



10010101010100

AWS Fargate



CCBDA Q1 - MEI UPC

Pol Plana - Zixin Zhang - Zhehan Xiang - Zhipeng Lin

1
0
1
0
0
1
0
1
0
1
0
0
1
0
1
0



Table of contents

1
0
1
0
1
0
1
1
0
0
0
0
1
1
1
0

01

Introducción y
Relevancia

02

Arquitectura y
Comparativa

03

“Hands-on”
Teórico: Tutorial

04

Perspectiva de
Costes y Recursos

1
1
0
0
1
0
1
0
1
0
1
0
1
1
1
0

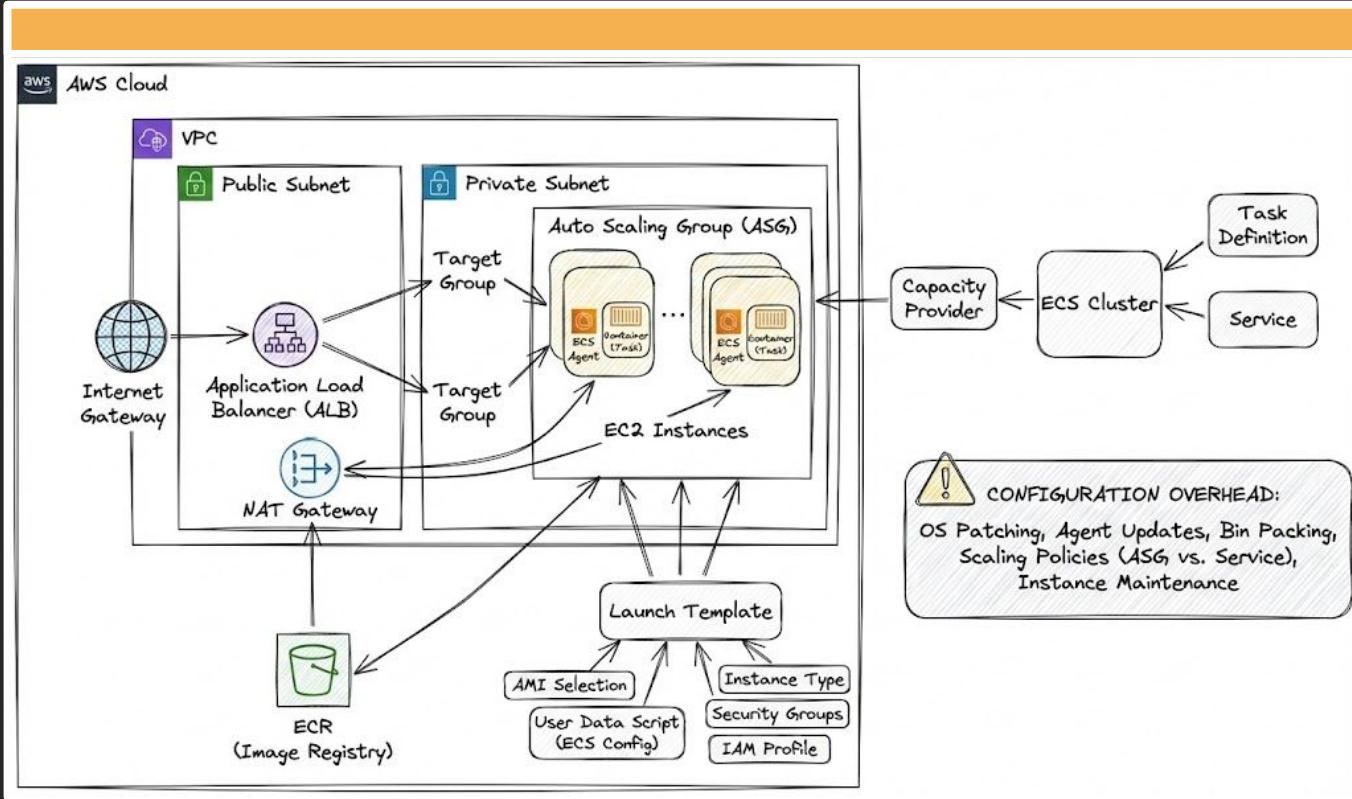


Introducción y Relevancia

SERVERLESS COMPUTE FOR CONTAINER

