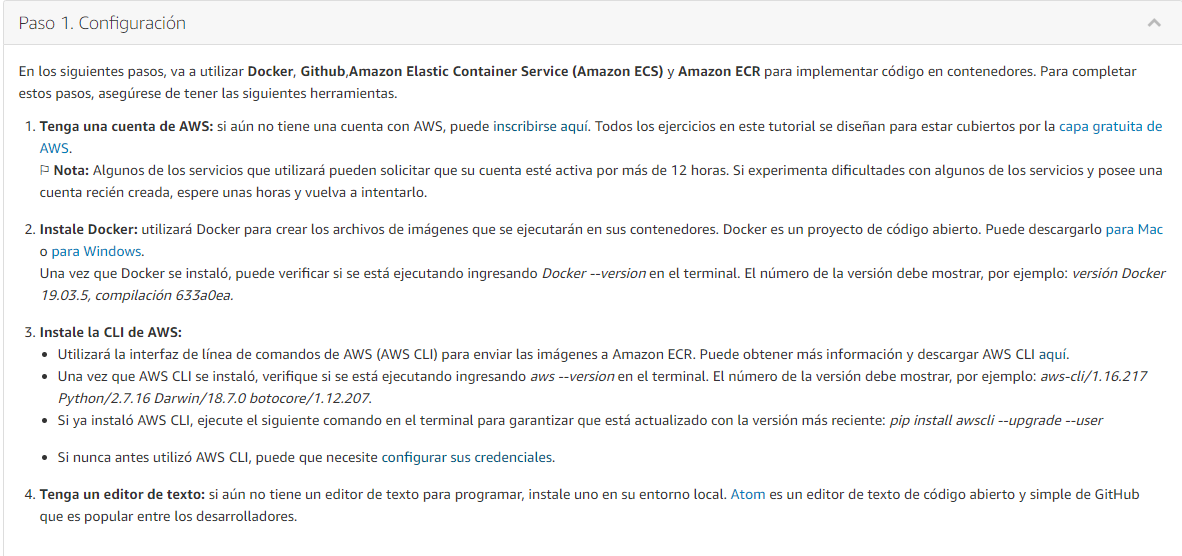
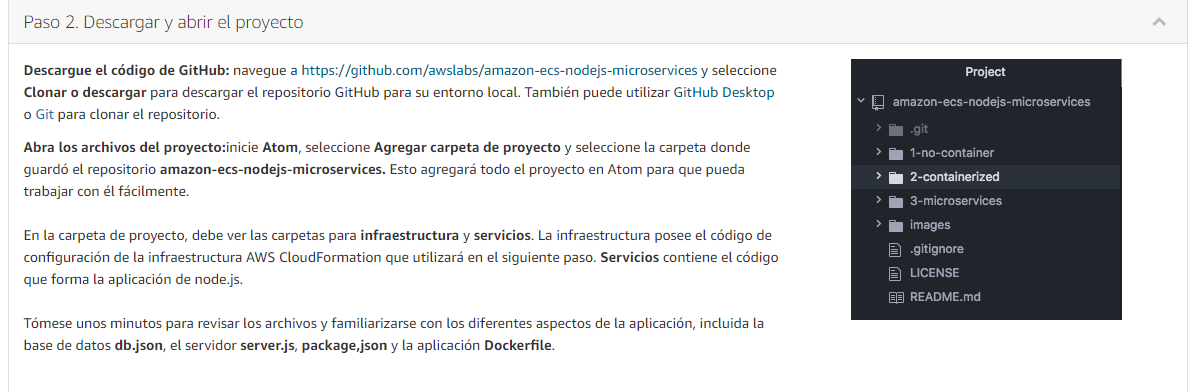
# Instructivo

# Configuraciones iniciales

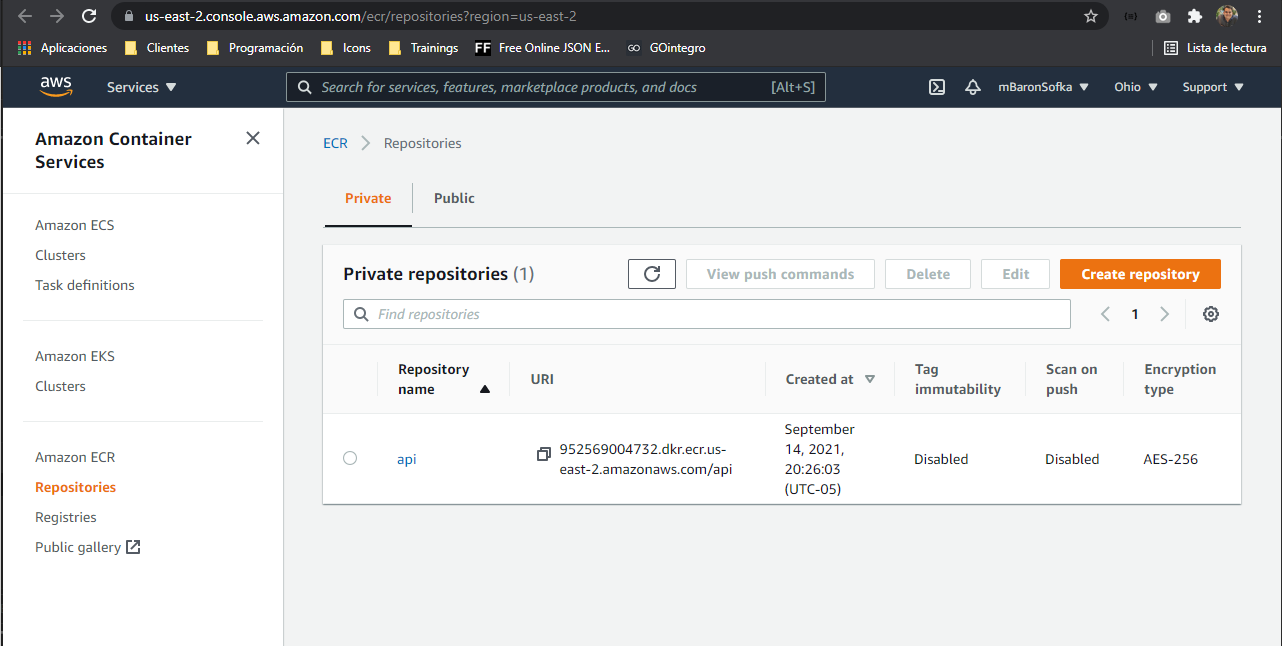
1. Prerequisitos: Configurar cuenta e instalar programas necesarios



