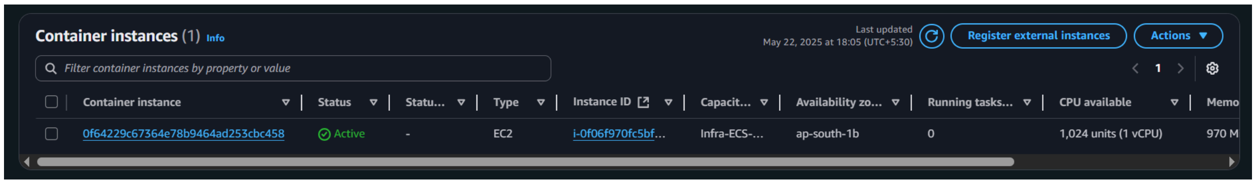
Specify your group size.

1

Scaling limits

Set limits on how much your desired capacity can be increased or decreased.

Min desired capacity	Max desired capacity
0	2
Equal or less than desired capacity	Equal or greater than desired capacity


Container instances (1) Info

Last updated May 22, 2025 at 18:05 (UTC+5:30)

Register external instances
Actions
Filter container instances by property or value

Container instance	Status	Statu...	Type	Instance ID	Capacit...	Availability zo...	Running tasks...	CPU available	Memo
0f64229c67364e78b9464ad253cbc458	Active	-	EC2	i-0f06f970fc5bf...	Infra-ECS-...	ap-south-1b	0	1,024 units (1 vCPU)	970 M

🚀 Create a New Task Definition in ECS (Fargate) — Example with `nginxdemos/hello` 🎯

This example will walk you through creating a simple **ECS Task Definition** using **Fargate**, which runs a container from the image `nginxdemos/hello`. This is perfect for testing or learning the ECS basics!

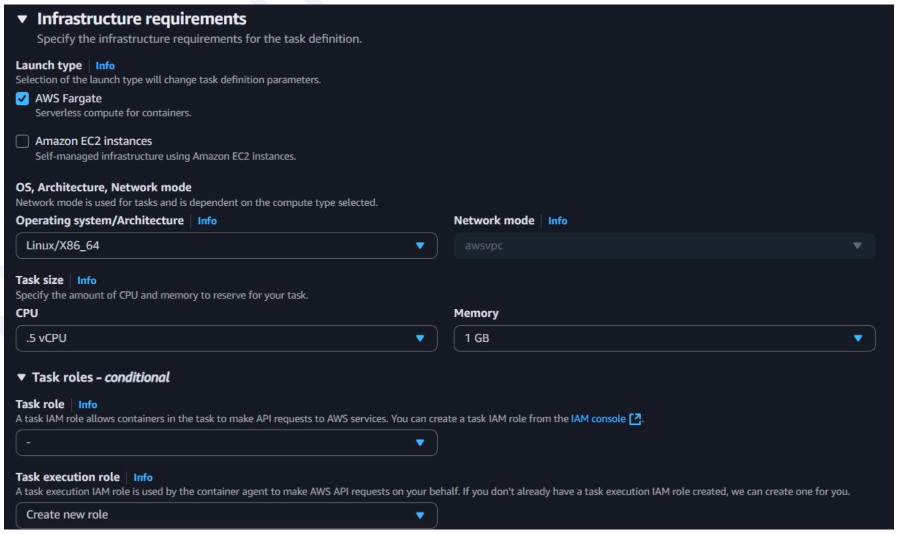
🧱 Step-by-Step Guide

1️⃣ Create a New Task Definition

- Go to **Amazon ECS Console**
- Choose **Task Definitions** → **Create new task definition —> Give Name**
- Select **FARGATE** as the launch type (✅ serverless)

2️⃣ Infrastructure Requirements

- **Operating system/Architecture**: `Linux/X86_64` (default)
- **Network mode**: `awsvpc` (auto-selected for Fargate)
- **Task Size**:
  - **CPU**: `0.5 vCPU`
  - **Memory**: `1 GB`
    (You can change this as per your workload needs)



▼ Infrastructure requirements
Specify the infrastructure requirements for the task definition.

Launch type | Info
Selection of the launch type will change task definition parameters.
[✓] AWS Fargate
Serverless compute for containers.
[ ] Amazon EC2 instances
Self-managed infrastructure using Amazon EC2 instances.

OS, Architecture, Network mode
Network mode is used for tasks and is dependent on the compute type selected.
Operating system/Architecture | Info: Linux/X86_64
Network mode | Info: awsvpc

Task size | Info
Specify the amount of CPU and memory to reserve for your task.
CPU: .5 vCPU
Memory: 1 GB

▼ Task roles - *conditional*

Task role | Info
A task IAM role allows containers in the task to make API requests to AWS services. You can create a task IAM role from the IAM console.
-

Task execution role | Info
A task execution IAM role is used by the container agent to make AWS API requests on your behalf. If you don't already have a task execution IAM role created, we can create one for you.
Create new role

3️⃣ Task Roles

- **Task Role**: Leave blank if your container doesn't need to call AWS services.
- **Task Execution Role**:
  - Click **Create new role** (ECS will create a role with permissions to pull the image and log to CloudWatch)

4️⃣ Add Container

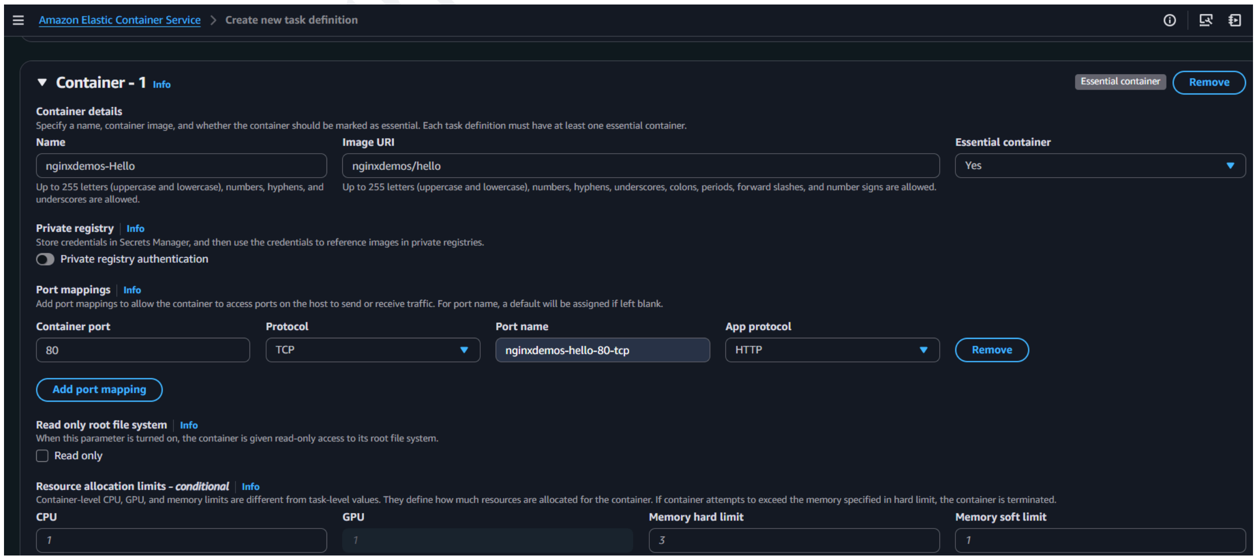
- Click **Add container**
- **Container name**: `hello-container` *(or any name you like)*

Image URI:

bash
CopyEdit
```
nginxdemos/hello
```

- 👉 This image is hosted on Docker Hub: https://hub.docker.com/r/nginxdemos/hello/

- **Port mappings**:
  - Container port: `80`
  - Protocol: `tcp`


Amazon Elastic Container Service > Create new task definition
Essential container
Remove
Add port mapping
Remove

Container - 1 Info

Container details
Specify a name, container image, and whether the container should be marked as essential. Each task definition must have at least one essential container.