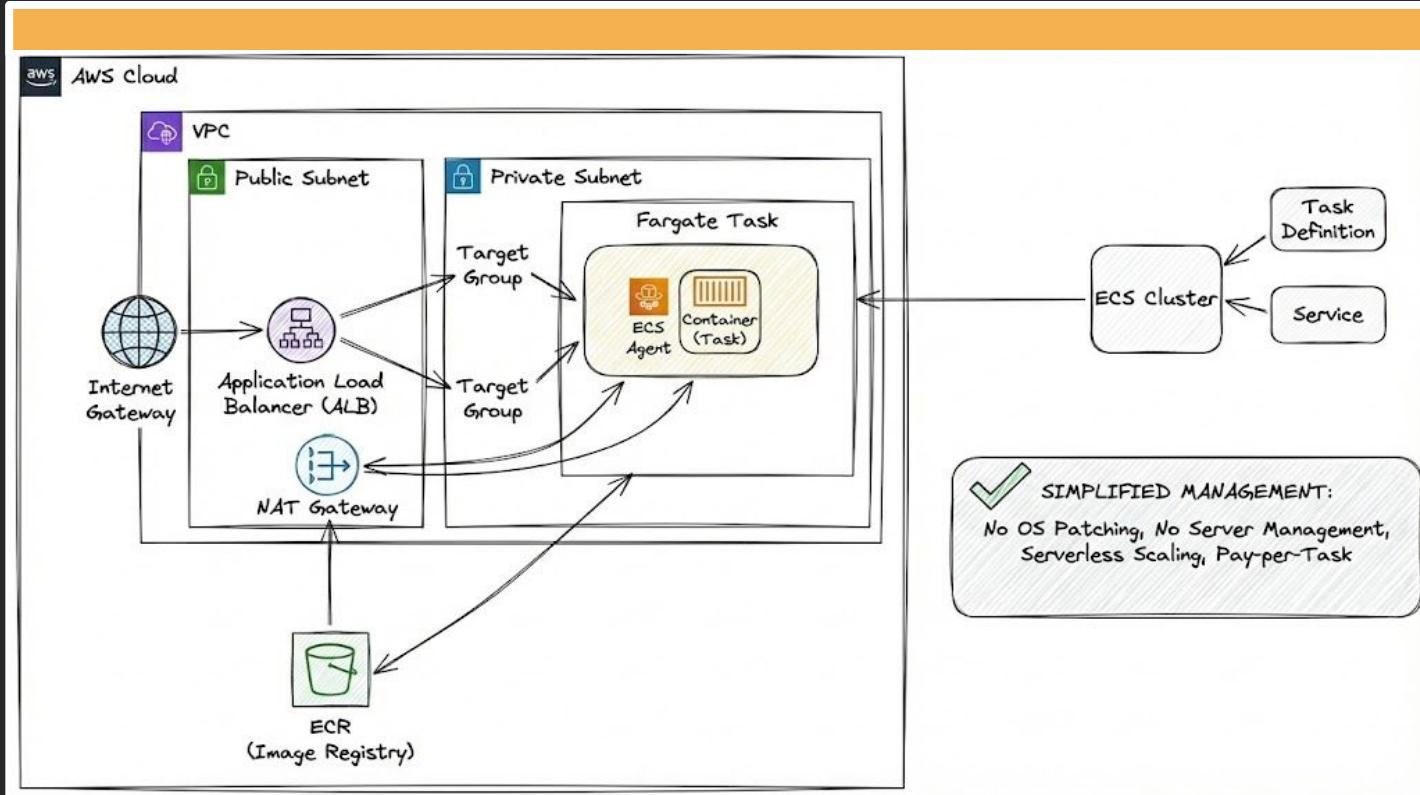
1
0
1
0
1
0
1
0
1
1
0
0
0
0
1
1
1
1
0

1
1
0
0
1
0
1
0
1
0
1
0
1
0
1
1
1
0



1
0
1
0
1
1
1
0
0
0
1
1
1
0

1
1
0
0
1
0
1
0
1
0
1
0
1
1
1
0



1
0
1
0
1
1
1
0
0
0
1
1
1
0

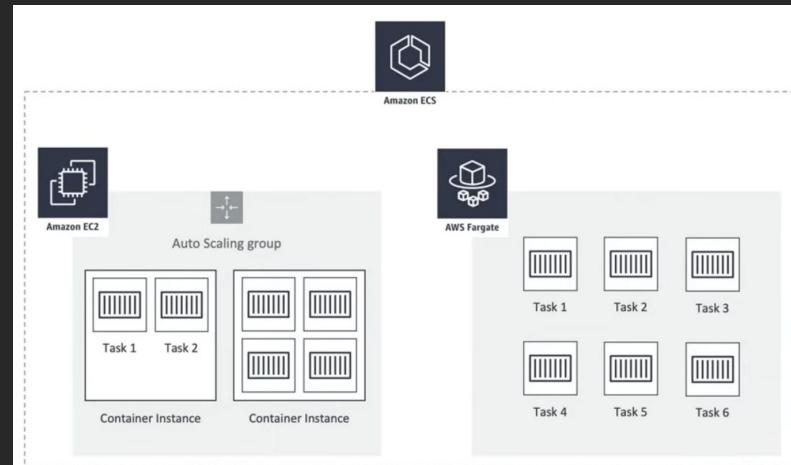
1
1
0
0
1
0
1
0
1
0
1
0
1
1
1
0



EC2 vs. Fargate

0
0
0
1
0
0
Modelo EC2: Eres dueño de la VM,
pagas por capacidad reservada
(aunque no la uses), tienes control total
pero responsabilidad total.