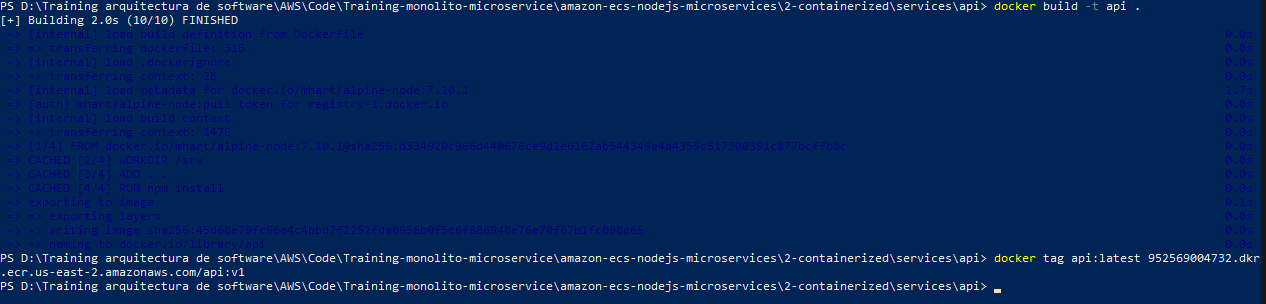
1. Descargar proyecto de prueba



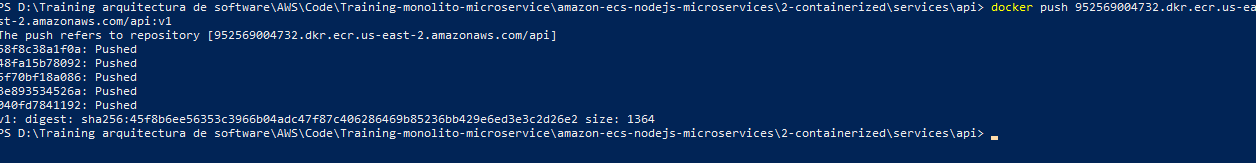
1. Aprovisionar repositorio

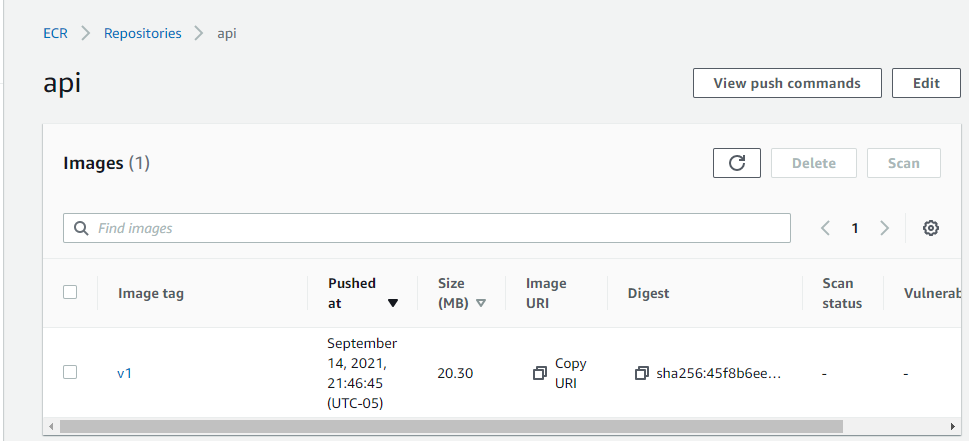


1. Crear la imagen docker



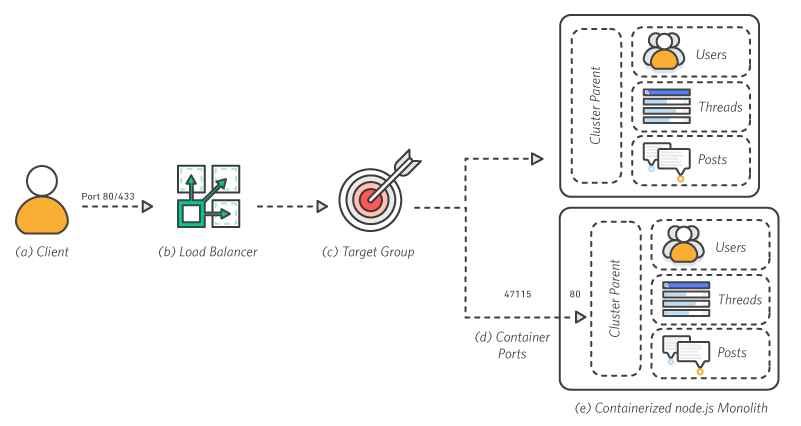
1. Hacer push



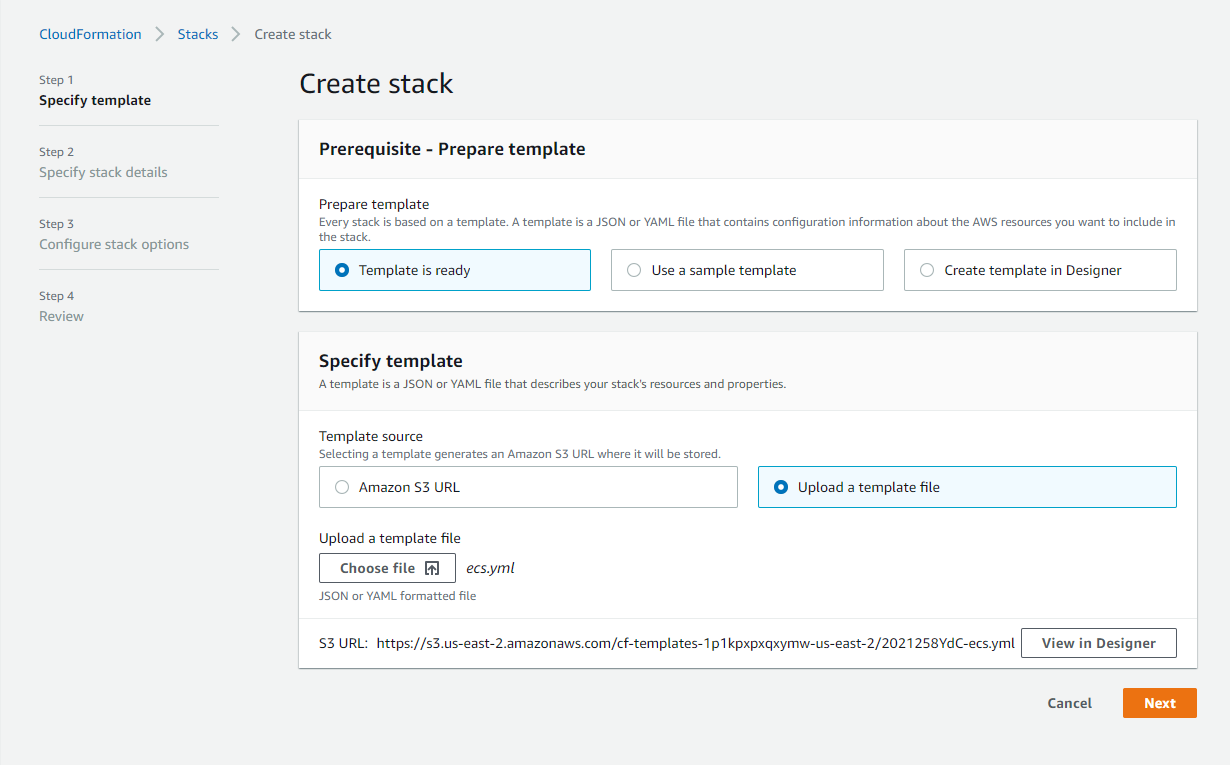


# Implementar el monolito

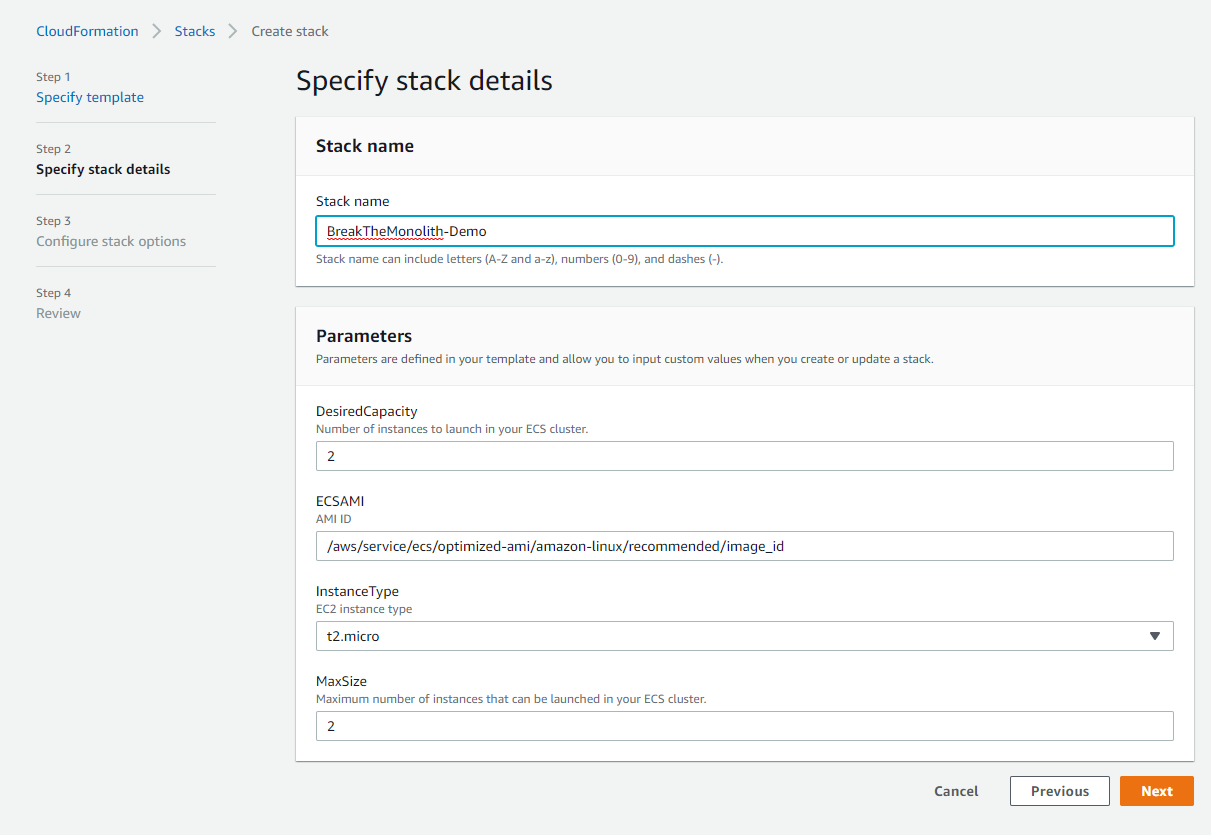
Información general de la arquitectura



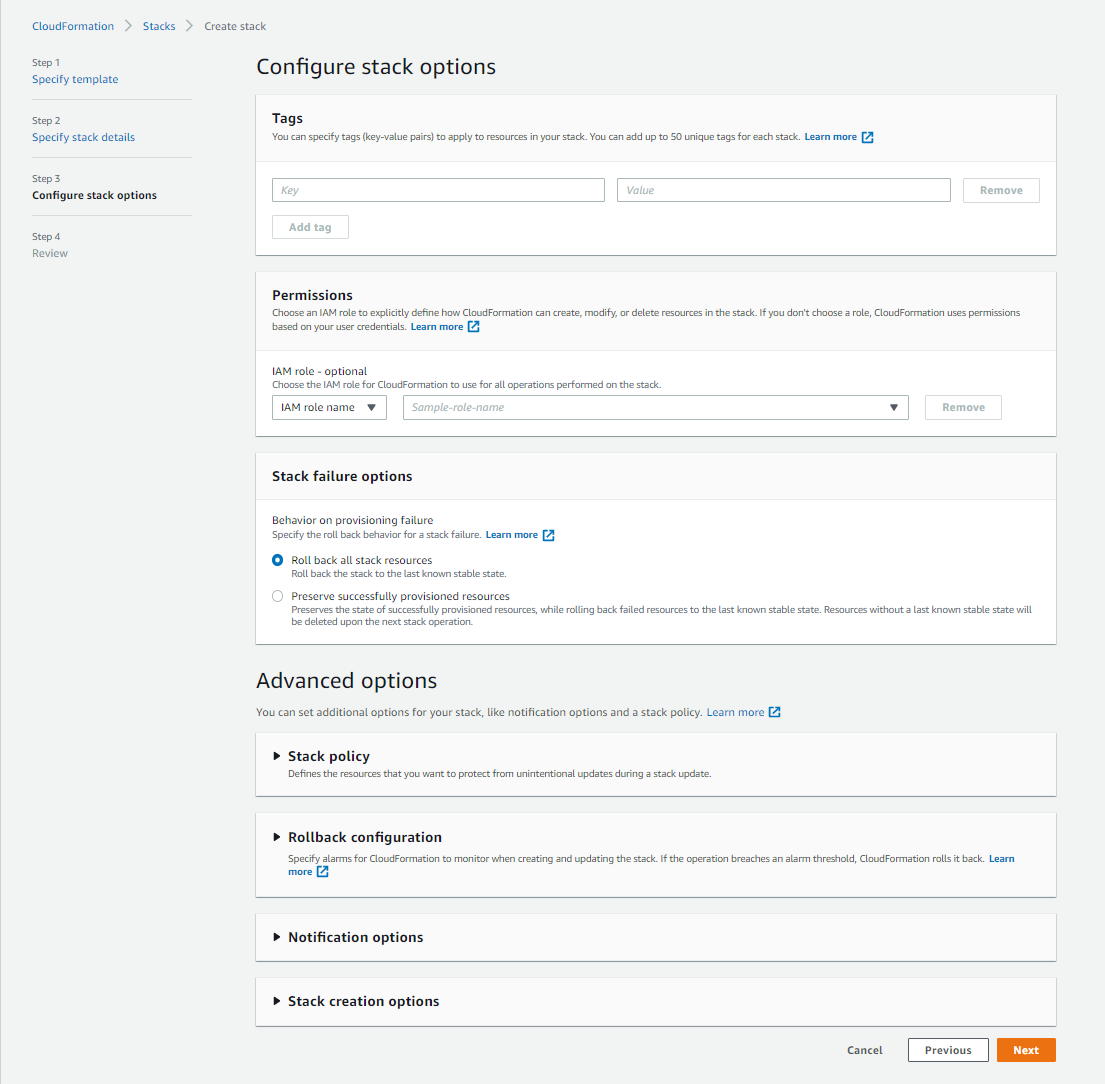
1. Crear Pila



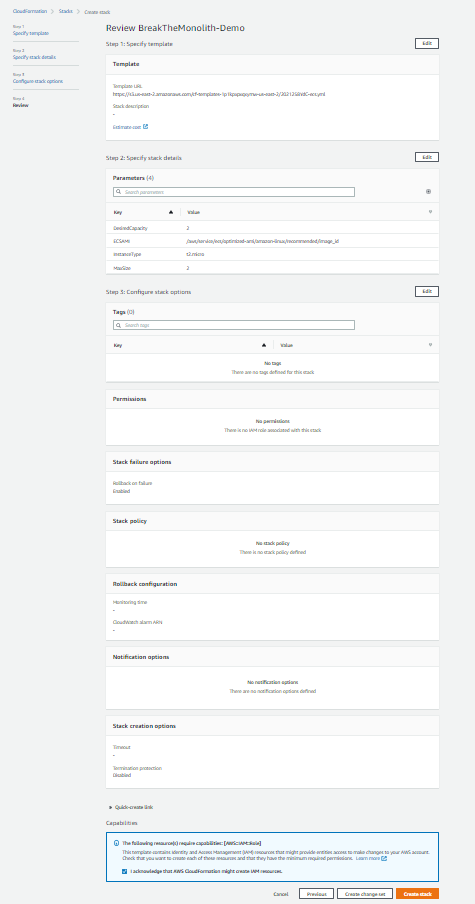
1. Especificar detalles de la pila



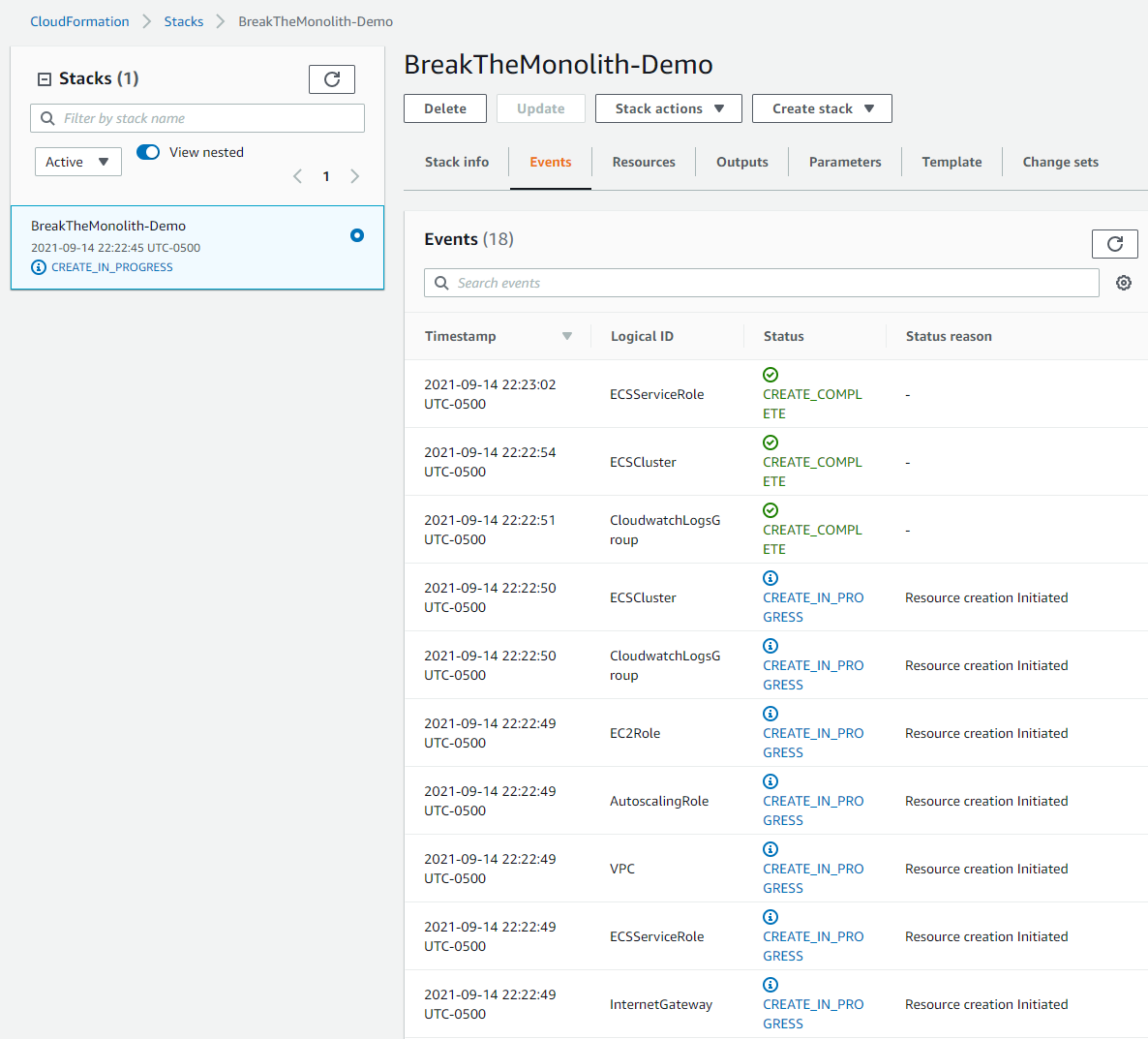
1. Configurar opciones de la pila



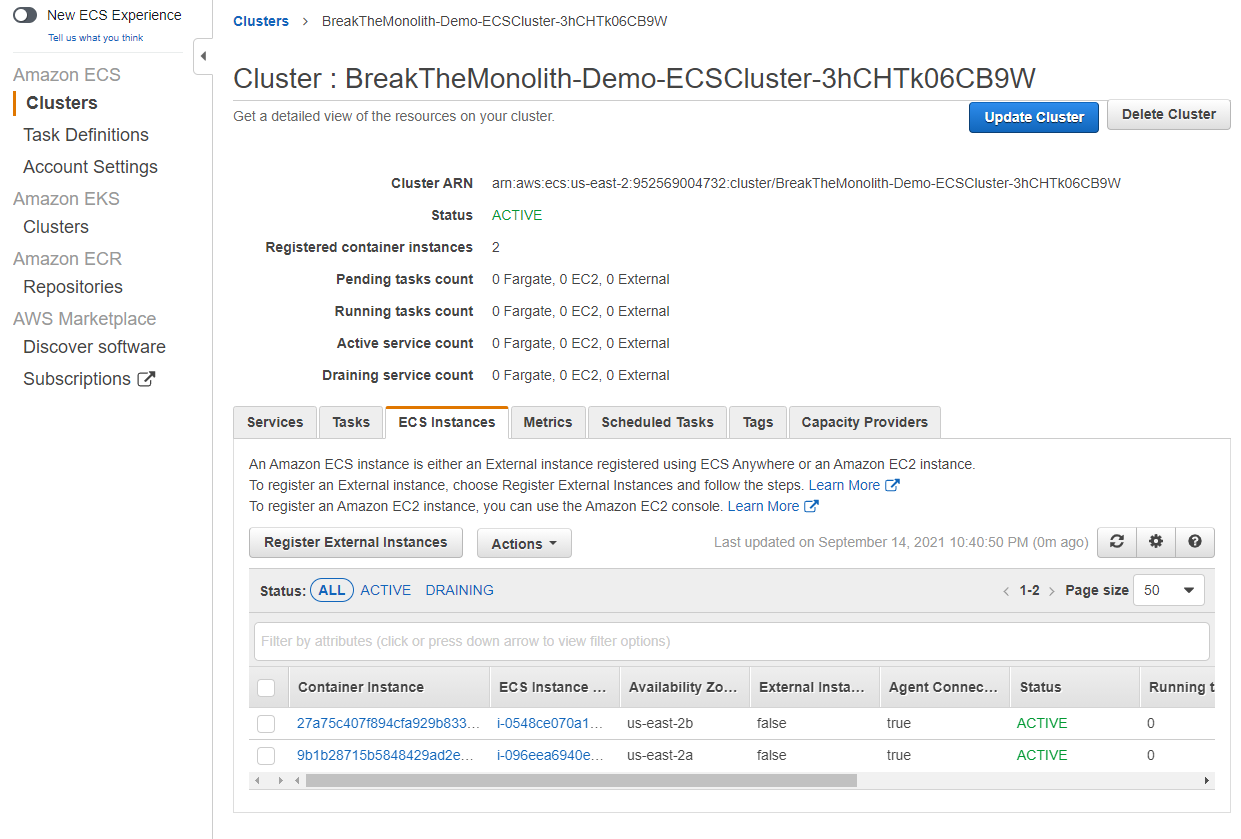
1. Verificar los datos y aceptar los terminos