Name	Image URI	Essential container
nginxdemos-Hello	nginxdemos/hello	Yes

Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.
Up to 255 letters (uppercase and lowercase), numbers, hyphens, underscores, colons, periods, forward slashes, and number signs are allowed.

Private registry | Info
Store credentials in Secrets Manager, and then use the credentials to reference images in private registries.
Private registry authentication

Port mappings | Info
Add port mappings to allow the container to access ports on the host to send or receive traffic. For port name, a default will be assigned if left blank.

Container port	Protocol	Port name	App protocol
80	TCP	nginxdemos-hello-80-tcp	HTTP

Read only root file system | Info
When this parameter is turned on, the container is given read-only access to its root file system.
☐ Read only

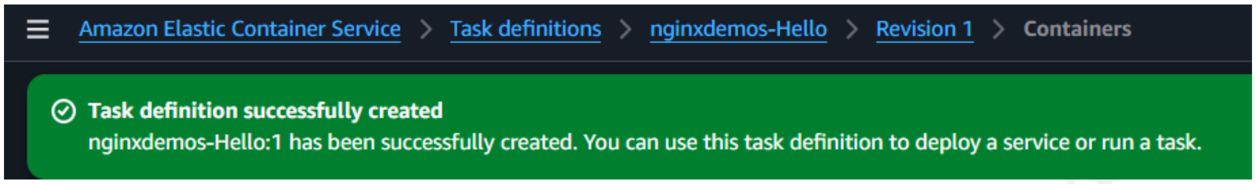
Resource allocation limits - *conditional* | Info
Container-level CPU, GPU, and memory limits are different from task-level values. They define how much resources are allocated for the container. If container attempts to exceed the memory specified in hard limit, the container is terminated.

CPU	GPU	Memory hard limit	Memory soft limit
1	1	3	1

✅ **Leave all other settings as default**

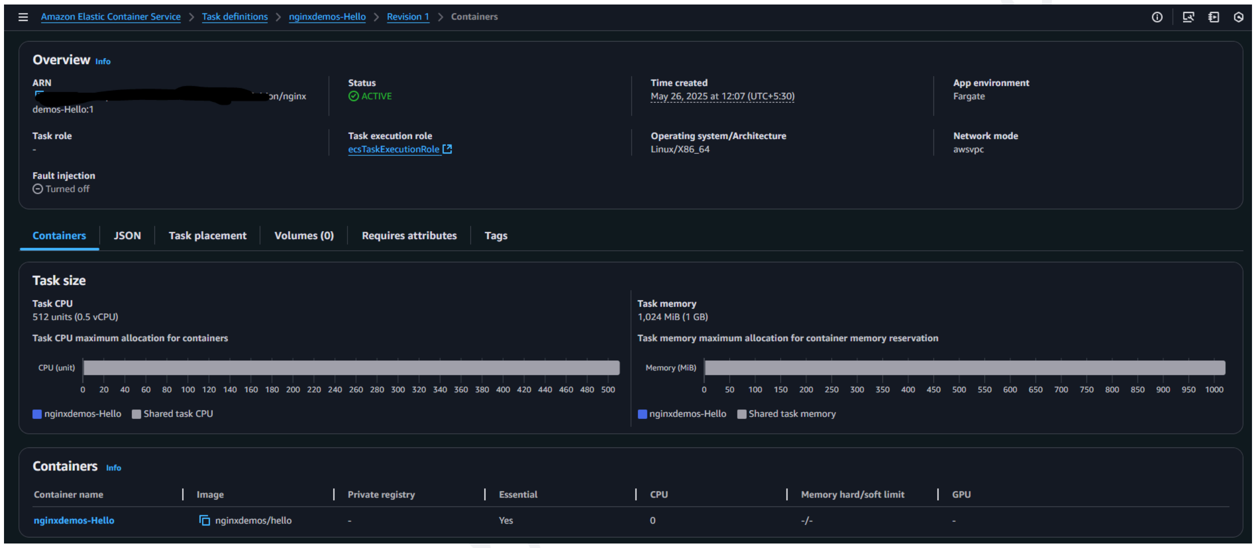
5️⃣ Review and Create

- Review your configuration
- Click **Create**
- You now have a task definition ready to be used in a service or for a one-time task run!



Amazon Elastic Container Service > Task definitions > nginxdemos-Hello > Revision 1 > Containers

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



Amazon Elastic Container Service > Task definitions > nginxdemos-Hello > Revision 1 > Containers

Overview Info

ARN: [redacted]on/nginxdemos-Hello:1	Status: ACTIVE	Time created: May 26, 2025 at 12:07 (UTC+5:30)	App environment: Fargate
Task role: -	Task execution role: ecsTaskExecutionRole	Operating system/Architecture: Linux/X86_64	Network mode: awsvpc
Fault injection: Turned off			

Containers | JSON | Task placement | Volumes (0) | Requires attributes | Tags

Task size

Task CPU
512 units (0.5 vCPU)

Task CPU maximum allocation for containers

Task memory
1,024 MiB (1 GB)

Task memory maximum allocation for container memory reservation

Containers Info

Container name	Image	Private registry	Essential	CPU	Memory hard/soft limit	GPU
nginxdemos-Hello	nginxdemos/hello	-	Yes	0	-/-	-

🚀 Launch ECS Task Definition as a Service (Fargate + ALB)

1️⃣ Go to ECS → Clusters → Select DemoCluster

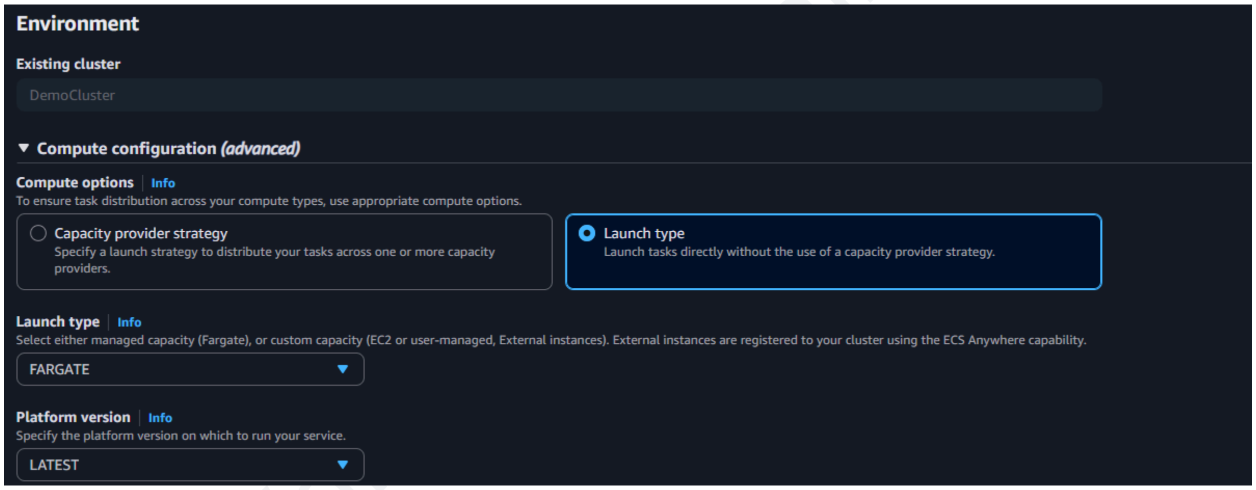
Click **Services** → then **Add Service**

2️⃣ Service Details

- **Launch type**: Select `Launch Type` (🟦 FARGATE)
- **Deployment type**: `Replica`
- Keep other values as **default**

3️⃣ Environment Configuration

- Cluster: DemoCluster
- Launch type: FARGATE
- Platform version: LATEST


Environment

Existing cluster
DemoCluster

▼ **Compute configuration *(advanced)***

Compute options | Info
To ensure task distribution across your compute types, use appropriate compute options.