0
1
0
0
1
0
Modelo Fargate: AWS gestiona toda la
infraestructura. Pagas por vCPU/RAM
consumida, mayor aislamiento y
seguridad.

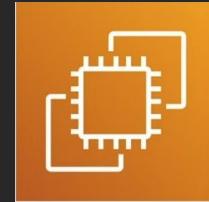


¿Cuándo usar cuál?

Fargate: Cargas de trabajo variables, entornos de desarrollo, o cuando no quieres administrar SO.



EC2: Cargas muy constantes (coste), control profundo del hardware (GPUs específicas), o Reserved Instances.

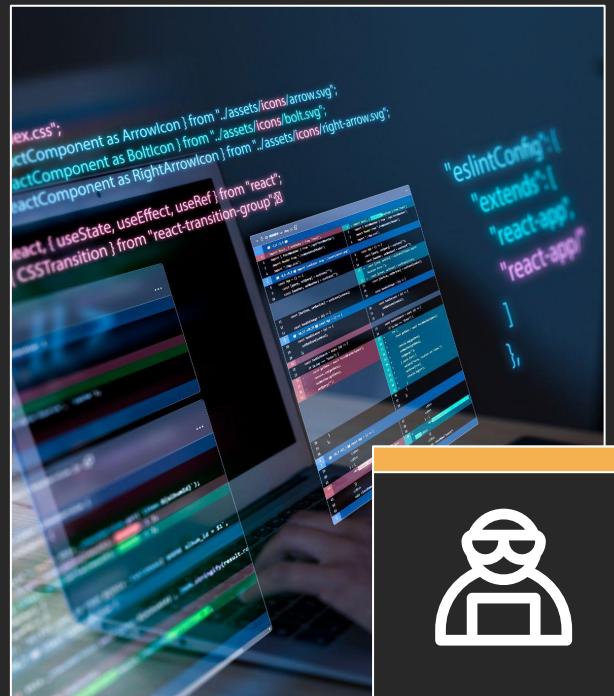




“Hands-on” Teórico: Tutorial

ECS Key Terms

- **Task** - The lowest level building block of ECS - Runtimes instances
 - **Task Definitions** - Templates for your **Tasks**. This is where you specify your Docker image, Memory/CPU Requirements, etc.
 - **Container (EC2 Only)** - Virtualized Instance that run your **Tasks**.
 - **Cluster** - EC2 - A group of Containers which run **Tasks**
Fargate - A group of **Tasks**
 - **Service** - A Task management system that ensures X amount of tasks are up and running.

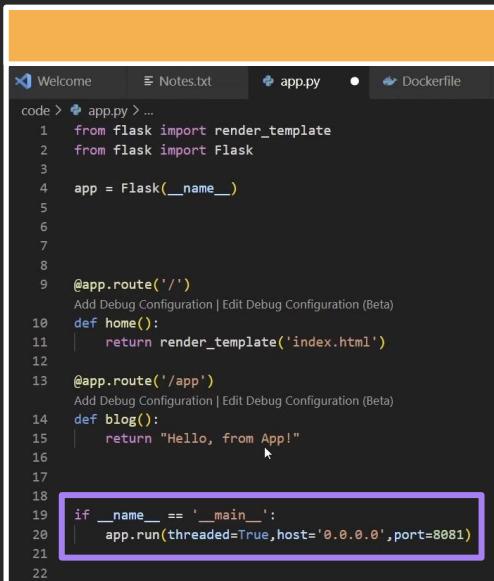


1
0
1
0
0
1
0
1
0
1
0
0
1
0
1
0
1
0

“Hands-on” Teórico: Tutorial

Preparamos nuestro proyecto y lo Dockerizamos

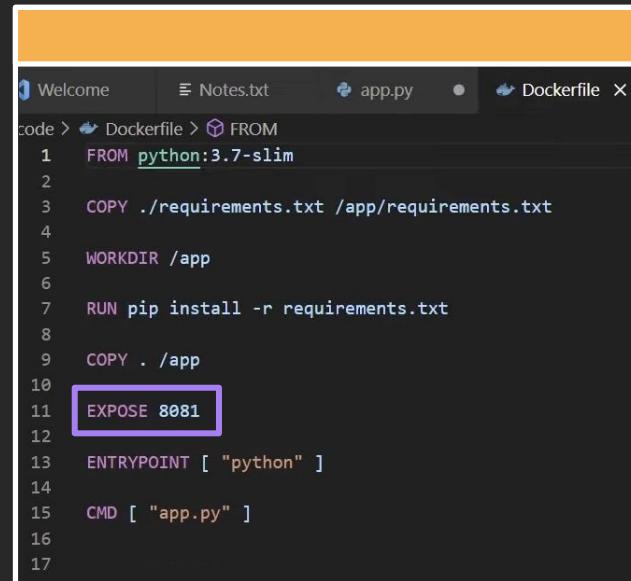
1
0
0
1
0
1
0
1
0
1
0
1
0
0
1
0
1
0
0
1
0
0



```
1 Welcome      Notes.txt    app.py    Dockerfile
2 code > app.py > ...
3
4     from flask import render_template
5     from flask import Flask
6
7     app = Flask(__name__)
8
9     @app.route('/')
10    def home():
11        return render_template('index.html')
12
13    @app.route('/app')
14    def blog():
15        return "Hello, from App!"
16
17
18    if __name__ == '__main__':
19        app.run(threaded=True,host='0.0.0.0',port=8081)
20
21
22
```