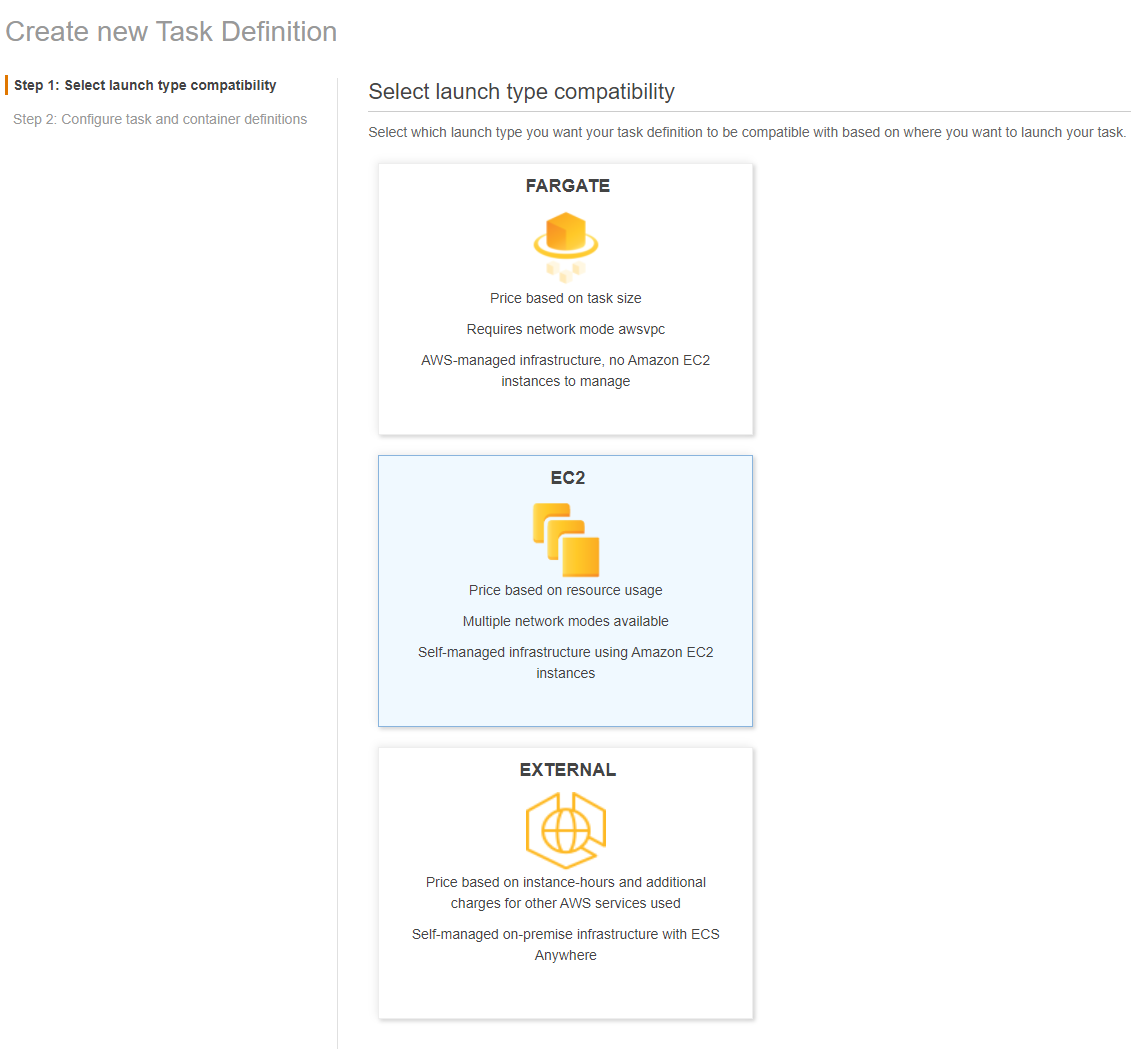
1. Esperar que se creen los recurtsos



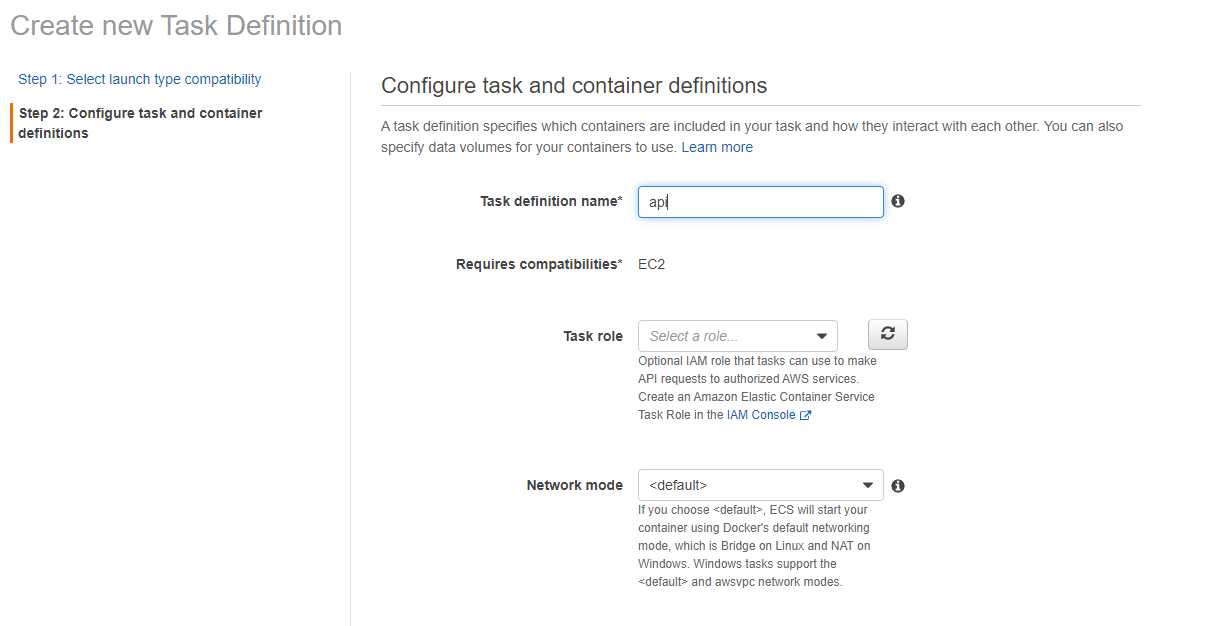
1. Comprobar que el clúster se encuentre en ejecución



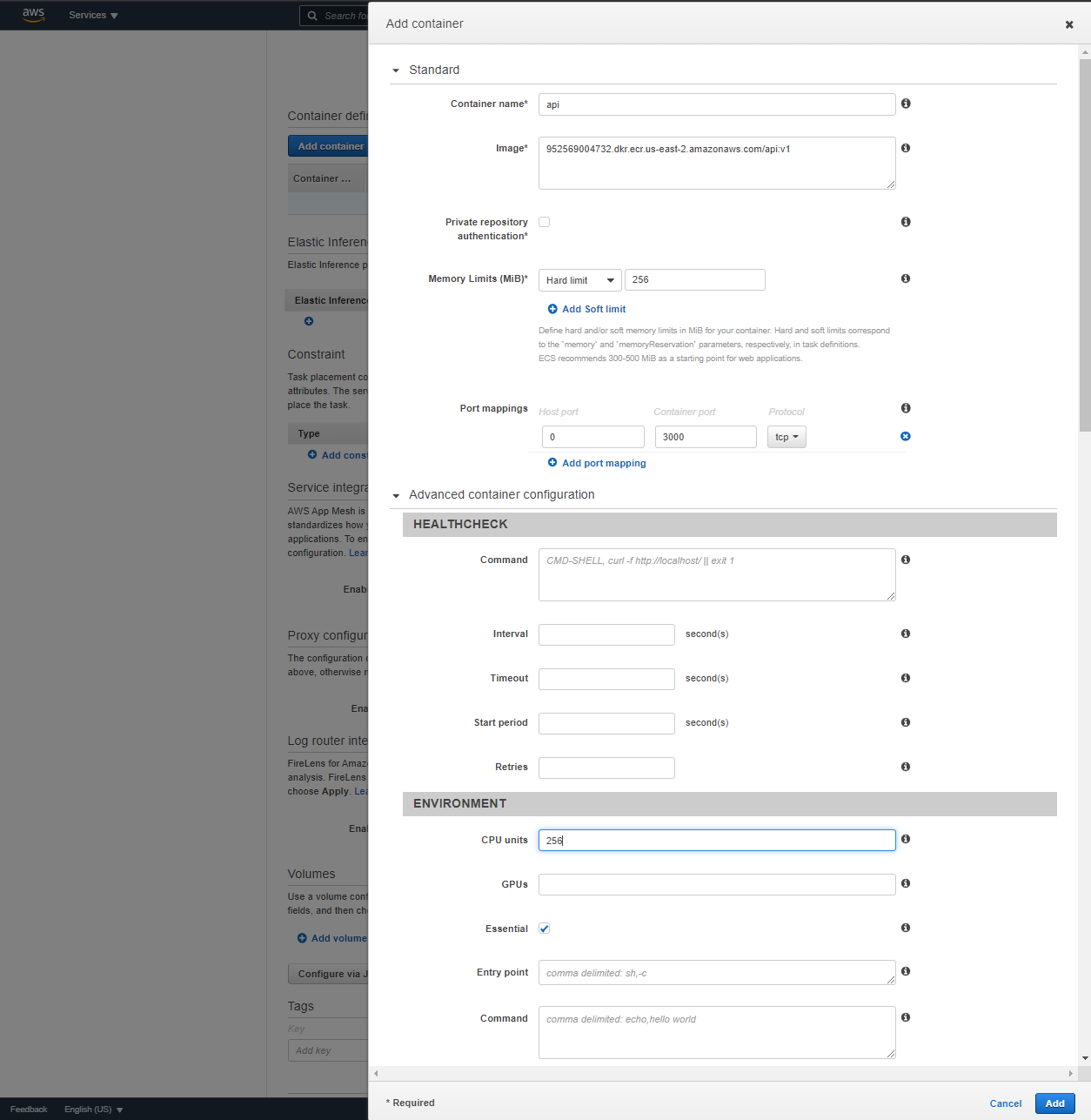
1. Crear la definición de la tarea
   1. Seleccionar el tiop de compatibilidad