○ Capacity provider strategy
Specify a launch strategy to distribute your tasks across one or more capacity providers.

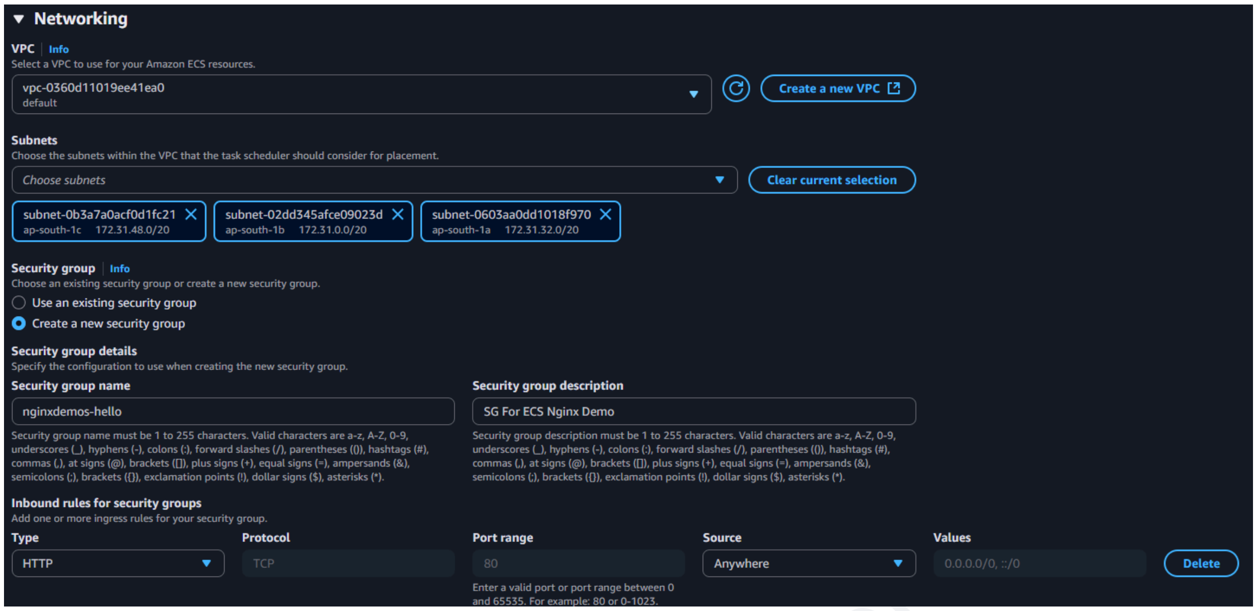
● Launch type
Launch tasks directly without the use of a capacity provider strategy.

Launch type | Info
Select either managed capacity (Fargate), or custom capacity (EC2 or user-managed, External instances). External instances are registered to your cluster using the ECS Anywhere capability.
FARGATE

Platform version | Info
Specify the platform version on which to run your service.
LATEST

4️⃣ Networking

- **VPC**: Select your VPC
- **Subnets**: Select **multiple subnets** (high availability across AZs)
- **Security group**:
  - Select Create a new security group
  - Name: nginxdemos-hello
  - Description: SG For ECS Nginx Demo
  - Inbound rule:
    - Type: HTTP
    - Port: 80
    - Source: Anywhere (0.0.0.0/0 and ::/0)



Networking

VPC | Info
Select a VPC to use for your Amazon ECS resources.
vpc-0360d11019ee41ea0
default

Subnets
Choose the subnets within the VPC that the task scheduler should consider for placement.
Choose subnets
- subnet-0b3a7a0acf0d1fc21 — ap-south-1c 172.31.48.0/20
- subnet-02dd345afce09023d — ap-south-1b 172.31.0.0/20
- subnet-0603aa0dd1018f970 — ap-south-1a 172.31.32.0/20

Security group | Info
Choose an existing security group or create a new security group.
Use an existing security group [✓ Create a new security group]

Security group details
Specify the configuration to use when creating the new security group.

Security group name
nginxdemos-hello
Security group name must be 1 to 255 characters. Valid characters are a-z, A-Z, 0-9, underscores (_), hyphens (-), colons (:), forward slashes (/), parentheses (()), hashtags (#), commas (,), at signs (@), brackets ([]), plus signs (+), equal signs (=), ampersands (&), semicolons (;), brackets ({}), exclamation points (!), dollar signs ($), asterisks (*).

Security group description
SG For ECS Nginx Demo
Security group description must be 1 to 255 characters. Valid characters are a-z, A-Z, 0-9, underscores (_), hyphens (-), colons (:), forward slashes (/), parentheses (()), hashtags (#), commas (,), at signs (@), brackets ([]), plus signs (+), equal signs (=), ampersands (&), semicolons (;), brackets ({}), exclamation points (!), dollar signs ($), asterisks (*).

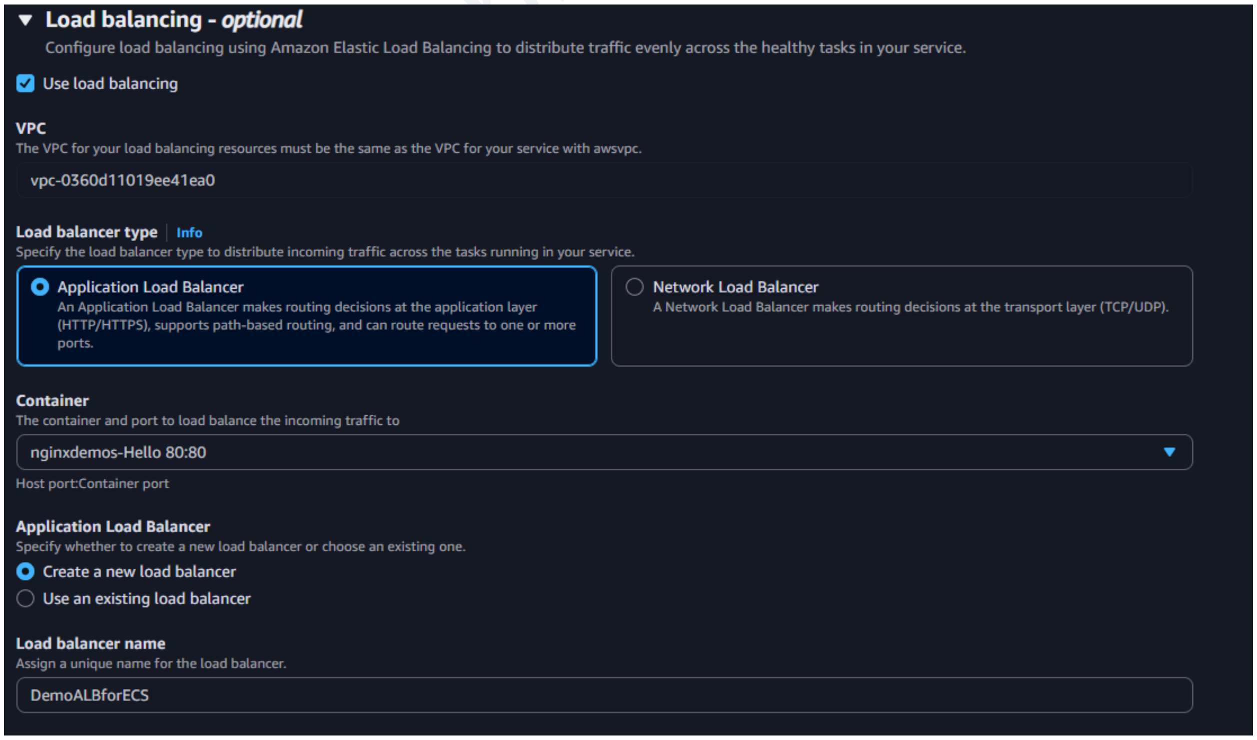
Inbound rules for security groups
Add one or more ingress rules for your security group.