app.py code
Dockerfile code

1
0
1
0
0
1
0
1
0
1
0
0
1
0
0
1
0
1
0
0
1
0
0



```
1 Welcome      Notes.txt    app.py    Dockerfile X
2 code > Dockerfile > FROM
3     FROM python:3.7-slim
4
5     COPY ./requirements.txt /app/requirements.txt
6
7     WORKDIR /app
8
9     RUN pip install -r requirements.txt
10
11    EXPOSE 8081
12
13    ENTRYPOINT [ "python" ]
14
15    CMD [ "app.py" ]
16
17
```



“Hands-on” Teórico: Tutorial

Tenemos nuestra imagen de Docker almacenada en ECR

The screenshot shows the AWS Management Console interface for the Amazon Container Services (Amazon ECR) service. The left sidebar navigation bar includes links for Amazon ECS, Clusters, Task definitions, Amazon EKS, Clusters, Amazon ECR, Repositories, Images, Permissions, Lifecycle Policy, Tags, Registries, and Public gallery. The main content area displays the 'Image details' for a specific repository and image. The 'Image URI' is highlighted with a purple box and contains the value: `755314965794.dkr.ecr.us-east-1.amazonaws.com/flask-app`. Other details shown include the Digest (`sha256:b8639ed6d3f66cb44315d68c4b23c07f341b922c6fa60ad8f68848dbb2e4433a`), Image tags (latest), Repository (flask-app), Pushed at (May 08, 2021 04:30:43 PM (UTC-04)), Size (64.18 MB), Type (Artifact media type: application/vnd.docker.container.image.v1+json, Image manifest type: application/vnd.docker.distribution.manifest.v2+json), and a Scanning section.



“Hands-on” Teórico: Tutorial

The screenshot shows the AWS ECS Create Cluster wizard. In Step 1: Select cluster template, there are three options:

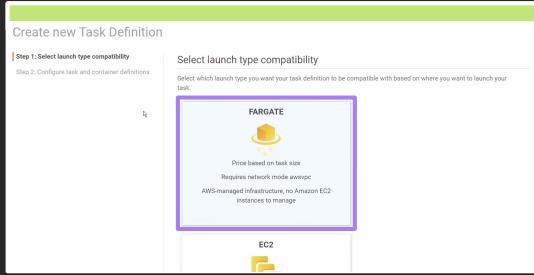
- Networking only**: Contains "Cluster" and "VPC (optional)". It is highlighted with a purple box.
- EC2 Linux + Networking**: Contains "Cluster", "VPC", and "Subnets". It also includes "Auto Scaling group with Linux AMI".
- EC2 Windows + Networking**: Contains "Cluster" and "VPC".

Below the templates is a "Powered by AWS Fargate" section. To the right, a green bar labeled "Configure cluster" leads to the next step. In the "Networking" section, a cluster name "DemoAppCluster" is entered, and there are checkboxes for "Create VPC" and "Create a new VPC for this cluster".

Cambiamos de servicio, ahora ECS (ejecutar, administrar y escalar contenedores Docker), para crear un Cluster.

“Hands-on” Teórico: Tutorial

Ahora definimos una tarea con nuestro proyecto Dockerizado y almacenado en ECR



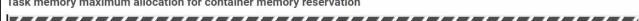
1
0
0
1
0
1
0
1
0
1
0
1
0
1
0
0

Configure task and container definitions

Task Definition Name* DemoTaskDefinition i

Task memory (GB) 2GB ▼
The valid memory range for 1 vCPU is: 2GB - 8GB.

Task CPU (vCPU) 1 vCPU ▼
The valid CPU range for 2GB memory is: 0.25 vCPU - 1 vCPU.

Task memory maximum allocation for container memory reservation

0 2048 shared of 2048 MiB

Task CPU maximum allocation for containers

0 1024 shared of 1024 CPU units

Container Definitions

Add container ▼
Standard

Container name* DemoAppContainer

Image* 755314965794.dkr.ecr.us-east-1.amazonaws.com/flask-app:latest i

Create Task Definition: DemoTaskDefinition

DemoTaskDefinition succeeded

Create CloudWatch Log Group

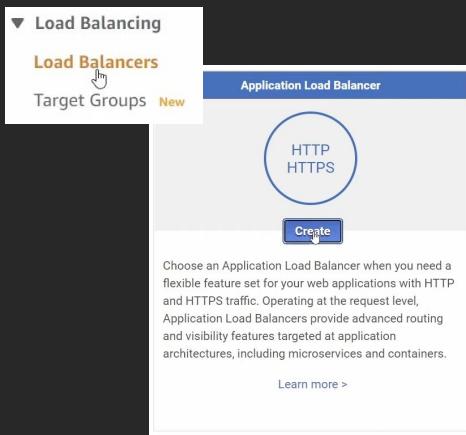
CloudWatch Log Group created CloudWatch Log Group /ecs/DemoTaskDefinition i