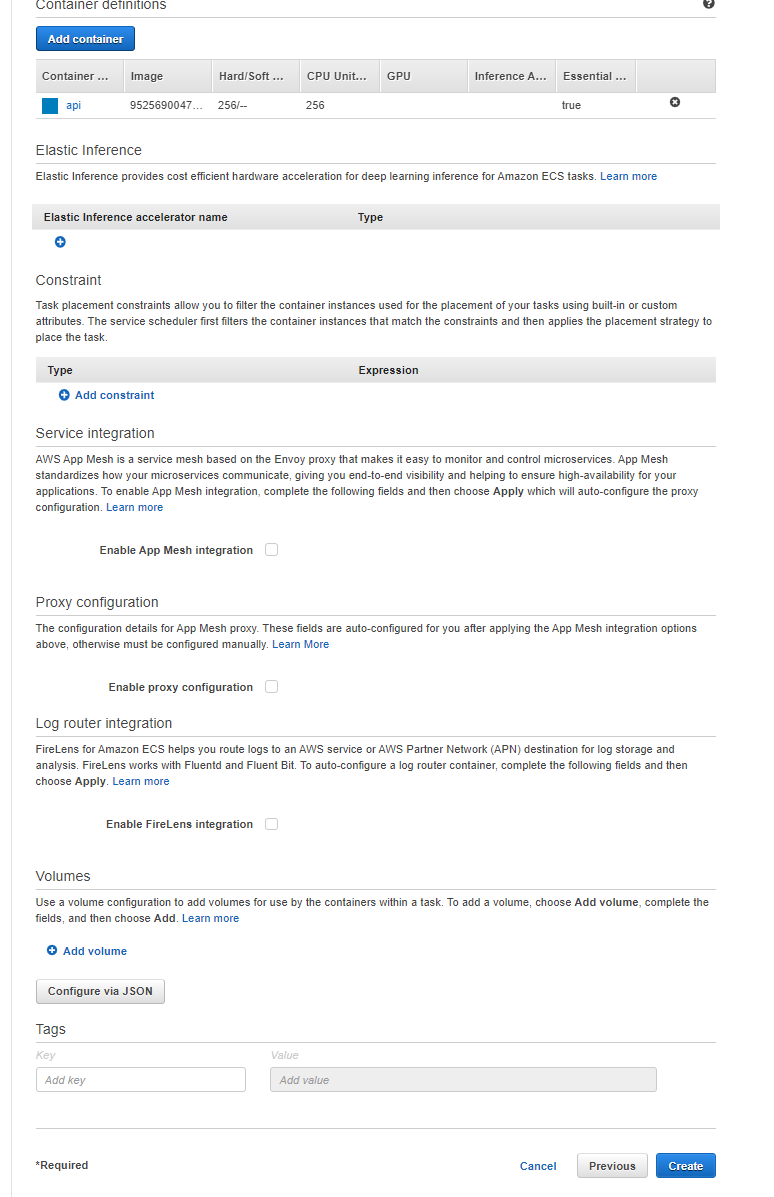
* 1. Configurar la tarea

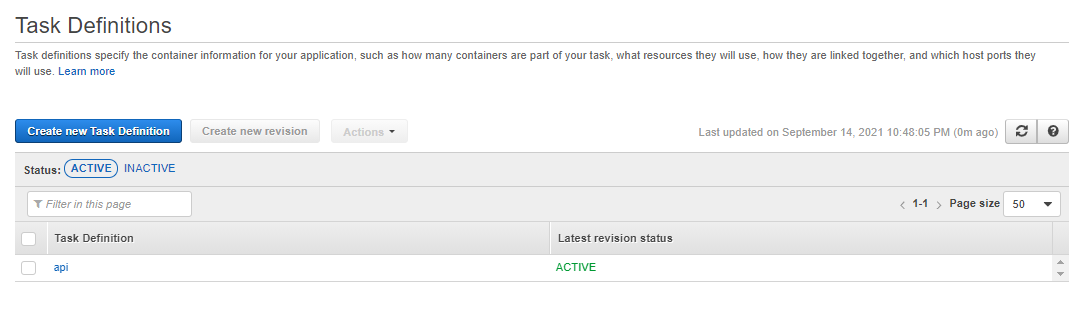


* 1. Agregar contenedor

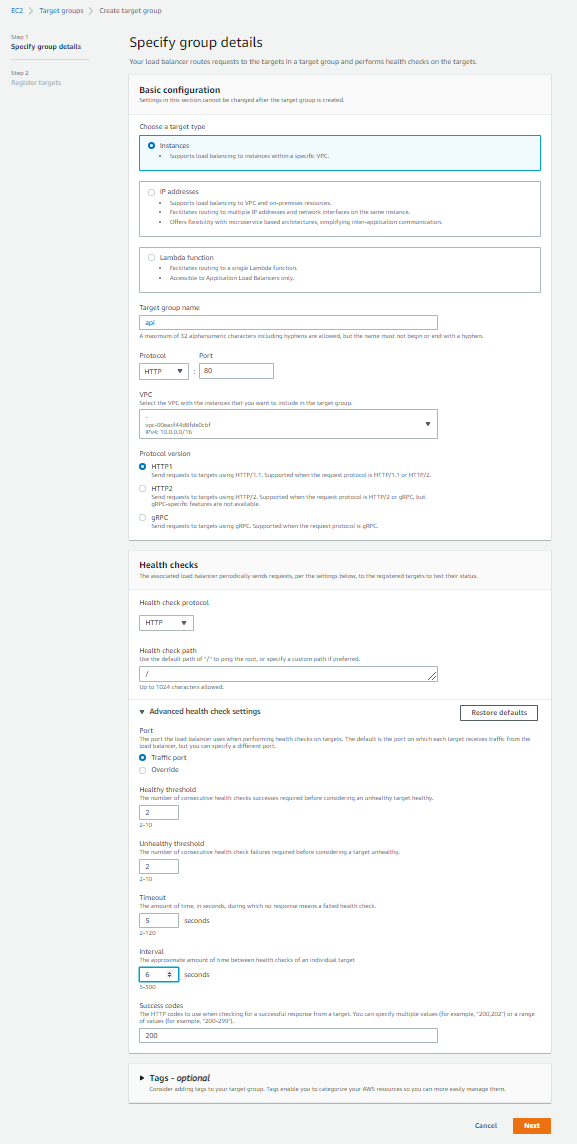


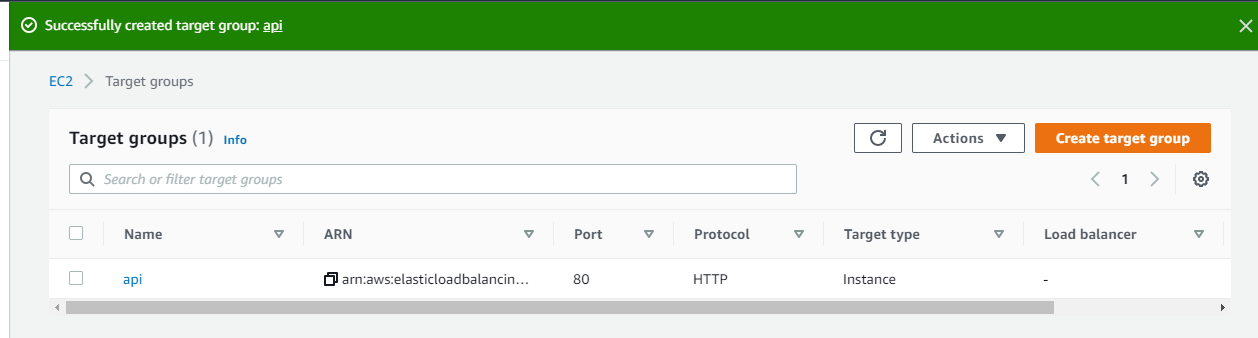
* 1. Crear tarea



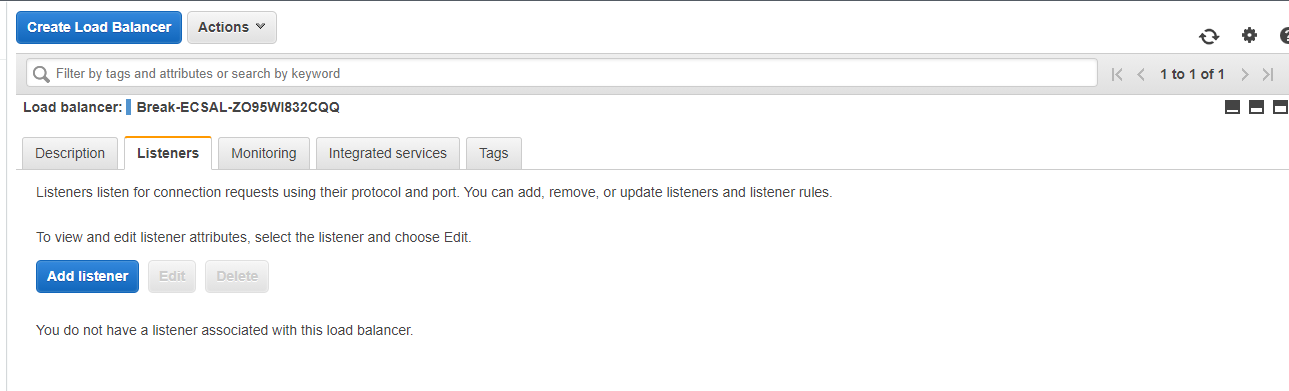


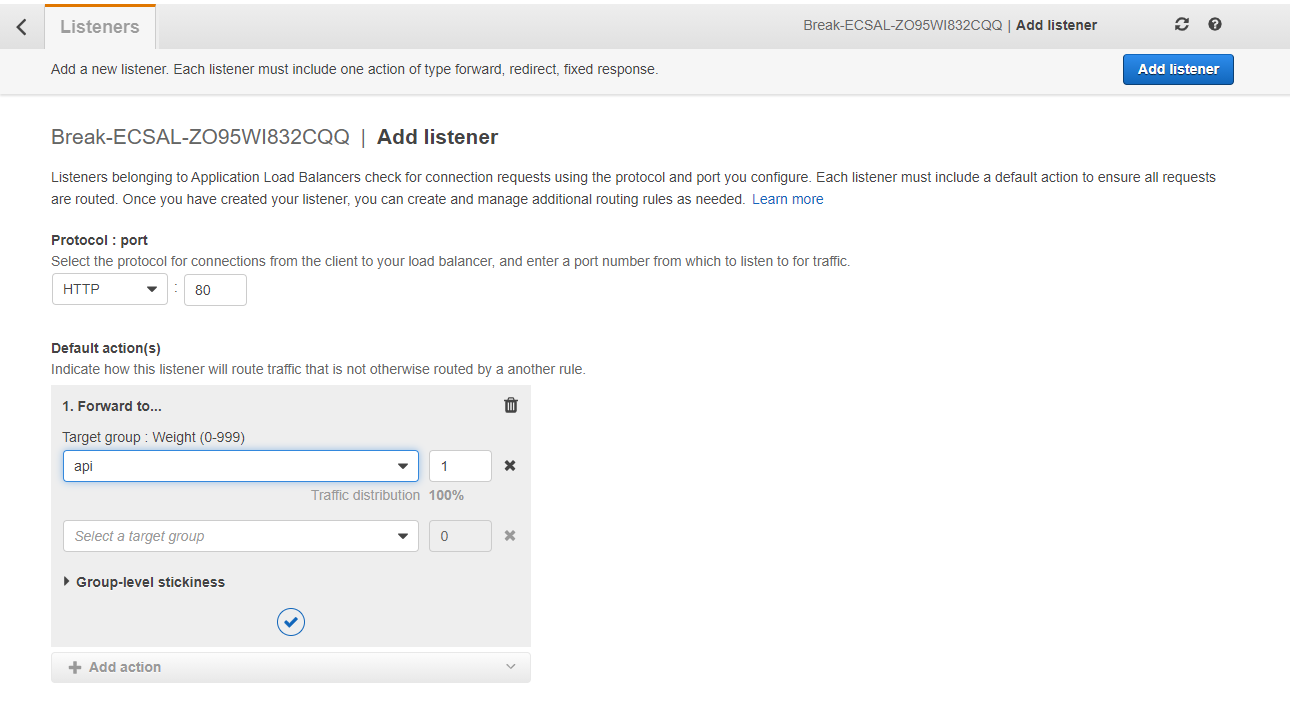
1. Configurar el balanceador de carga de aplicaciones: grupo de destino

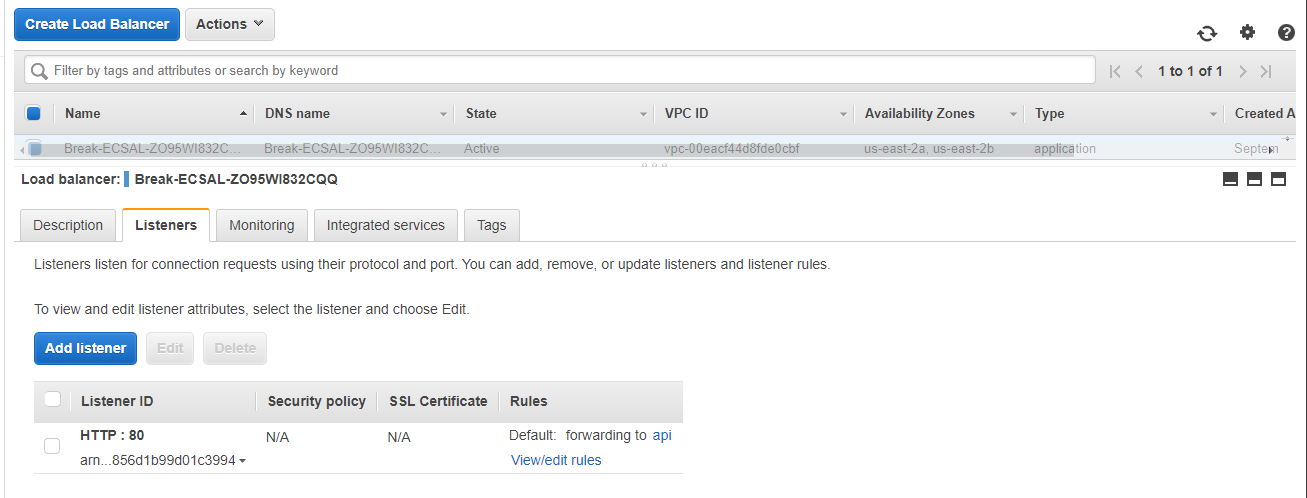




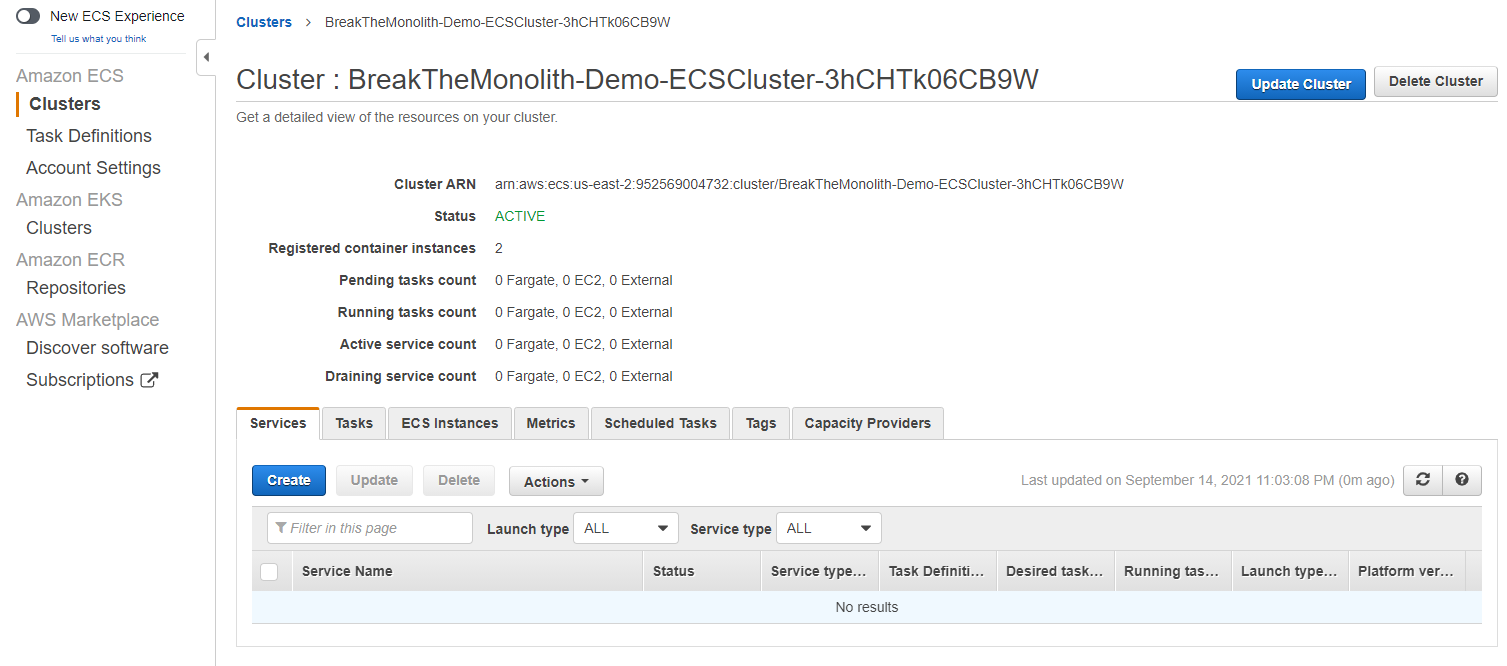
1. Configurar el balanceador de carga: escucha



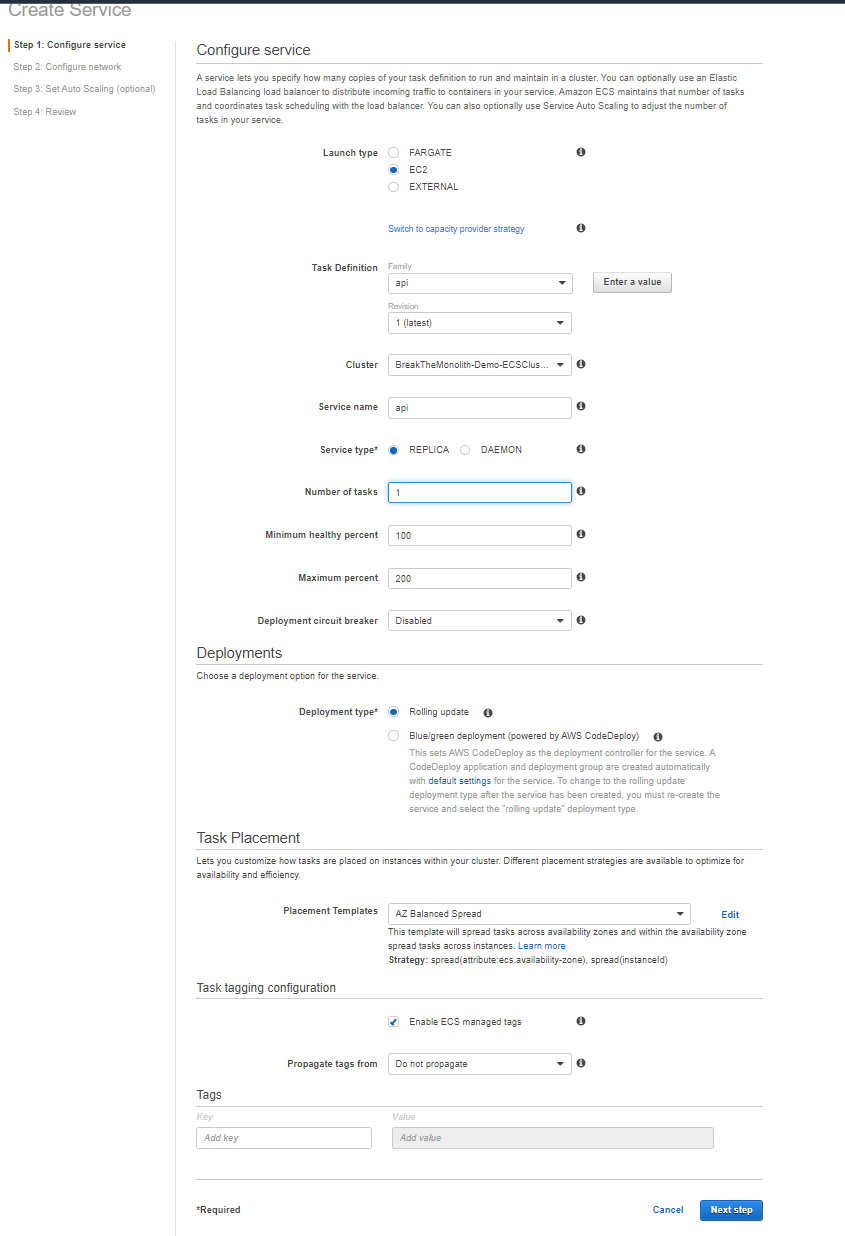




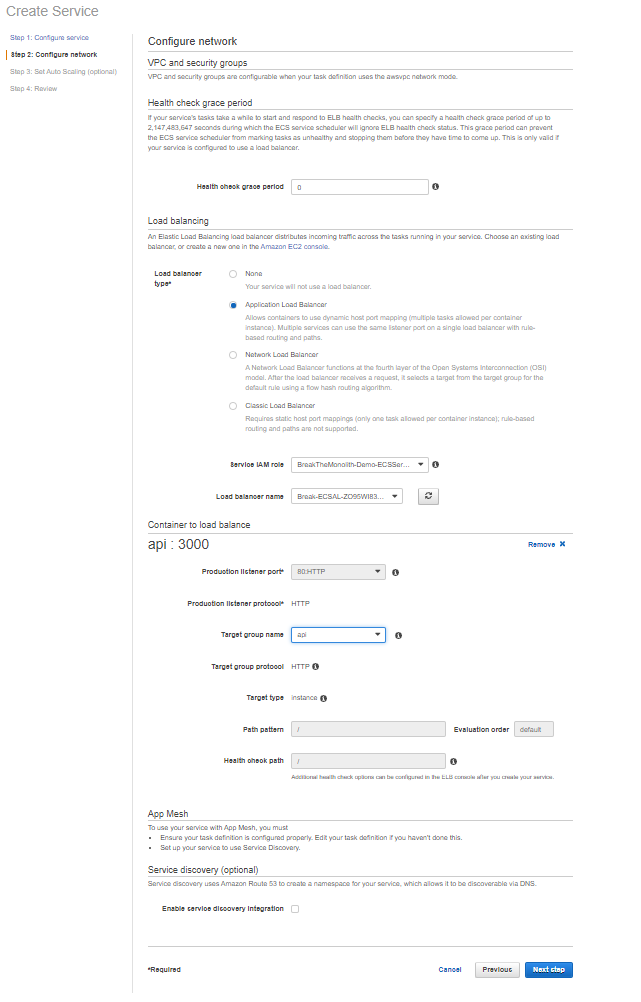
1. Desplegar el monolito como un servicio
   1. Crear el servicio