Type	Protocol	Port range	Source	Values
HTTP	TCP	80	Anywhere	0.0.0.0/0, ::/0

Enter a valid port or port range between 0 and 65535. For example: 80 or 0-1023.

5️⃣ Load Balancing

- ✅ Enable **"Use load balancing"**
- **Load balancer type**: Application Load Balancer
- **Container to load balance**:
    - Choose nginxdemos-Hello 80:80
- **Create a new load balancer**:
    - Name: DemoALBforECS



Load balancing - *optional*
Configure load balancing using Amazon Elastic Load Balancing to distribute traffic evenly across the healthy tasks in your service.

[✓ Use load balancing]

VPC
The VPC for your load balancing resources must be the same as the VPC for your service with awsvpc.
vpc-0360d11019ee41ea0

Load balancer type | Info
Specify the load balancer type to distribute incoming traffic across the tasks running in your service.
[✓ Application Load Balancer] An Application Load Balancer makes routing decisions at the application layer (HTTP/HTTPS), supports path-based routing, and can route requests to one or more ports.
Network Load Balancer — A Network Load Balancer makes routing decisions at the transport layer (TCP/UDP).

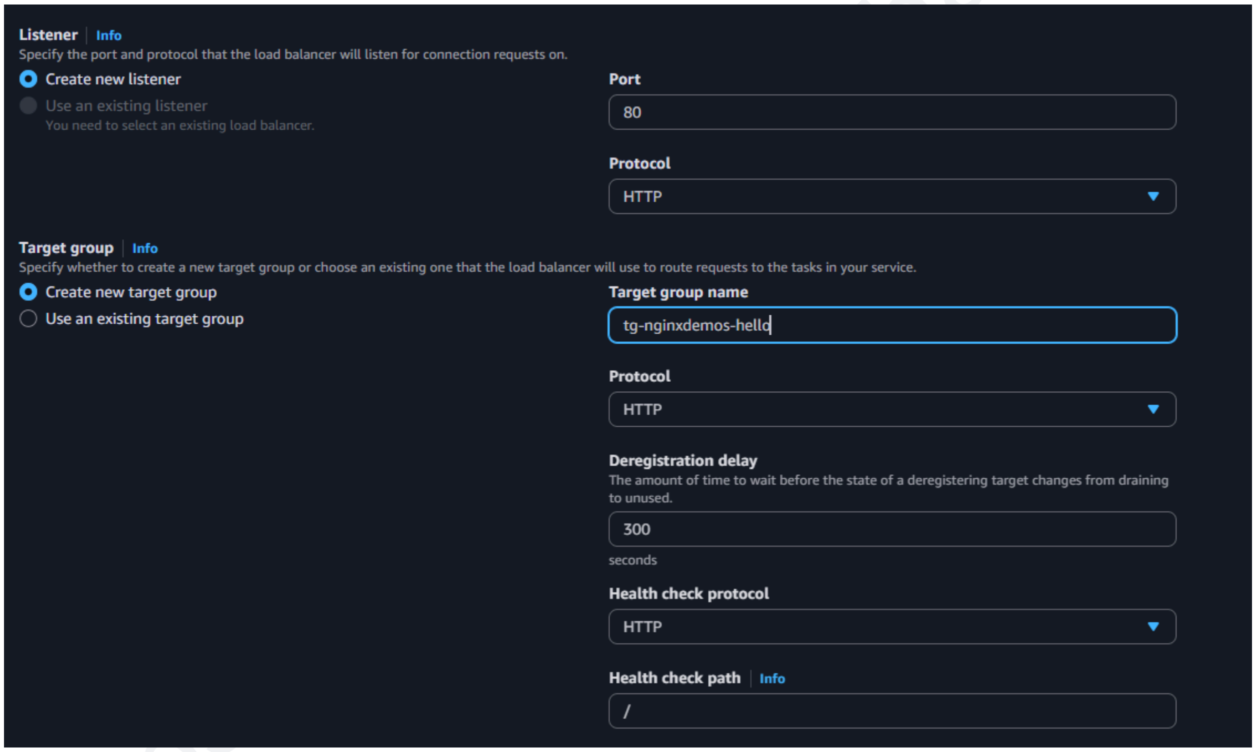
Container
The container and port to load balance the incoming traffic to
nginxdemos-Hello 80:80
Host port:Container port

Application Load Balancer
Specify whether to create a new load balancer or choose an existing one.
[✓ Create a new load balancer] Use an existing load balancer

Load balancer name
Assign a unique name for the load balancer.
DemoALBforECS

6️⃣ Listener & Target Group

- **Create new listener**:
    - Port: 80
    - Protocol: HTTP
- **Create new target group**:
    - Name: tg-nginxdemos-hello
    - Protocol: HTTP
    - Deregistration delay: 300
    - Health check protocol: HTTP
    - Health check path: /



Listener | Info
Specify the port and protocol that the load balancer will listen for connection requests on.
[✓ Create new listener]
Use an existing listener
You need to select an existing load balancer.

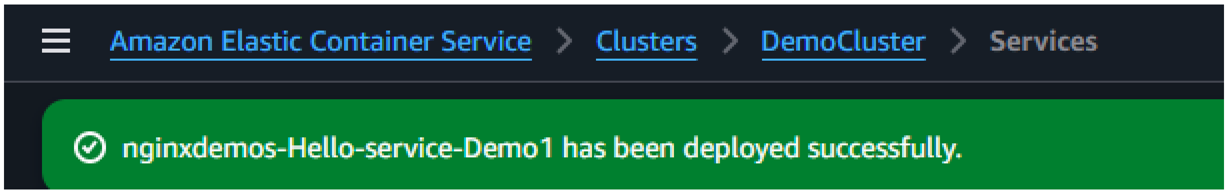
Port: 80
Protocol: HTTP

Target group | Info
Specify whether to create a new target group or choose an existing one that the load balancer will use to route requests to the tasks in your service.
[✓ Create new target group]
Use an existing target group

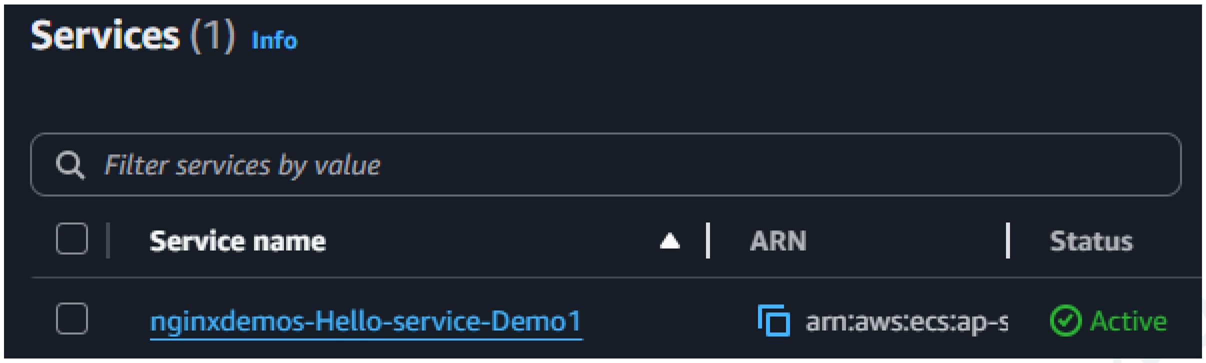
Target group name: tg-nginxdemos-hello
Protocol: HTTP
Deregistration delay
The amount of time to wait before the state of a deregistering target changes from draining to unused.
300 seconds
Health check protocol: HTTP
Health check path | Info: /

Keep Rest Of The Configuration Default For The Same.