Memory Limits (MiB) Soft limit 2048 ▼

Port mappings Container port Protocol
8081 tcp ▼

“Hands-on” Teórico: Tutorial

A continuación creamos un balanceador de carga a través del servicio EC2



Step 1: Configure Load Balancer

Basic Configuration

To configure your load balancer, provide a name, select a scheme, specify one or more listeners, and select a network with a listener that receives HTTP traffic on port 80.

Name: DemoAppLB

Scheme: internet-facing internal

IP address type: ipv4

Load Balancer Protocol: HTTP

Load Balancer Port: 80

VPC: vpc-29a85454 (172.31.0.0/16) (default)

Availability Zones:

- us-east-1a subnet-da99e997
- us-east-1b subnet-ed7fd2b2
- us-east-1c subnet-d3cc6cb5
- us-east-1d subnet-5e2e817f
- us-east-1e subnet-5851a569

IPv4 address: Assigned by AWS



“Hands-on” Teórico: Tutorial

No necesitamos hacer nada en la step 2, ya que hemos usado los valores por defecto (HTTP). No se ha configurado HTTPS en este entorno de prueba.

1. Configure Load Balancer 2. Configure Security Settings **3. Configure Security Groups** 4. Configure Routing 5. Register Targets 6. Review

Step 3: Configure Security Groups

A security group is a set of firewall rules that control the traffic to your load balancer. On this page, you can add rules to allow specific traffic to reach your load balancer. First, decide whether to create a new security group or select an existing one.

Assign a security group	<input checked="" type="radio"/> Create a new security group <input type="radio"/> Select an existing security group			
Security group name	<input type="text" value="DemoAppLB-SecurityGroup"/>			
Description	<input type="text" value="DemoAppLB-SecurityGroup"/>			
Type <small>i</small>	Protocol <small>i</small>	Port Range <small>i</small>	Source <small>i</small>	
<input type="button" value="Custom TCP F ▾"/>	<input type="text" value="TCP"/>	<input type="text" value="80"/>	<input type="button" value="Custom ▾"/>	<input type="text" value="0.0.0.0/0, ::/0"/> 

“Hands-on” Teórico: Tutorial

No necesitamos hacer nada en la step 2, ya que hemos usado los valores por defecto (HTTP).
No se ha configurado HTTPS en este entorno de prueba.

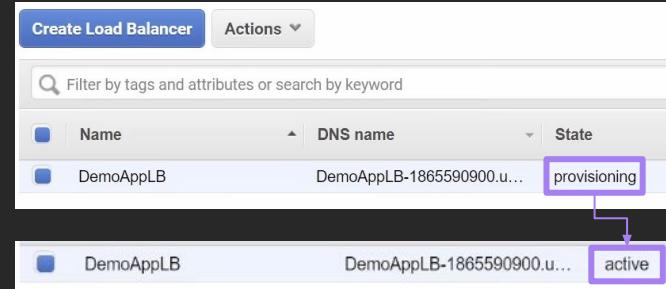
Step 4: Configure Routing

Your load balancer routes requests to the targets in this target group using the protocol and port you specify in this step will apply to all of the listeners configured on this load balancer.

Target group

Target group	i	New target group
Name	i	DemoAppTargetGroup
Target type	<input type="radio"/> Instance <input checked="" type="radio"/> IP <input type="radio"/> Lambda function	
Protocol	i	HTTP
Port	i	8081

Tampoco vamos a registrar targets (step 5), ya que aún no tenemos nuestras instancias ejecutándose en Fargate



Create Load Balancer			Actions
Filter by tags and attributes or search by keyword			
Name	DNS name	State	
DemoAppLB	DemoAppLB-1865590900.u...	provisioning	Edit
DemoAppLB	DemoAppLB-1865590900.u...	active	Edit



“Hands-on” Teórico: Tutorial

The screenshot shows the AWS ECS Cluster details page for 'DemoAppCluster'. The cluster ARN is arn:aws:ecs:us-east-1:755314965794:cluster/DemoAppCluster and the status is ACTIVE. There are 0 registered container instances, 0 pending tasks, 0 running tasks, 0 active service count, and 0 draining service count. The interface includes tabs for Services, Tasks, ECS Instances, Metrics, Scheduled Tasks, Tags, and Capacity Providers. A prominent 'Create' button is visible. A message box on the right states: 'Ya podemos crear nuestro servicio en el clúster creado previamente en ECS'.

Cluster : DemoAppCluster

Get a detailed view of the resources on your cluster.