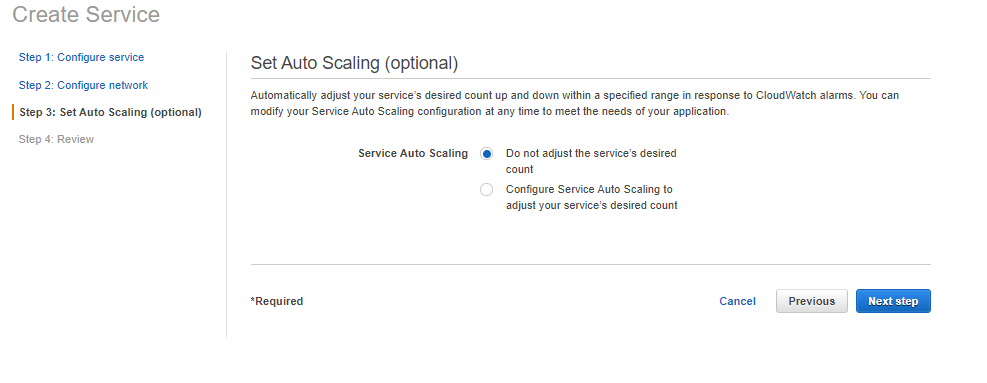
Configurar servicio



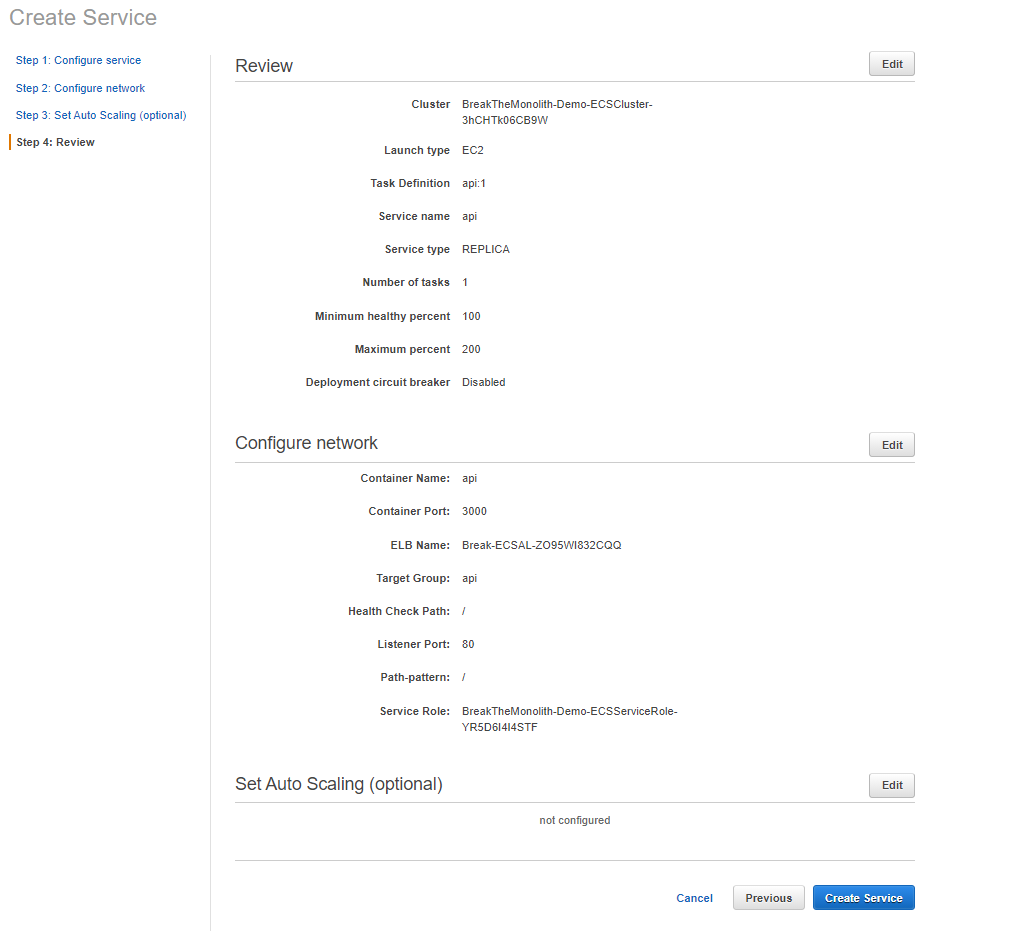
Configurar red

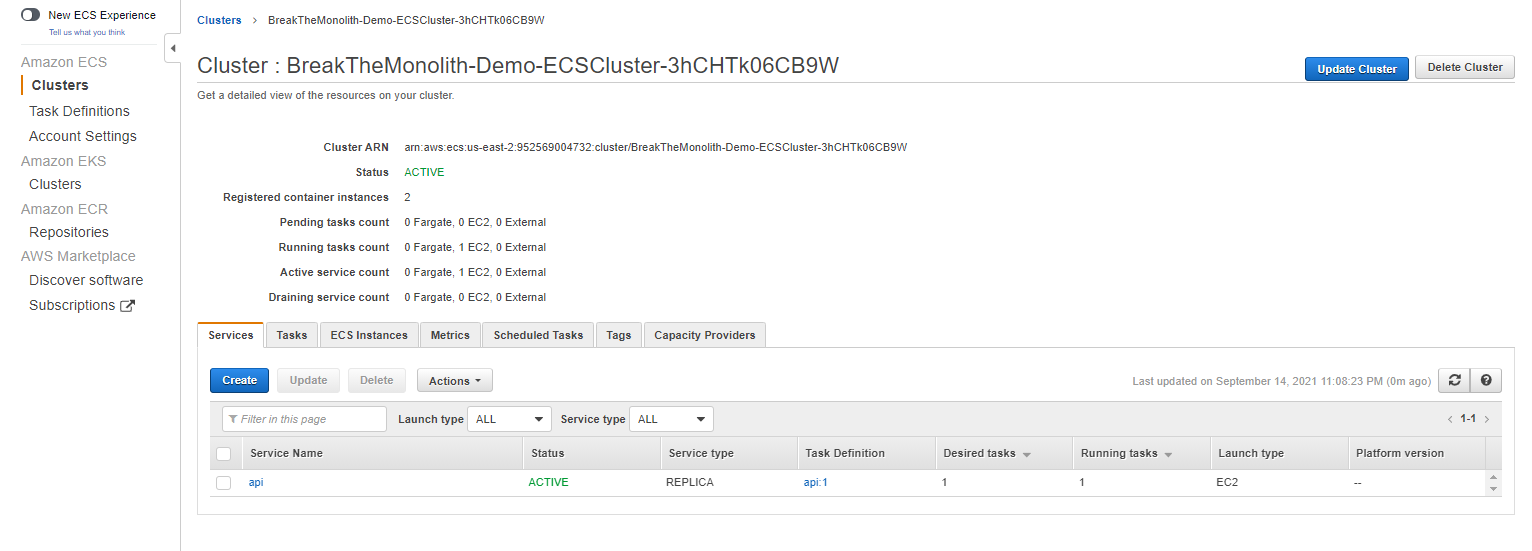


Configurar auto escalamiento (opcional)