✅ Final Step: Click **Create Service**


Amazon Elastic Container Service > Clusters > DemoCluster > Services

nginxdemos-Hello-service-Demo1 has been deployed successfully.

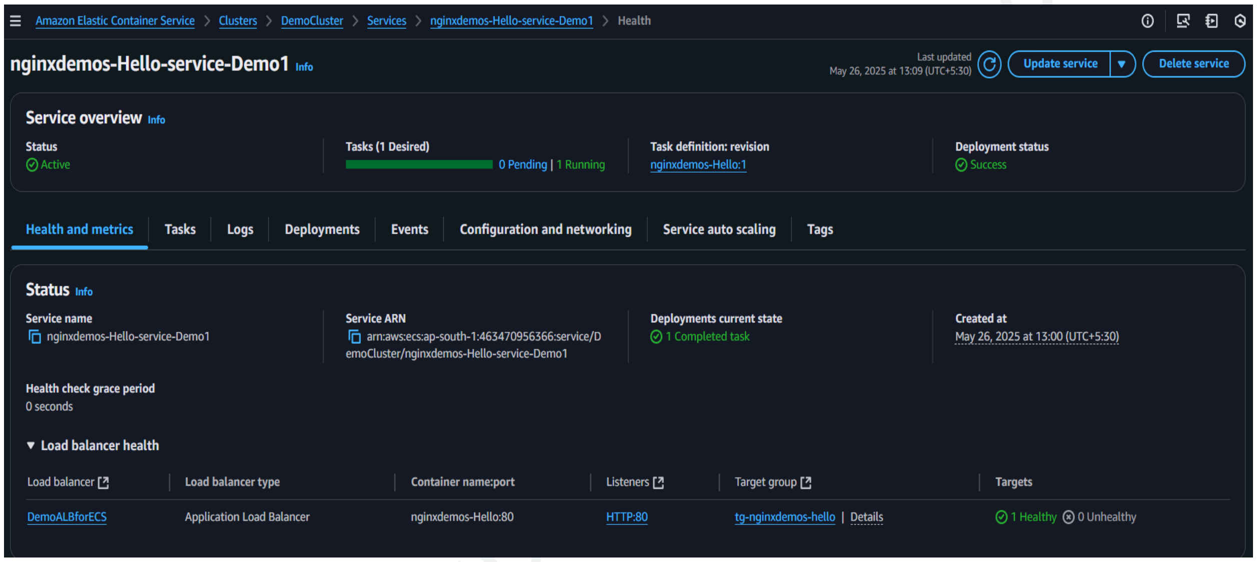


Services (1) Info

Filter services by value

Service name	ARN	Status
nginxdemos-Hello-service-Demo1	arn:aws:ecs:ap-s	Active

Click On The Service



Amazon Elastic Container Service > Clusters > DemoCluster > Services > nginxdemos-Hello-service-Demo1 > Health

nginxdemos-Hello-service-Demo1 Info

Last updated May 26, 2025 at 13:09 (UTC+5:30)

Service overview Info

Status	Tasks (1 Desired)	Task definition: revision	Deployment status
Active	0 Pending | 1 Running	nginxdemos-Hello:1	Success

Status Info

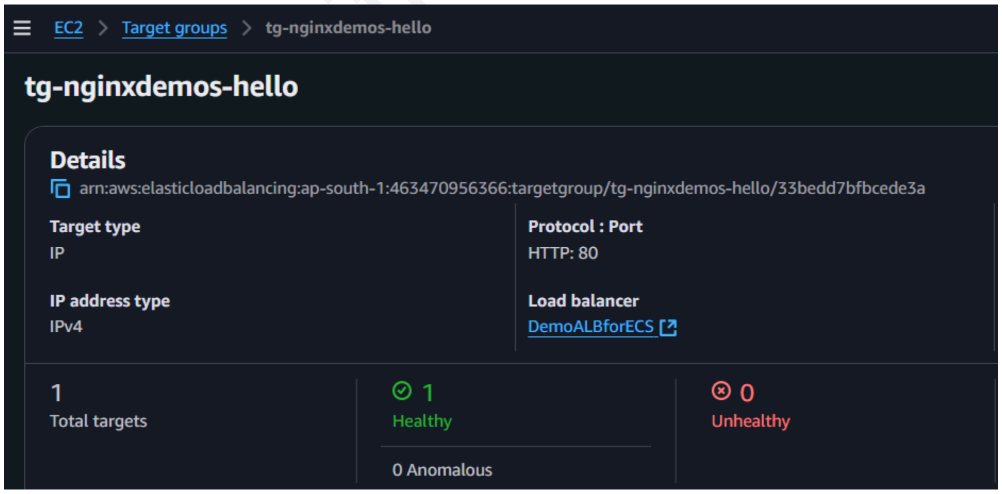
Service name	Service ARN	Deployments current state	Created at
nginxdemos-Hello-service-Demo1	arn:aws:ecs:ap-south-1:463470956366:service/DemoCluster/nginxdemos-Hello-service-Demo1	1 Completed task	May 26, 2025 at 13:00 (UTC+5:30)

Health check grace period
0 seconds

▼ Load balancer health

Load balancer	Load balancer type	Container name:port	Listeners	Target group	Targets
DemoALBforECS	Application Load Balancer	nginxdemos-Hello:80	HTTP:80	tg-nginxdemos-hello | Details	1 Healthy 0 Unhealthy

You Can See The Service is Linked To Target Group.



EC2 > Target groups > tg-nginxdemos-hello

tg-nginxdemos-hello

Details
arn:aws:elasticloadbalancing:ap-south-1:463470956366:targetgroup/tg-nginxdemos-hello/33bedd7bfbcede3a

Target type
IP

Protocol : Port
HTTP: 80

IP address type
IPv4

Load balancer
DemoALBforECS

1 Total targets

1 Healthy

0 Anomalous

0 Unhealthy

You can see the Target Group is linked to Application Load Balancer.

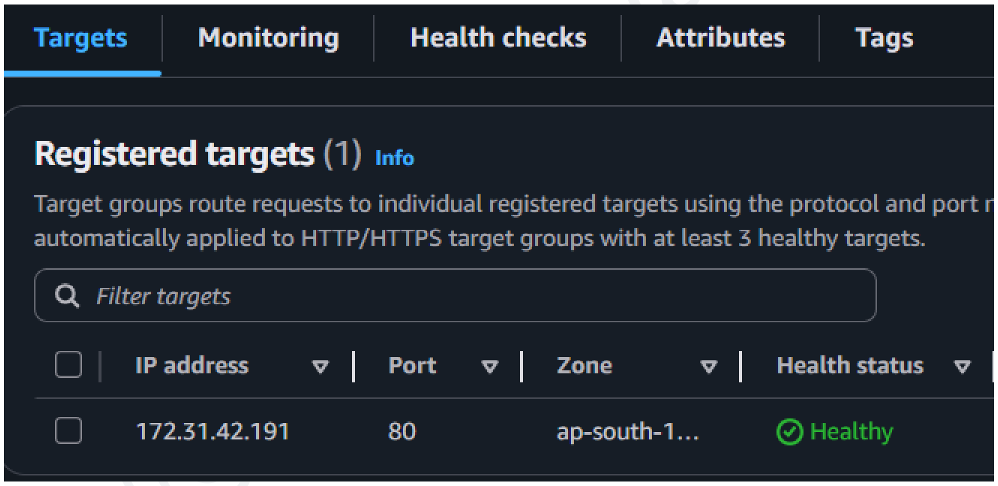
✅ Verifying Load Balancer and Target Group for nginx Container

Once your ECS container (running nginx) is up and running, follow the steps below to verify everything is working perfectly! 🚀

🔹 Step 1: Check Registered Targets in the Target Group

1. Go to the **EC2 Console**.
2. In the left menu, click on **Target Groups** under *Load Balancing*.
3. Select your target group (used by the ALB).
4. Click on the **Targets** tab.
5. You will see a registered **IP address** (e.g., `172.31.42.191`).