Cluster ARN: arn:aws:ecs:us-east-1:755314965794:cluster/DemoAppCluster

Status: ACTIVE

Registered container instances: 0

Pending tasks count: 0 Fargate, 0 EC2

Running tasks count: 0 Fargate, 0 EC2

Active service count: 0 Fargate, 0 EC2

Draining service count: 0 Fargate, 0 EC2

Services Tasks ECS Instances Metrics Scheduled Tasks Tags Capacity Providers

Create Update Delete Actions ▾

Last updated on May 8, 2021 9:31:36 PM (0m ago)

Filter in this page Launch type: ALL Service type: ALL

No results

Ya podemos crear nuestro servicio en el clúster creado previamente en ECS



“Hands-on” Teórico: Tutorial

Step 1: Configure service

Step 2: Configure network

Step 3: Set Auto Scaling (optional)

Step 4: Review

Launch type FARGATE EC2 (i)

[Switch to capacity provider strategy](#) (i)

Task Definition Family ▼

Revision ▼

Platform version ▼ (i)

Cluster ▼ (i)

Service name (i)

Service type* REPLICA (i)

Number of tasks ◀ ▶

Number of tasks
represents the
minimum tasks that
will be running. A
maximum can be set
through autoscaling.

“Hands-on” Teórico: Tutorial

Step 2: Configure network

Configure network

VPC and security groups

VPC and security groups are configurable when your task definition uses the awsvpc network mode.

Cluster VPC*	vpc-29a85454 (172.31.0.0/16)	▼	i
Subnets*	<div style="border: 1px solid #ccc; padding: 5px;"> subnet-da99e997 (172.31.16.0/20) - us-east-1a assign ipv6 on creation: Disabled </div> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;"> subnet-ed7fd2b2 (172.31.32.0/20) - us-east-1b assign ipv6 on creation: Disabled </div>		
Security groups*	DemoAp-6734	Edit	i
Auto-assign public IP	ENABLED	▼	i

Load balancer type*	<input type="radio"/> None Your service will not use a load balancer.
	<input checked="" type="radio"/> Application Load Balancer Allows containers to use dynamic host port mapping (multiple tasks allowed per container instance). Multiple services can use the same listener port on a single load balancer with rule-based routing and paths.
	<input type="radio"/> Network Load Balancer A Network Load Balancer functions at the fourth layer of the Open Systems Interconnection (OSI) model. After the load balancer receives a request, it selects a target from the target group for the default rule using a flow hash routing algorithm.
	<input type="radio"/> Classic Load Balancer Requires static host port mappings (only one task allowed per container instance); rule-based routing and paths are not supported.
Service IAM role	Task definitions that use the awsvpc network mode use the AWSServiceRoleForECS service-linked role, which is created for you automatically. Learn more.
Load balancer name	<input type="text" value="DemoAppLB"/> 

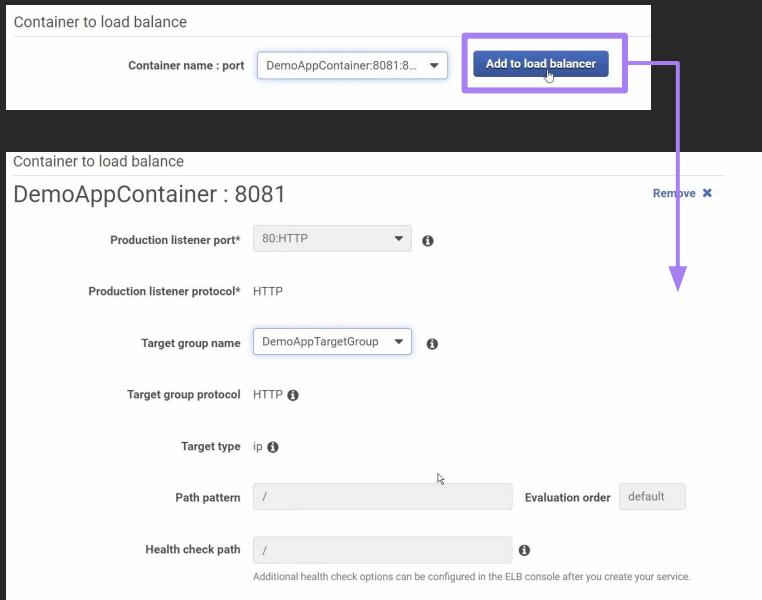
Container to load balance

Container name : port

DemoAppContainer:8081:8...

[Add to load balancer](#)

“Hands-on” Teórico: Tutorial



Si quisieramos que el número de tareas definido anteriormente se ajuste a la demanda, podríamos configurarlo.

Step 3: Set Auto Scaling (optional)

Set Auto Scaling (optional)

Automatically adjust your service's desired count up and down within a specified range in response to traffic. You can modify your Service Auto Scaling configuration at any time to meet the needs of your application.

- Service Auto Scaling
- Do not adjust the service's desired count
 - Configure Service Auto Scaling to adjust your service's desired count

A continuación creamos el servicio.