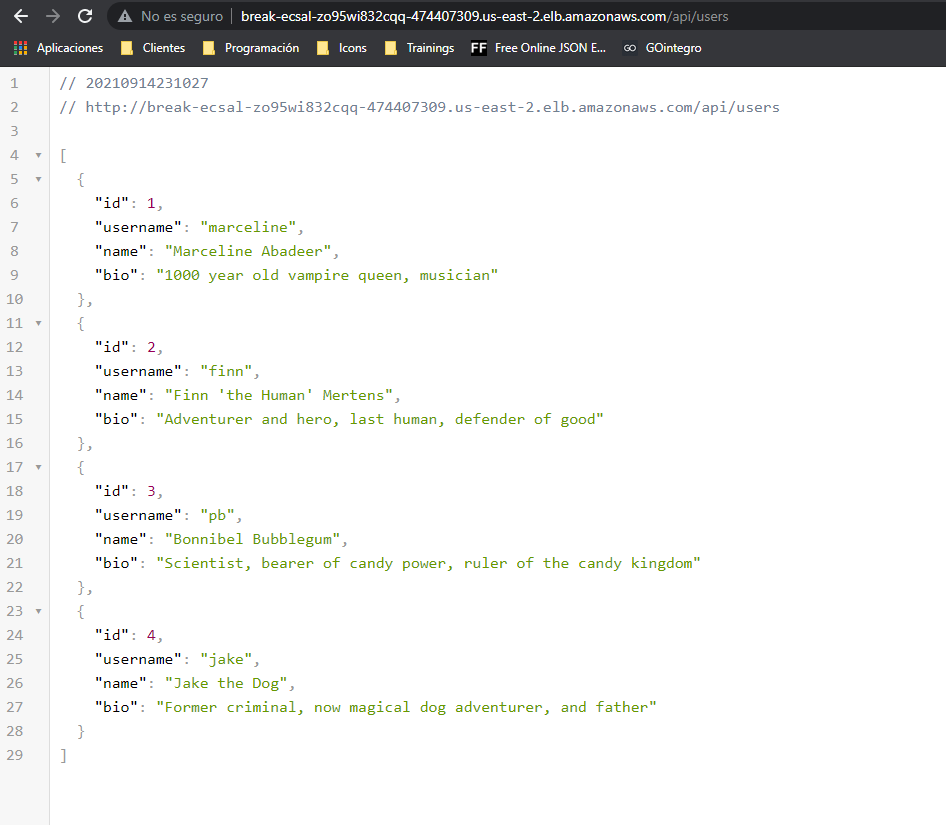


Revisar datos y crear servicio

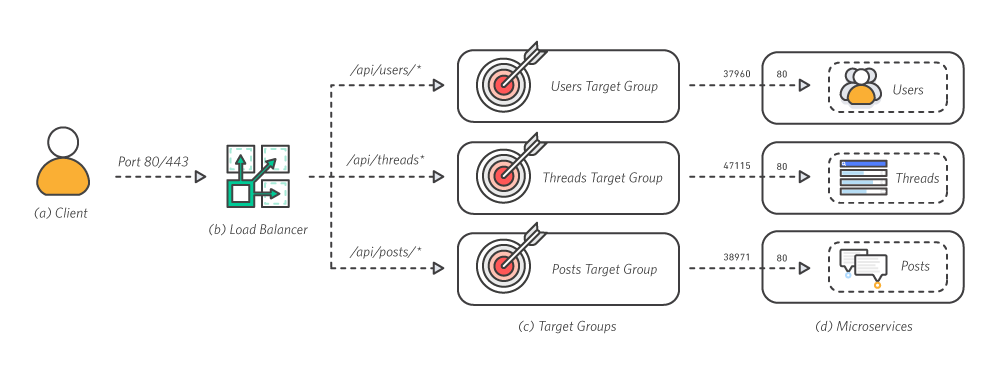




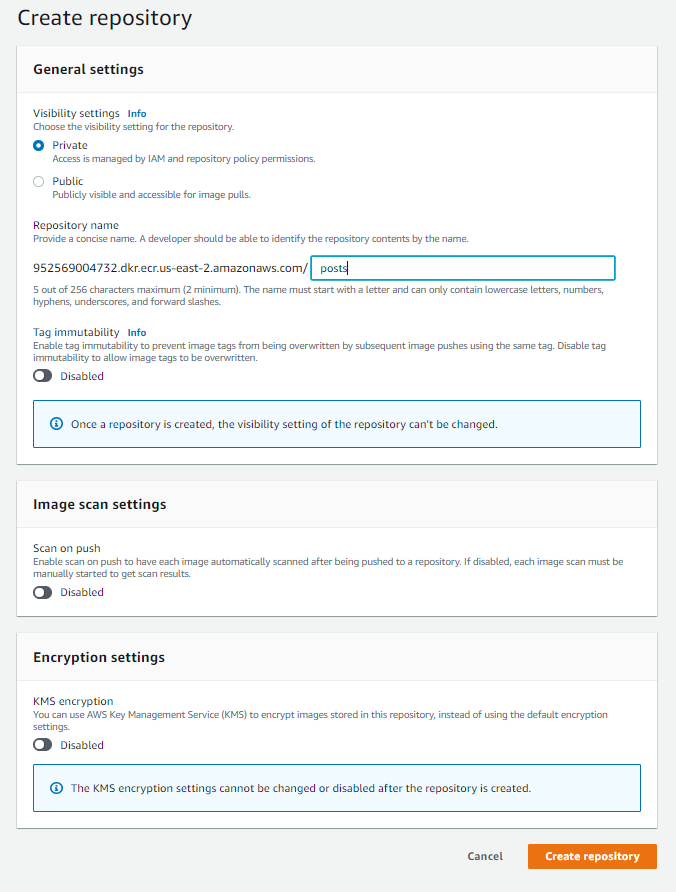
1. Probar el monolito

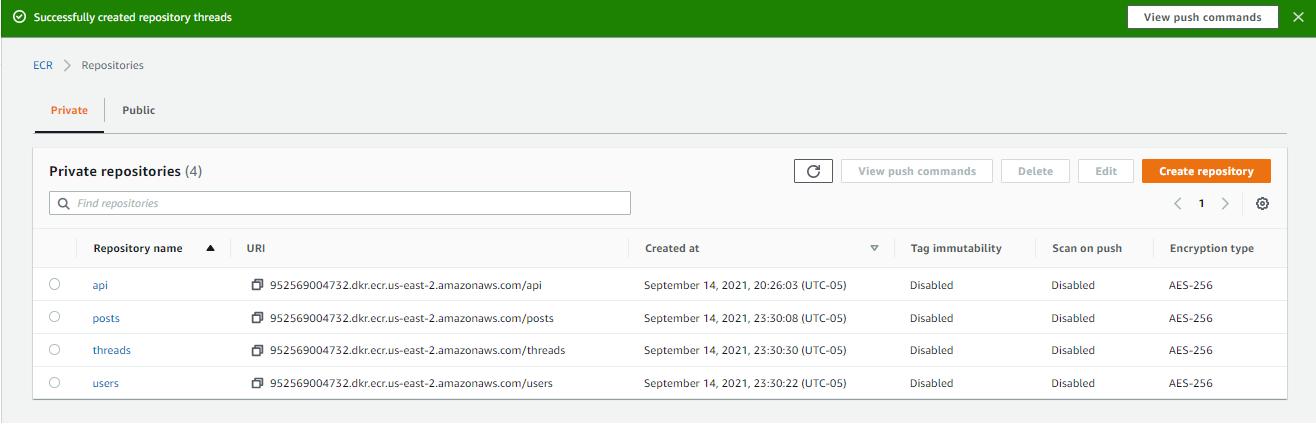


## Romper el monolito



1. Provisionar los repositorios ECR para cada uno de los endpoints (posts, users, threads)





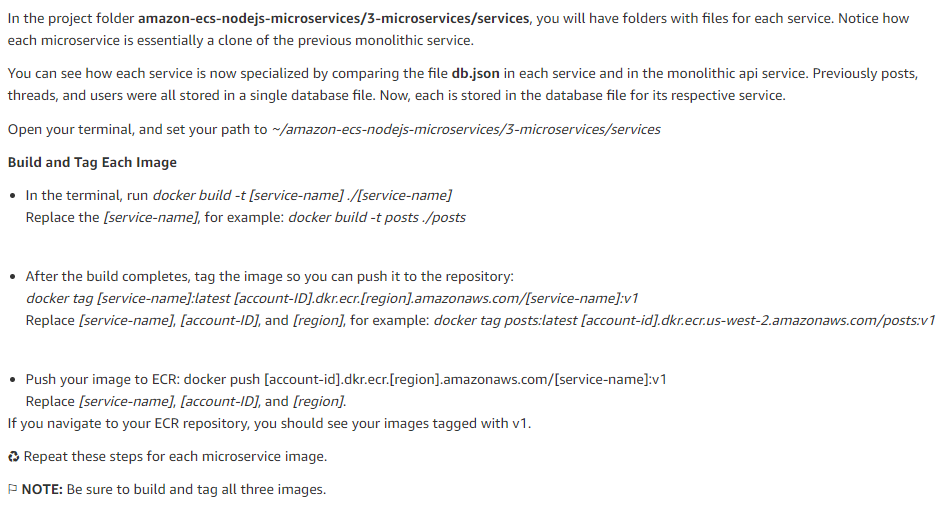
1. Autenticar Docker con AWS (Opcional)

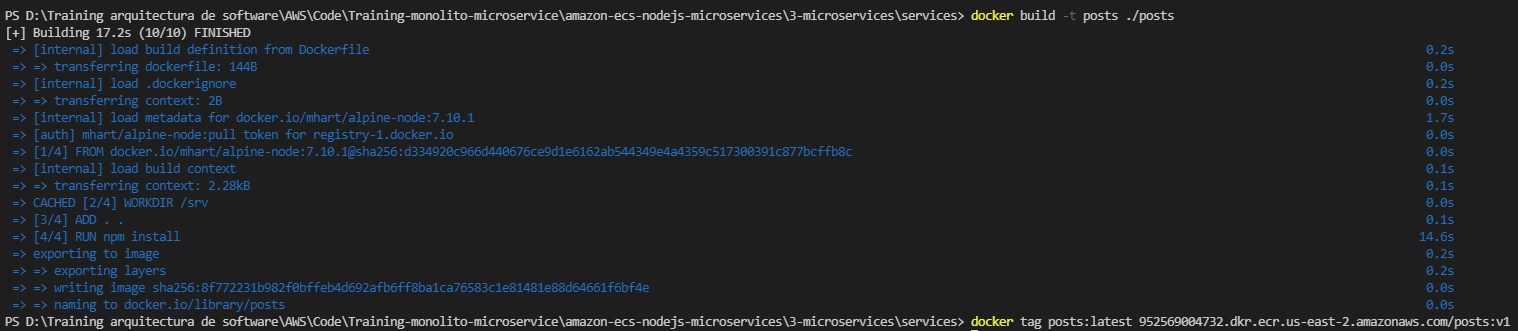
Se puede ejecutar alguno de los comandos para verificar que se encuentre logueado:

* 1. aws ecr get-login-password --region us-east-2 | docker login --username AWS --password-stdin 952569004732.dkr.ecr.us-east-2.amazonaws.com
  2. $(aws ecr get-login --no-include-email --region us-east-2)

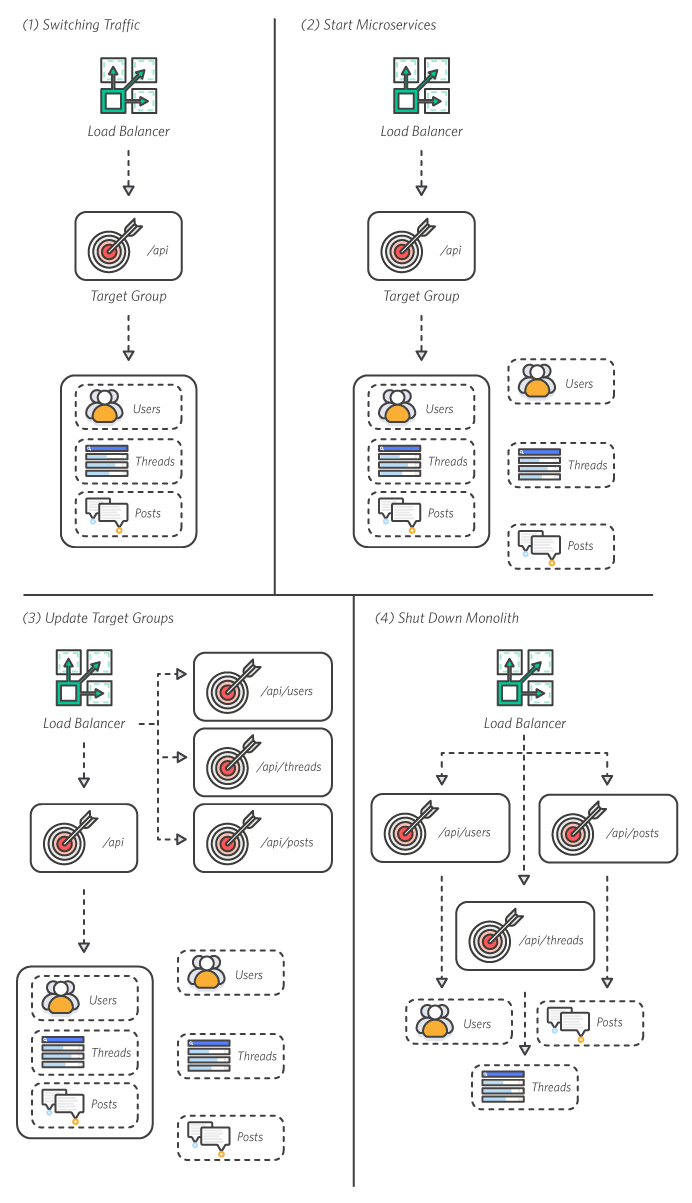


1. Construir y hacer push a cada servicio

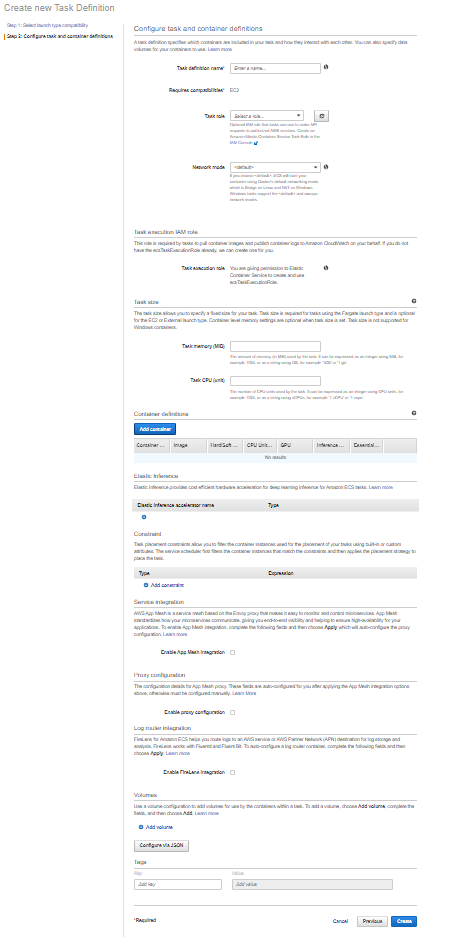




# Desplegar los microservicios



1. Escribir definiciones de tareas para los servicios
   1. Para mayor rapidez se puede hacer la configuración por JSON



* Name = *[service-name: posts, threads, and users]*
* Image =*[Amazon ECR repository image URL]:latest*
* cpu =*256*
* memory = *256*
* Container Port = *3000*
* Host Post = *0*

{

"containerDefinitions": [

{

"name": "[service-name]",

"image": "[account-id].dkr.ecr.[region].amazonaws.com/[service-name]:[tag]",

"memoryReservation": "256",

"cpu": "256",

"essential": true,

"portMappings": [

{

"hostPort": "0",

"containerPort": "3000",

"protocol": "tcp"

}

]

}

],

"volumes": [],

"networkMode": "bridge",

"placementConstraints": [],

"family": "[service-name]"

}

1. Configurar el balanceador de aplicación: Target groups
   1. Se pueden crear a través del siguiente comando:

aws elbv2 create-target-group --region [region] --name [service-name] --protocol HTTP --port 80 --vpc-id [vpc-attribute] --healthy-threshold-count 2 --unhealthy-threshold-count 2 --health-check-timeout-seconds 5 --health-check-interval-seconds 6

* + 1. Cambiar region, service-name y vpc-attribute por los datos correspondientes

PS D:\Training arquitectura de software\AWS\Code\Training-monolito-microservice\amazon-ecs-nodejs-microservices\3-microservices\services> aws elbv2 create-target-group --region us-east-2 --name posts --protocol HTTP --port 80 --vpc-id vpc-00eacf44d8fde0cbf --healthy-threshold-count 2 --unhealthy-threshold-count 2 --health-check-timeout-seconds 5 --health-check-interval-seconds 6

{

"TargetGroups": [

{

"TargetGroupArn": "arn:aws:elasticloadbalancing:us-east-2:952569004732:targetgroup/posts/cfaf4f96d98b404b",

"TargetGroupName": "posts",

"Protocol": "HTTP",

"Port": 80,

"VpcId": "vpc-00eacf44d8fde0cbf",

"HealthCheckProtocol": "HTTP",

"HealthCheckPort": "traffic-port",

"HealthCheckEnabled": true,

"HealthCheckIntervalSeconds": 6,

"HealthCheckTimeoutSeconds": 5,

"HealthyThresholdCount": 2,

"UnhealthyThresholdCount": 2,

"HealthCheckPath": "/",

"Matcher": {

"HttpCode": "200"

},

"TargetType": "instance",

"ProtocolVersion": "HTTP1"

}

]

}

PS D:\Training arquitectura de software\AWS\Code\Training-monolito-microservice\amazon-ecs-nodejs-microservices\3-microservices\services> aws elbv2 create-target-group --region us-east-2 --name threads --protocol HTTP --port 80 --vpc-id vpc-00eacf44d8fde0cbf --healthy-threshold-count 2 --unhealthy-threshold-count 2 --health-check-timeout-seconds 5 --health-check-interval-seconds 6

{

"TargetGroups": [

{

"TargetGroupArn": "arn:aws:elasticloadbalancing:us-east-2:952569004732:targetgroup/threads/dfea25c7d9bdbddc",

"TargetGroupName": "threads",

"Protocol": "HTTP",

"Port": 80,

"VpcId": "vpc-00eacf44d8fde0cbf",

"HealthCheckProtocol": "HTTP",

"HealthCheckPort": "traffic-port",

"HealthCheckEnabled": true,

"HealthCheckIntervalSeconds": 6,

"HealthCheckTimeoutSeconds": 5,

"HealthyThresholdCount": 2,

"UnhealthyThresholdCount": 2,

"HealthCheckPath": "/",

"Matcher": {

"HttpCode": "200"

},

"TargetType": "instance",

"ProtocolVersion": "HTTP1"

}

]

}

PS D:\Training arquitectura de software\AWS\Code\Training-monolito-microservice\amazon-ecs-nodejs-microservices\3-microservices\services> aws elbv2 create-target-group --region us-east-2 --name users --protocol HTTP --port 80 --vpc-id vpc-00eacf44d8fde0cbf --healthy-threshold-count 2 --unhealthy-threshold-count 2 --health-check-timeout-seconds 5 --health-check-interval-seconds 6

{

"TargetGroups": [

{

"TargetGroupArn": "arn:aws:elasticloadbalancing:us-east-2:952569004732:targetgroup/users/e22d1d0d5de3fea1",

"TargetGroupName": "users",

"Protocol": "HTTP",

"Port": 80,

"VpcId": "vpc-00eacf44d8fde0cbf",

"HealthCheckProtocol": "HTTP",

"HealthCheckPort": "traffic-port",

"HealthCheckEnabled": true,

"HealthCheckIntervalSeconds": 6,

"HealthCheckTimeoutSeconds": 5,

"HealthyThresholdCount": 2,

"UnhealthyThresholdCount": 2,

"HealthCheckPath": "/",

"Matcher": {

"HttpCode": "200"

},

"TargetType": "instance",

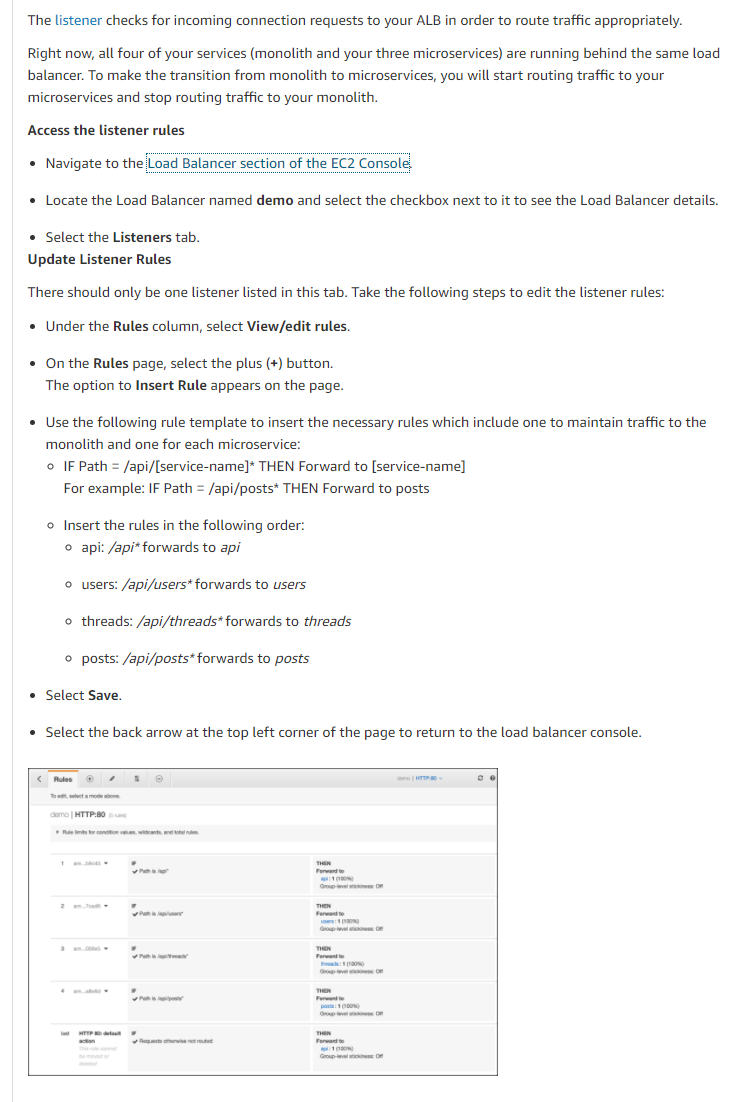
"ProtocolVersion": "HTTP1"