👉 This IP address is the **private IP of your container** running nginx.
✅ Make sure the **Health status** shows **"Healthy"** 💚 — this means the Load Balancer can reach your container.



Registered targets (1) Info

Target groups route requests to individual registered targets using the protocol and port r
automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

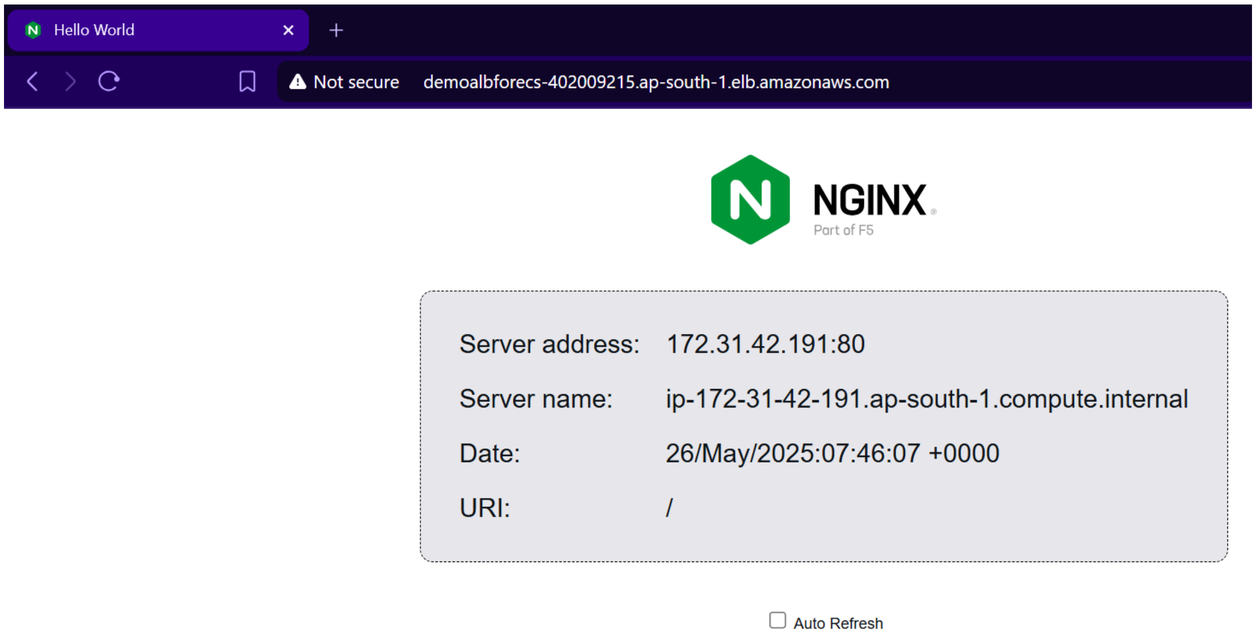
Filter targets

IP address	Port	Zone	Health status
172.31.42.191	80	ap-south-1...	Healthy

🔹 Step 2: Test the Application Using the Load Balancer

1. Go to the **EC2 Console** → **Load Balancers**.
2. Select your ALB (Application Load Balancer).
3. Copy the **DNS name** of the ALB (e.g., `demoalbforecs-xxxxxxxx.ap-south-1.elb.amazonaws.com`).
4. Paste it in your **web browser** and hit Enter. 🌐

5. You should see the **nginx welcome page** with some server details like:
- Server address
- Server name
- Date and URI


Hello World
Not secure demoalbforecs-402009215.ap-south-1.elb.amazonaws.com
NGINX
Part of F5

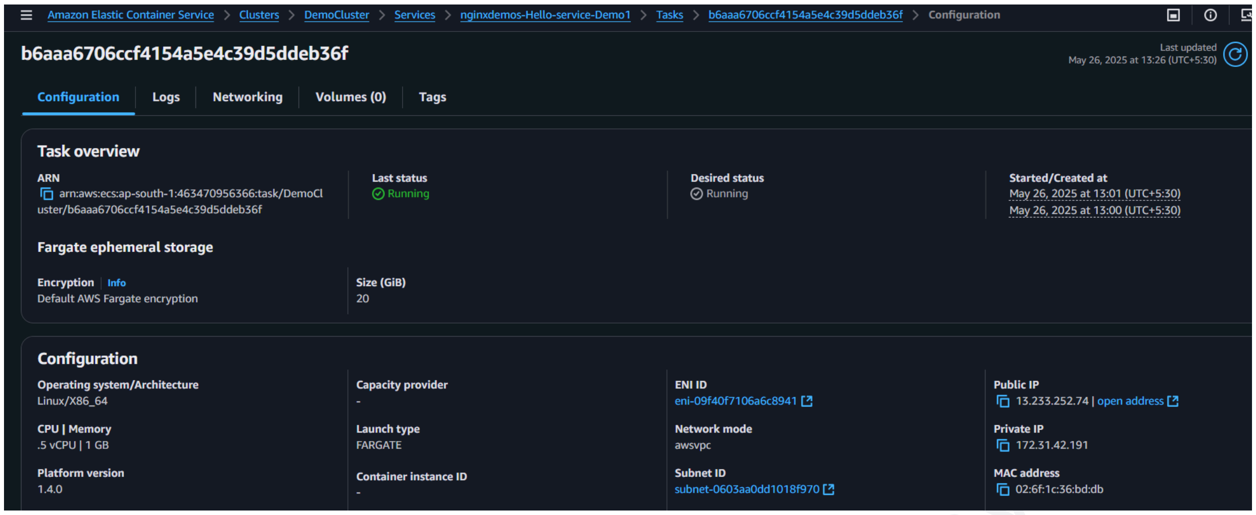
Server address: 172.31.42.191:80
Server name: ip-172-31-42-191.ap-south-1.compute.internal
Date: 26/May/2025:07:46:07 +0000
URI: /

🔍 ECS Task & Logs Verification

Check Task Details

1. Go to **ECS > Clusters > Your Cluster (e.g., `DemoCluster`) > Services > Your Service**.
2. Click on the **Tasks** tab.
3. Click on the running **Task ID**.
4. Under the **Configuration tab**, you can see:
   - Task details (e.g., Task ARN, Status, ENI, Public IP, Subnet)
   - Platform version, Memory/CPU allocation, Launch type (FARGATE)
   - ✅ Confirm task is in **Running** state.

   .