“Hands-on” Teórico: Tutorial

Task status: Running Stopped						
<input type="text" value="Filter in this page"/> < 1-2 > Page size 50 ▾						
Task	Task Definition ...	Last status	Desired status	Group	Launch type	Platform version ...
6898741f90e446...	DemoTaskDefiniti...	PENDING	RUNNING	service:DemoApp...	FARGATE	1.4.0
6bfc2d7f70fd487...	DemoTaskDefiniti...	PENDING	RUNNING	service:DemoApp...	FARGATE	1.4.0

Task	Task Definition ...	Last status
6898741f90e446...	DemoTaskDefiniti...	RUNNING
6bfc2d7f70fd487...	DemoTaskDefiniti...	RUNNING

Task	Task Definition ...	Last status
65584627712247...	DemoTaskDefiniti...	PENDING
d6b0d739b4b842...	DemoTaskDefiniti...	PROVISIONING

Hemos creado el servicio con sus dos tareas. Estas pasarán de pendiente a listo, y después fallarán. Necesitamos permitir que se conecten a nuestra red, con el Load Balancer, ya que este les hará “health pings”, pero no responderán, se pensará que no existen y se recrearán de nuevo infinitamente.

“Hands-on” Teórico: Tutorial

Accedemos a los detalles de nuestro Cluster de ECS y seleccionamos el Security group configurado para ver sus detalles. Vamos a editar las inbound rules.

The screenshot shows the 'Network Access' section of the ECS Cluster Details page. It includes fields for 'Health check grace period' (0), 'Allowed VPC' (vpc-29a85454), 'Allowed subnets' (subnet-da99e997, subnet-ed7fd2b2), and 'Security groups*' (sg-0827ce7c831e00105). The 'Auto-assign public IP' field is set to 'ENABLED'. A purple box highlights the 'Security groups*' field.

The screenshot shows the 'Inbound rules' tab for the security group sg-0827ce7c831e00105. It lists one rule: Type: HTTP, Protocol: TCP, Port range: 80, Source: 0.0.0.0/0, and Description: -. An orange box highlights the 'Edit inbound rules' button.