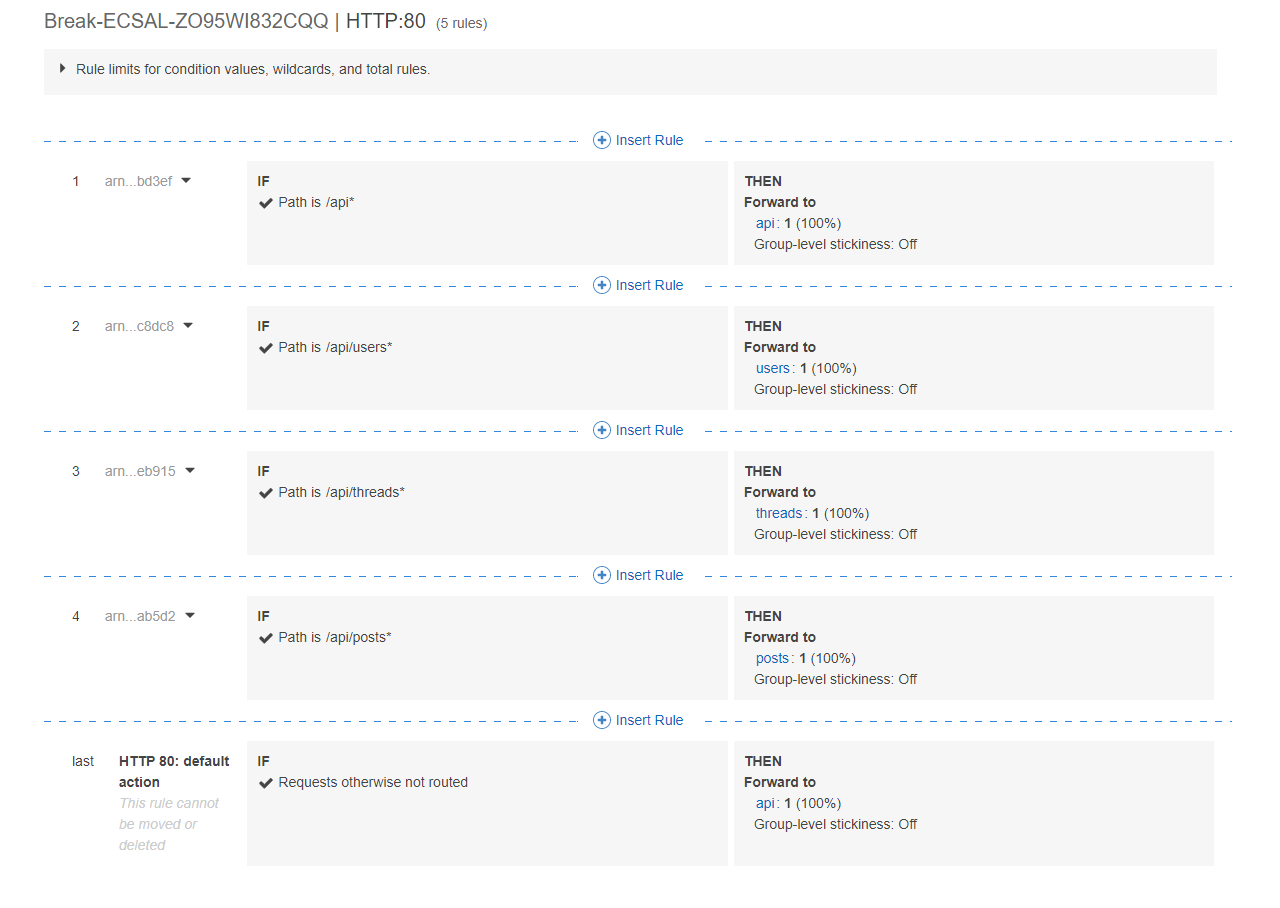
}

]