Amazon Elastic Container Service > Clusters > DemoCluster > Services > nginxdemos-Hello-service-Demo1 > Tasks > b6aaa6706ccf4154a5e4c39d5ddeb36f > Configuration

b6aaa6706ccf4154a5e4c39d5ddeb36f

Last updated May 26, 2025 at 13:26 (UTC+5:30)

Configuration | Logs | Networking | Volumes (0) | Tags

Task overview

ARN: arn:aws:ecs:ap-south-1:463470956366:task/DemoCluster/b6aaa6706ccf4154a5e4c39d5ddeb36f	Last status: Running	Desired status: Running	Started/Created at: May 26, 2025 at 13:01 (UTC+5:30) / May 26, 2025 at 13:00 (UTC+5:30)

Fargate ephemeral storage

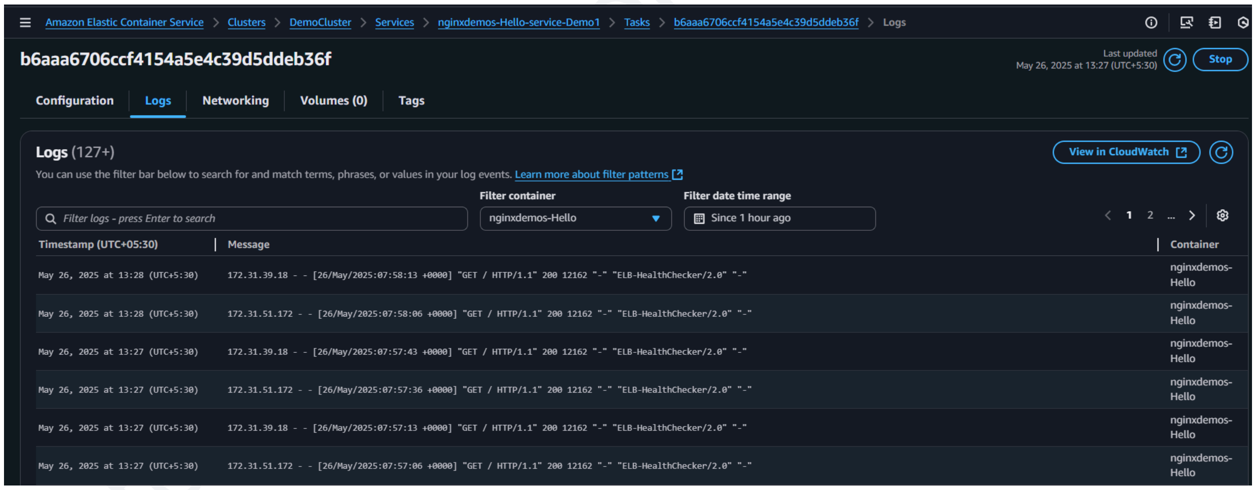
Encryption: Default AWS Fargate encryption	Size (GiB): 20

Configuration

Operating system/Architecture: Linux/X86_64	Capacity provider: -	ENI ID: eni-09f40f7106a6c8941	Public IP: 13.233.252.74 | open address
CPU | Memory: .5 vCPU | 1 GB	Launch type: FARGATE	Network mode: awsvpc	Private IP: 172.31.42.191
Platform version: 1.4.0	Container instance ID: -	Subnet ID: subnet-0603aa0dd1018f970	MAC address: 02:6f:1c:36:bd:db

📄 View Container Logs

1. Inside the task, switch to the **Logs** tab.
2. You will see **log entries** (like health check calls from ELB).
   - Helps verify container responses (e.g., HTTP 200 from Nginx).
   - 💡 Useful for debugging or checking traffic.


Amazon Elastic Container Service > Clusters > DemoCluster > Services > nginxdemos-Hello-service-Demo1 > Tasks > b6aaa6706ccf4154a5e4c39d5ddeb36f > Logs

b6aaa6706ccf4154a5e4c39d5ddeb36f

Last updated May 26, 2025 at 13:27 (UTC+5:30)

Configuration | Logs | Networking | Volumes (0) | Tags

Logs (127+)

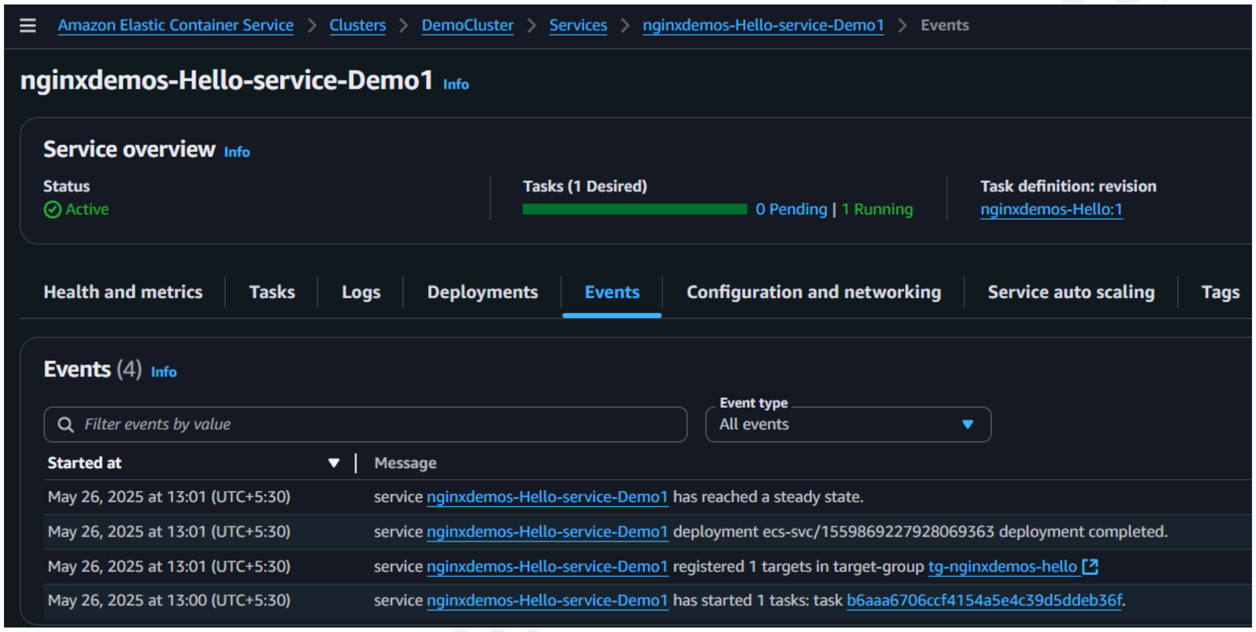
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns

Filter container: nginxdemos-Hello
Filter date time range: Since 1 hour ago

Timestamp (UTC+05:30)	Message	Container
May 26, 2025 at 13:28 (UTC+5:30)	172.31.39.18 - - [26/May/2025:07:58:13 +0000] "GET / HTTP/1.1" 200 12162 "-" "ELB-HealthChecker/2.0" "-"	nginxdemos-Hello
May 26, 2025 at 13:28 (UTC+5:30)	172.31.51.172 - - [26/May/2025:07:58:06 +0000] "GET / HTTP/1.1" 200 12162 "-" "ELB-HealthChecker/2.0" "-"	nginxdemos-Hello
May 26, 2025 at 13:27 (UTC+5:30)	172.31.39.18 - - [26/May/2025:07:57:43 +0000] "GET / HTTP/1.1" 200 12162 "-" "ELB-HealthChecker/2.0" "-"	nginxdemos-Hello
May 26, 2025 at 13:27 (UTC+5:30)	172.31.51.172 - - [26/May/2025:07:57:36 +0000] "GET / HTTP/1.1" 200 12162 "-" "ELB-HealthChecker/2.0" "-"	nginxdemos-Hello
May 26, 2025 at 13:27 (UTC+5:30)	172.31.39.18 - - [26/May/2025:07:57:13 +0000] "GET / HTTP/1.1" 200 12162 "-" "ELB-HealthChecker/2.0" "-"	nginxdemos-Hello
May 26, 2025 at 13:27 (UTC+5:30)	172.31.51.172 - - [26/May/2025:07:57:06 +0000] "GET / HTTP/1.1" 200 12162 "-" "ELB-HealthChecker/2.0" "-"	nginxdemos-Hello

🛠️ Check ECS Service Events

1. Go to the **Service** again in your ECS Cluster.
2. Click the **Events** tab.
3. You'll see recent activity such as:
   - Task launched and registered to target group
   - Deployment completed
   - Service reached a steady state.