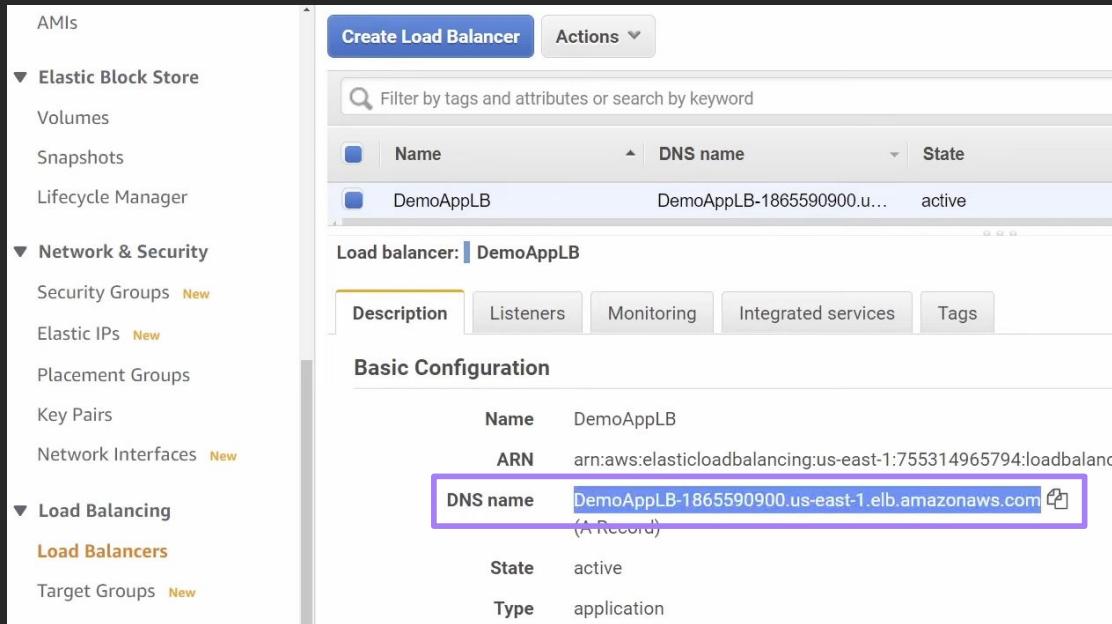
“Hands-on” Teórico: Tutorial

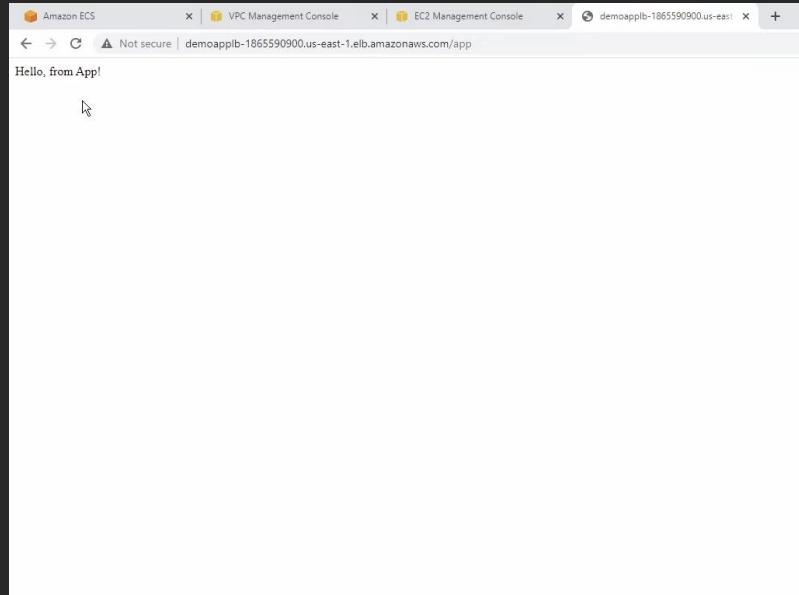
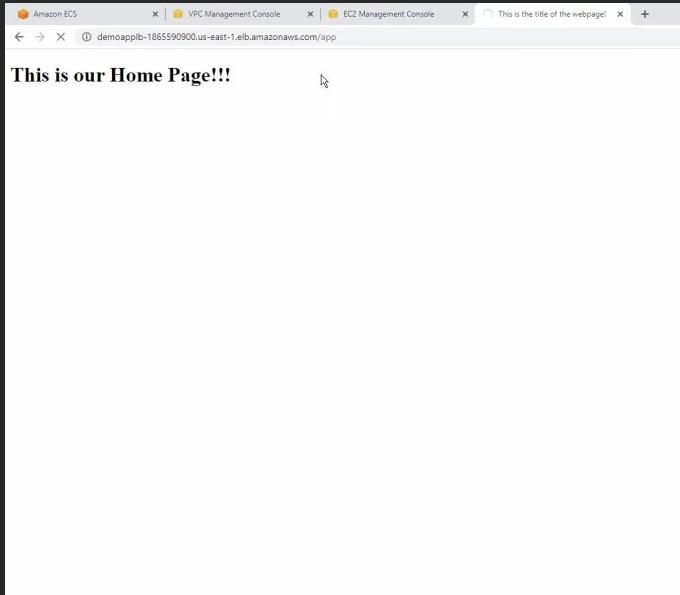
Accedemos a los detalles de nuestro Cluster de ECS y seleccionamos el Security group configurado para ver sus detalles. Vamos a editar las inbound rules.

The screenshot shows the 'Network Access' section of the ECS Cluster Details page. It includes fields for 'Health check grace period' (0), 'Allowed VPC' (vpc-29a85454), 'Allowed subnets' (subnet-da99e997, subnet-ed7fd2b2), and 'Security groups*' (sg-0827ce7c831e00105). A purple box highlights the 'Security groups*' field.

The screenshot shows the 'Inbound rules' tab for the security group sg-0827ce7c831e00105. It lists one rule: Type: HTTP, Protocol: TCP, Port range: 80, Source: 0.0.0.0/0. A purple box highlights the 'Edit inbound rules' button. Below it, a modal window shows the rule being edited with the source set to 'sg-0f7fd1b9338114e8f'. A green success message at the bottom states: 'Inbound security group rules successfully modified on security group (sg-0827ce7c831e00105 | DemoAp-6734)'.

“Hands-on” Teórico: Tutorial





10100101000101010

“Hands-on” Teórico: Tutorial



Modelos de precios

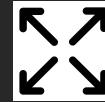
1
0
0
1
0
1
0
1
0
0
0
1
0
0
1
0
0
1
0
1
0
1
1
0
1
1
0
1
0
1
1
1
1



**Pagas por CPU y
GB de memoria
por segundo.**



**Cuidado con
dejar tareas
zombies
encendidas.**



Right-Sizing

1
0
1
0
0
1
1
0
0
1
1
1
0
1
1
1
1



Comparativa de TCO: Fargate vs EC2



Costes Operativos Ocultos

EC2:

- Parchear OS
 - Agentes de Docker
 - Escalar cluster
 - (...)

Fargate: Null

Eficiencia de uso

EC2: Espacio para picos.

Fargate: 100%



Recursos de aprendizaje

Página oficial: <https://aws.amazon.com/es/fargate/>

Workshop: <https://awsworkshop.io/tags/fargate/>

Otros: <https://spacelift.io/blog/what-is-aws-fargate#what-is-aws-fargate>

<https://www.youtube.com/watch?v=DVrGXjjkpig>

<https://www.youtube.com/watch?v=o7s-eigrMAl>



Conclusión final

1001010100010101010

1
0
1
0
0
1
0
0
1
0
1
1
0
1
1
1

line



Gracias!

Pol Plana - Zixin Zhang -
Zhehan Xiang - Zhipeng Lin

**CCBDA Q1
MEI UPC**