Started at	Message
May 26, 2025 at 13:01 (UTC+5:30)	service nginxdemos-Hello-service-Demo1 has reached a steady state.
May 26, 2025 at 13:01 (UTC+5:30)	service nginxdemos-Hello-service-Demo1 deployment ecs-svc/1559869227928069363 deployment completed.
May 26, 2025 at 13:01 (UTC+5:30)	service nginxdemos-Hello-service-Demo1 registered 1 targets in target-group tg-nginxdemos-hello
May 26, 2025 at 13:00 (UTC+5:30)	service nginxdemos-Hello-service-Demo1 has started 1 tasks: task b6aaa6706ccf4154a5e4c39d5ddeb36f.

🔄 Scaling Tasks in ECS (Fargate)

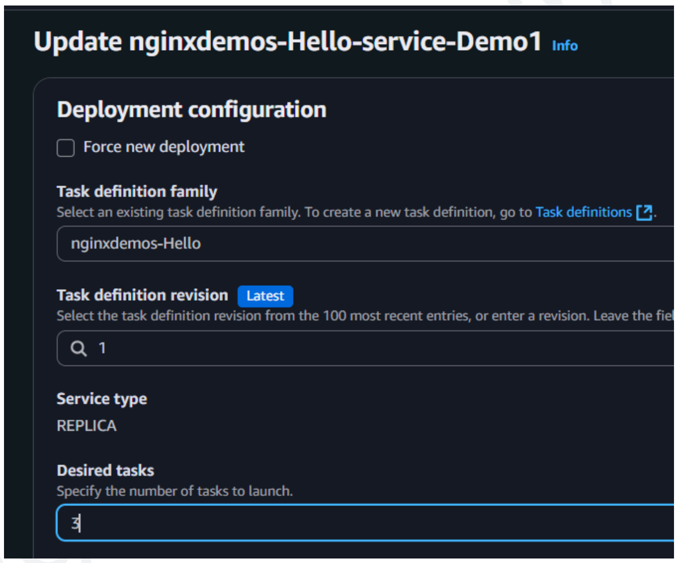
🚀 Current State

You currently have **1 running task** in your ECS Service

🔧 Step-by-Step to Scale Tasks

1. Go to **ECS > Clusters > Your Cluster > Services**.
2. Select your service (e.g., `nginxdemos-Hello-service-Demo1`).
3. Click on **Update**.
4. In the **Desired Tasks** field, change the value from 1 ➡️ 3.
5. Click **Update** to save.

✅ This tells ECS to **run 3 copies** (replicas) of your task.



Update nginxdemos-Hello-service-Demo1 Info

Deployment configuration

☐ Force new deployment

Task definition family
Select an existing task definition family. To create a new task definition, go to Task definitions.
nginxdemos-Hello

Task definition revision Latest
Select the task definition revision from the 100 most recent entries, or enter a revision. Leave the fiel
1

Service type
REPLICA

Desired tasks
Specify the number of tasks to launch.
3

⚙️ What Happens Behind the Scenes?

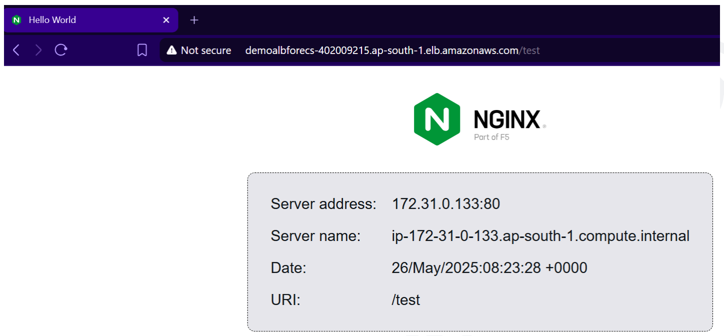
- **Fargate** provisions **2 more containers** (total: 3) to meet the new desired count.
- Each container runs the same task definition (image: `nginxdemos/hello`) but on **separate ENIs** (Elastic Network Interfaces) inside your subnet.
- These are **managed by ECS and Fargate**, meaning:
  - No need to manage EC2 instances
  - Fargate handles task placement, network setup, and IP assignment automatically

- All tasks get registered to the **Target Group** linked with your **ALB (Application Load Balancer)**.

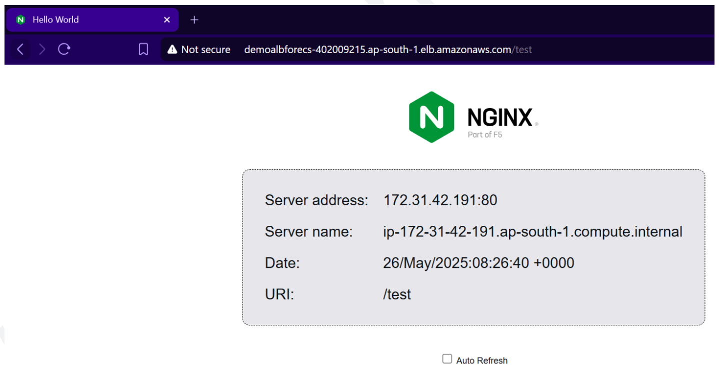
🌐 Web Page Behavior After Scaling

1. Open the **DNS of your ALB** in a browser.
2. Refresh the page several times.
3. 🎯 You'll notice:

   ○ The **IP address** shown on the Nginx demo page **keeps changing**.
   ○ Why? Because the **ALB is load balancing** the traffic across all **3 ECS containers**.
   ○ This verifies that your service is **horizontally scalable** and traffic is being handled efficiently.

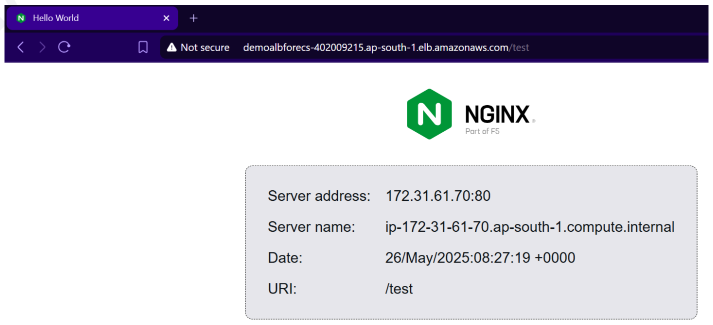

Hello World
Not secure demoalbforecs-402009215.ap-south-1.elb.amazonaws.com/test

Server address:	172.31.0.133:80
Server name:	ip-172-31-0-133.ap-south-1.compute.internal
Date:	26/May/2025:08:23:28 +0000
URI:	/test


Hello World
Not secure demoalbforecs-402009215.ap-south-1.elb.amazonaws.com/test

Server address:	172.31.42.191:80
Server name:	ip-172-31-42-191.ap-south-1.compute.internal
Date:	26/May/2025:08:26:40 +0000
URI:	/test

Auto Refresh


Hello World
Not secure demoalbforecs-402009215.ap-south-1.elb.amazonaws.com/test

Server address:	172.31.61.70:80
Server name:	ip-172-31-61-70.ap-south-1.compute.internal
Date:	26/May/2025:08:27:19 +0000
URI:	/test