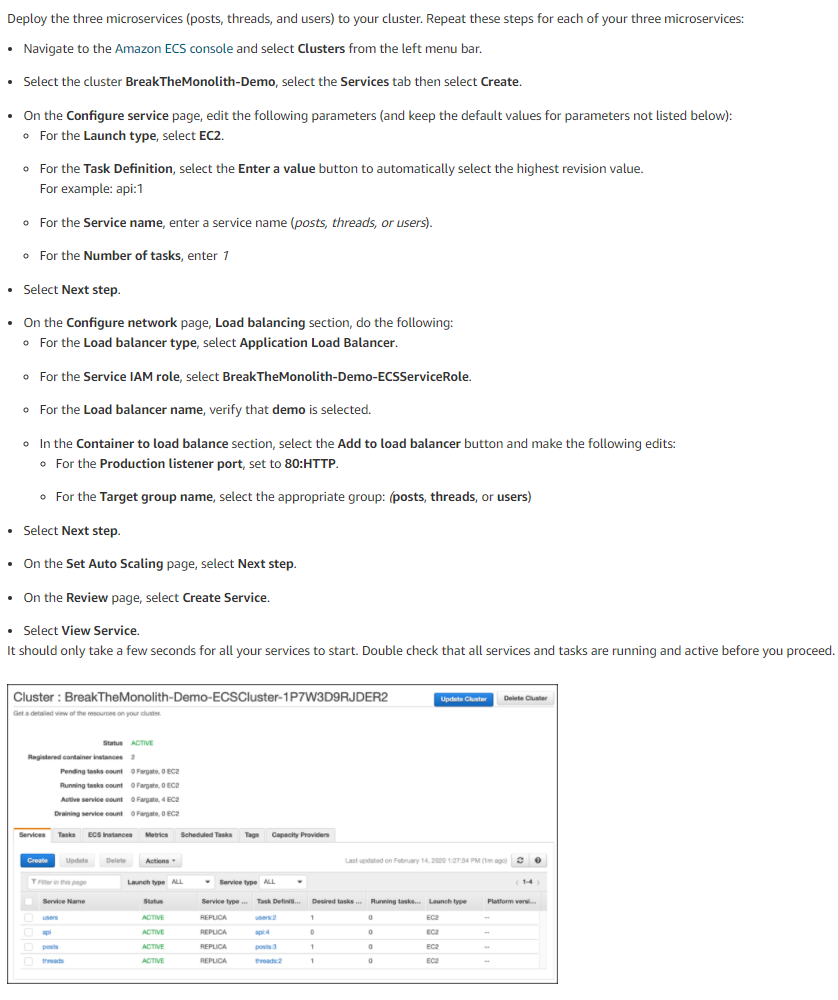
}

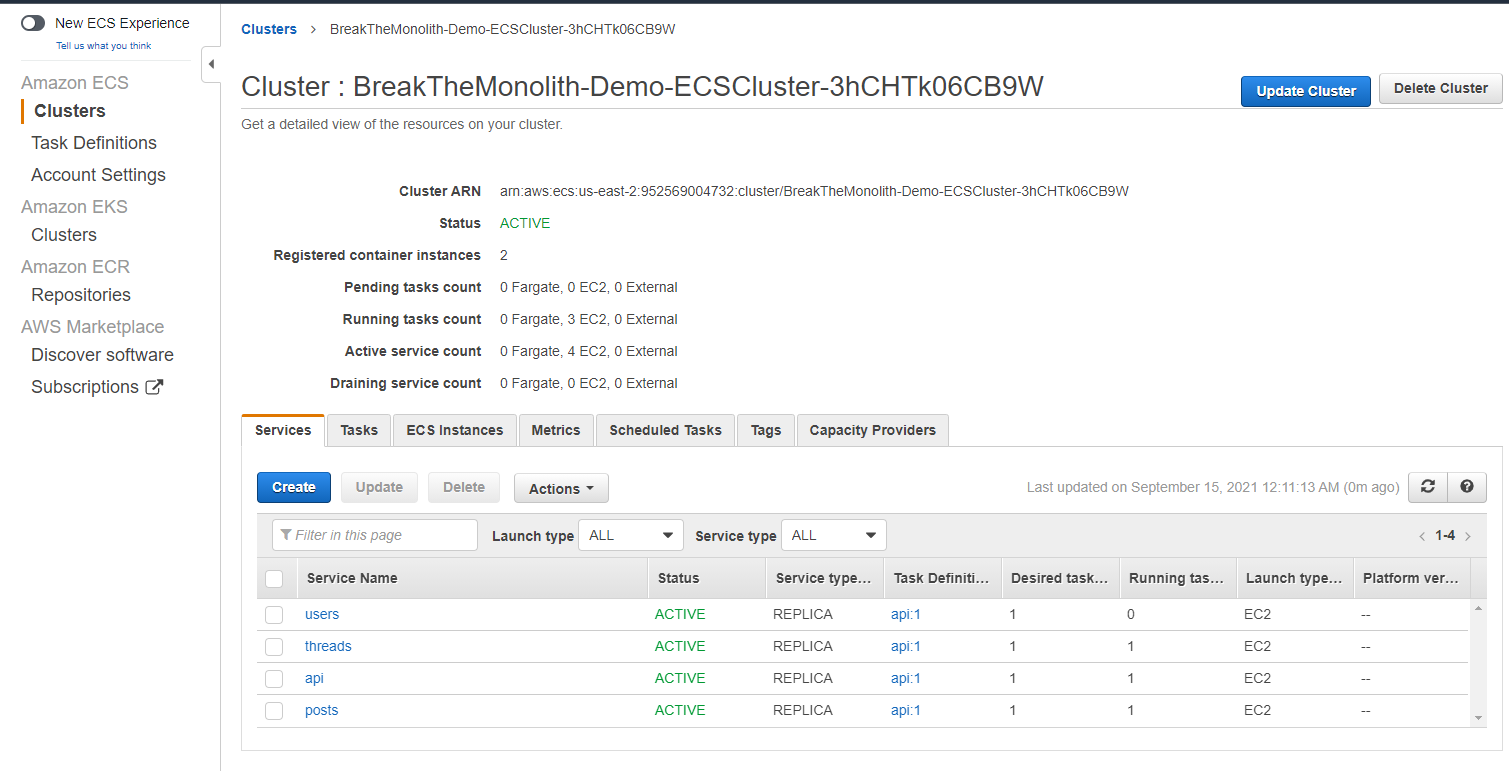
1. Configurar las reglas de escucha



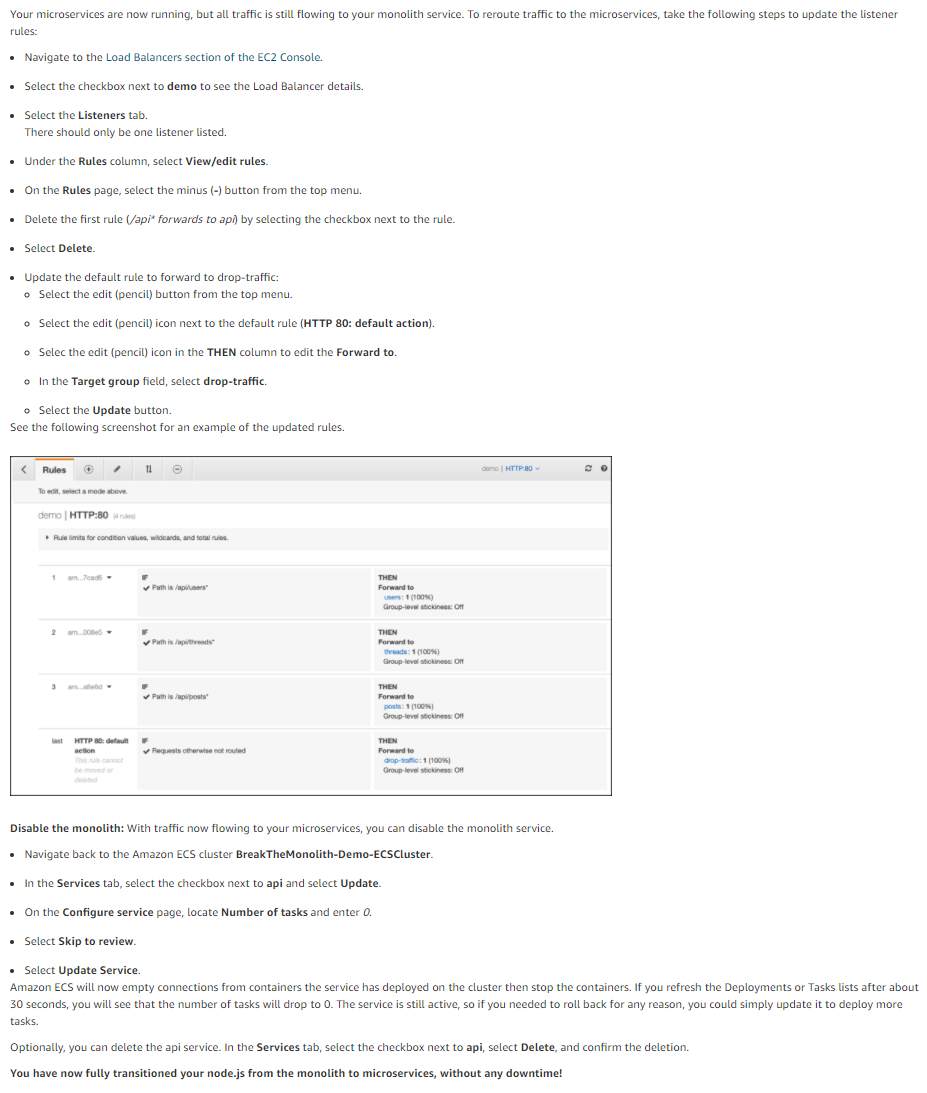


1. Desplegar los microservicios

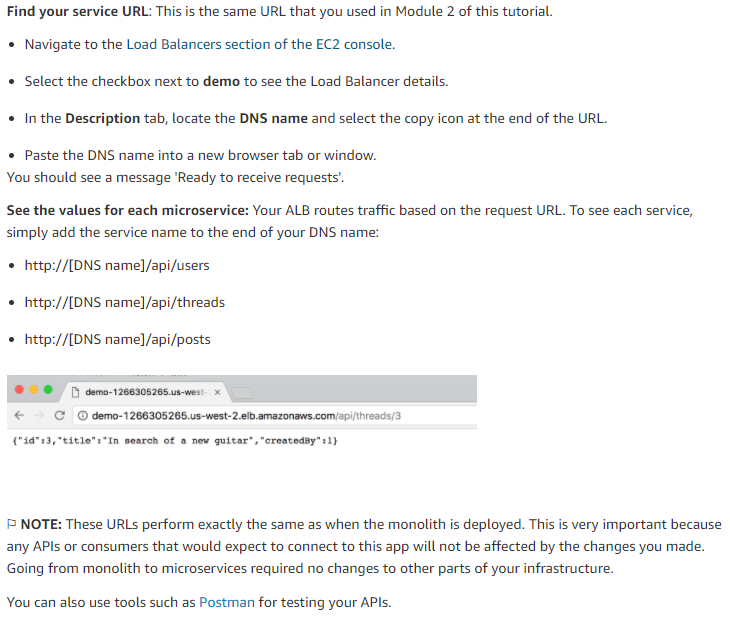


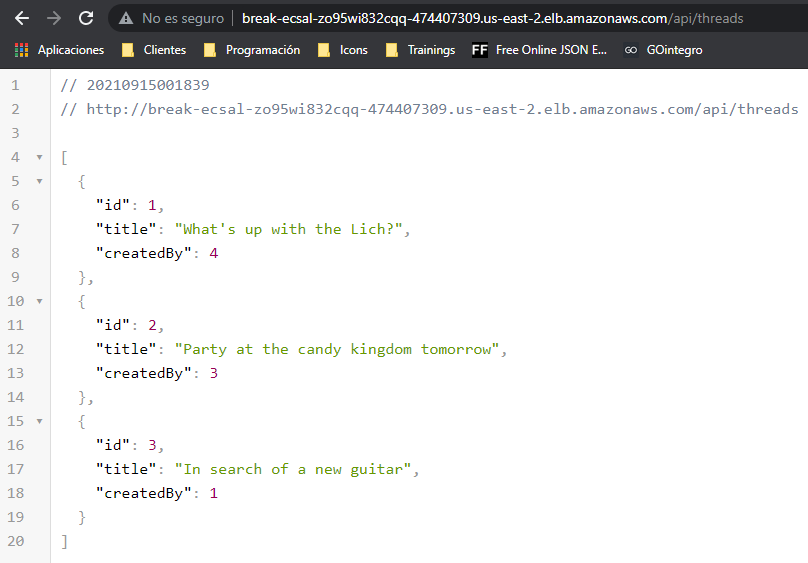


1. Enviar trafico hacia los microservicios



1. Validar despliegue

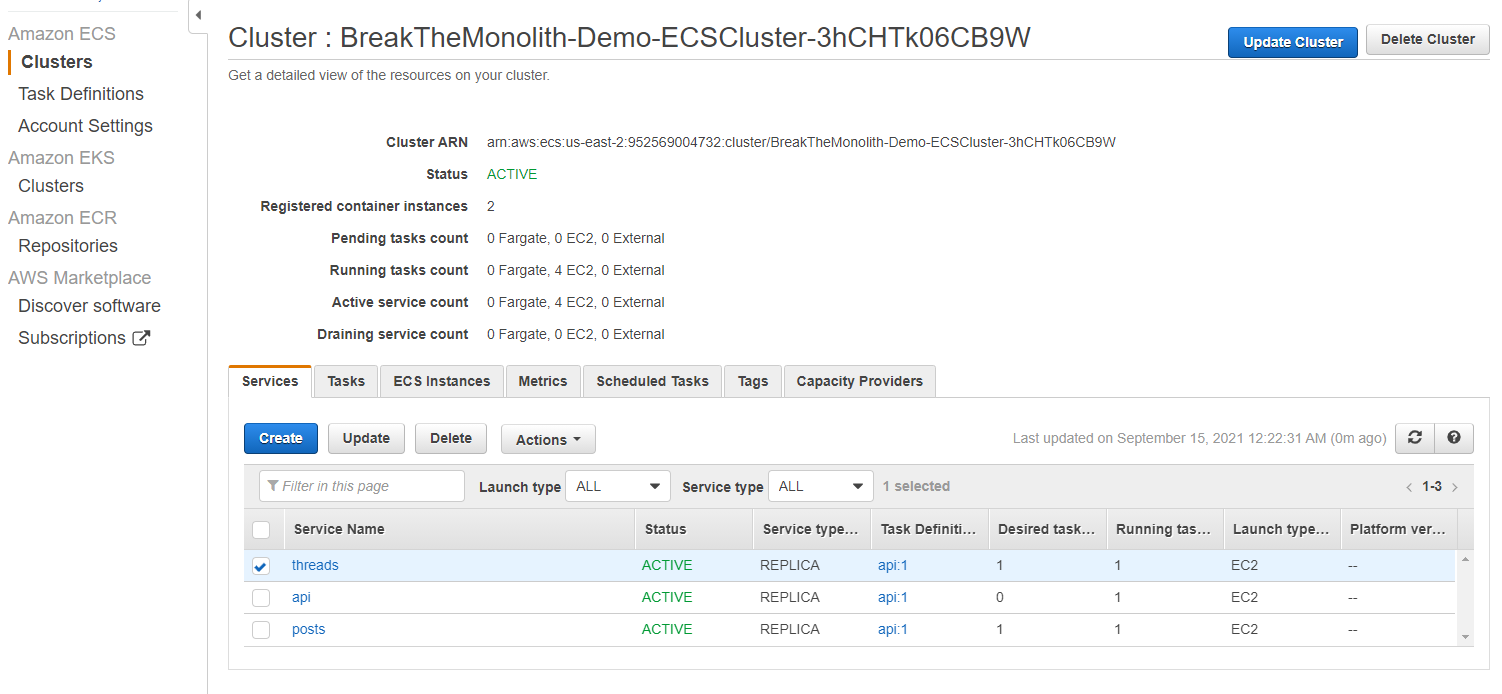




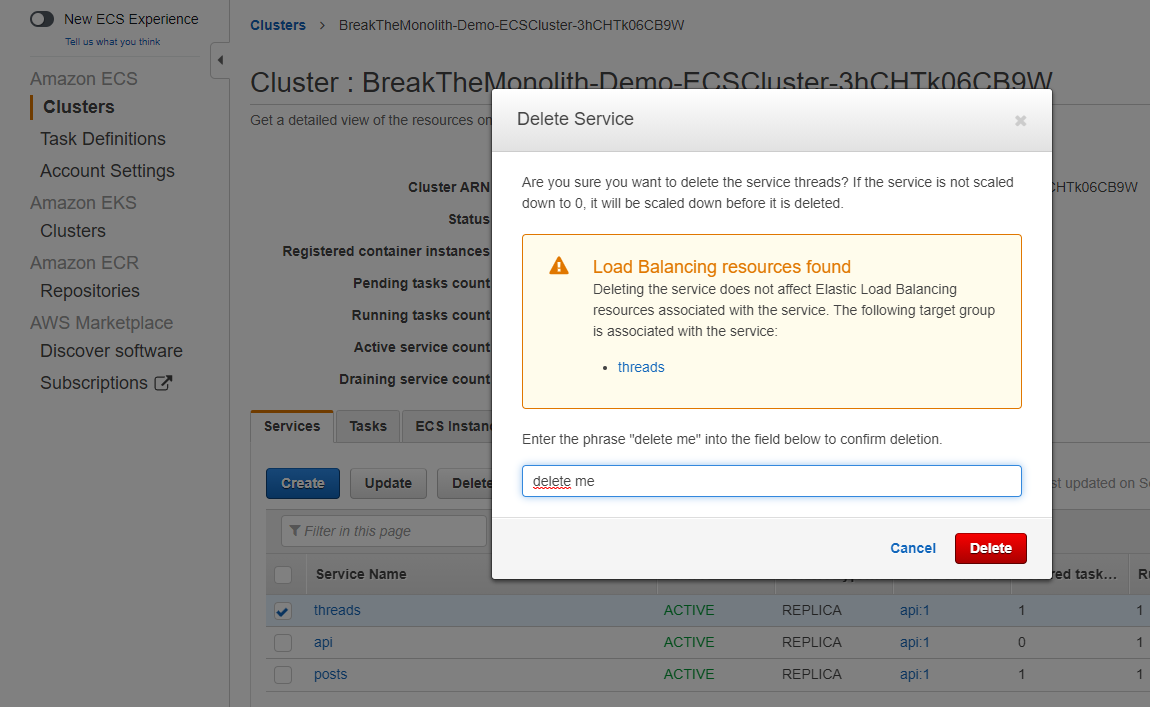
# Limpiar

Esta sección pretnede mostrar como eliminar todos los recursos creados durante este tutorial

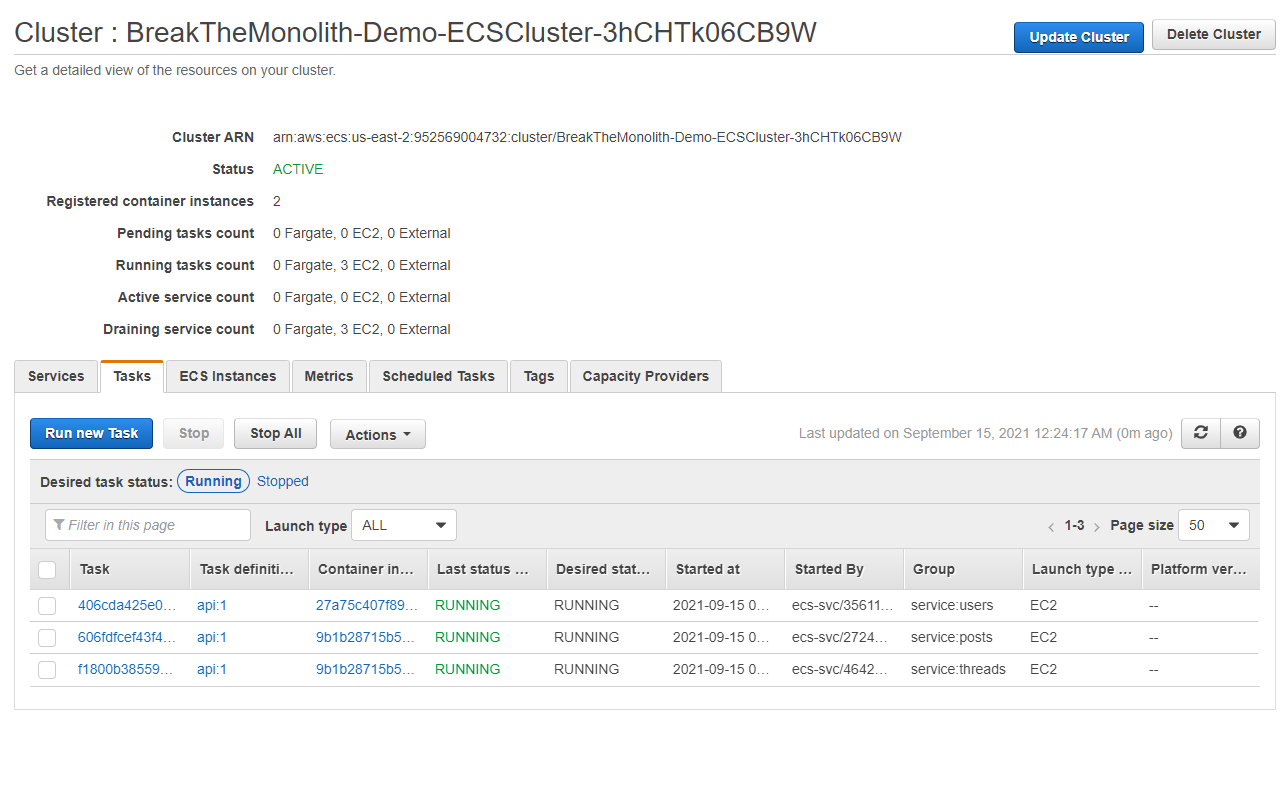
1. Apagar los servicios
   1. Para cada servicio:
      1. Seleccionar y dar clic en eliminar



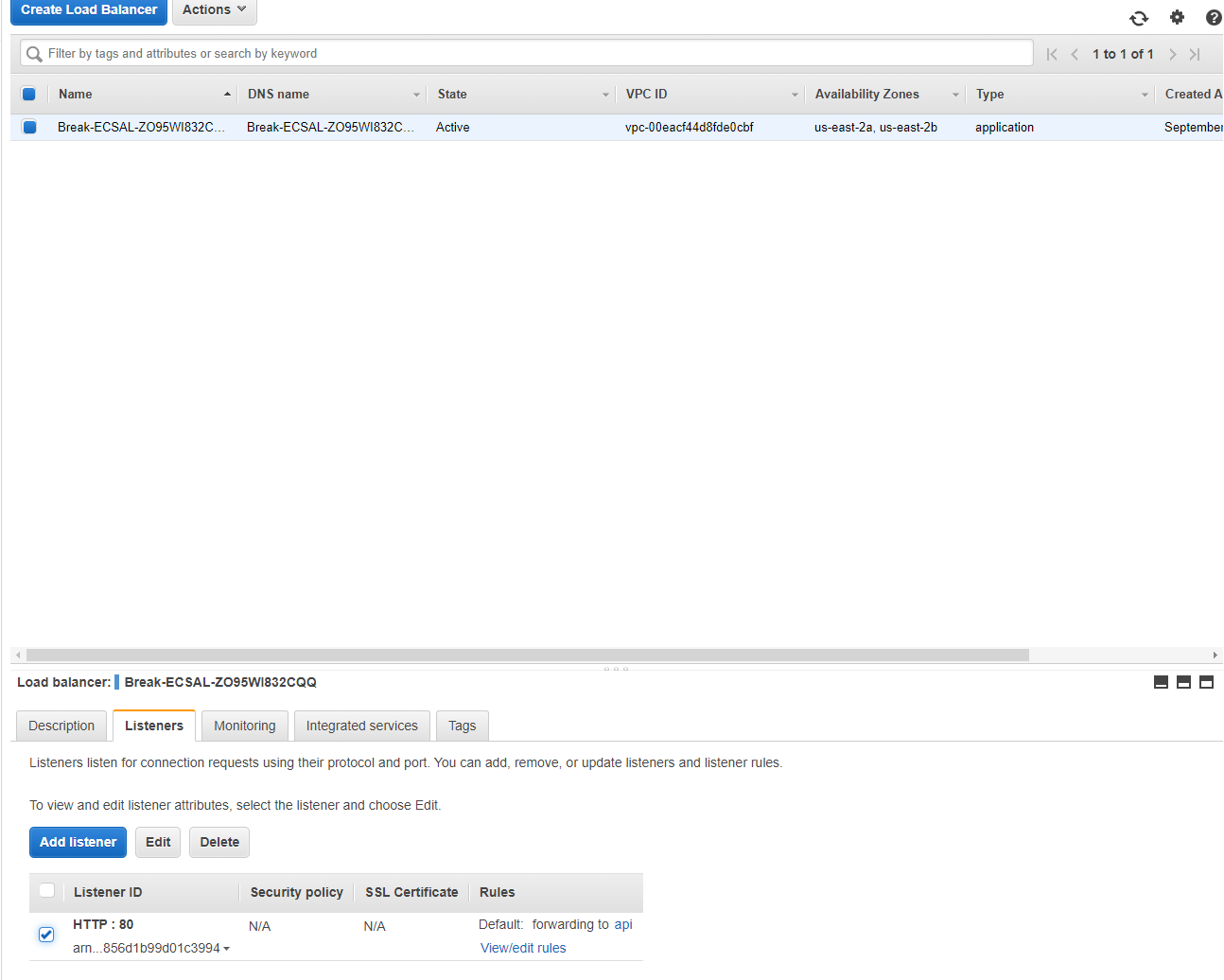
* + 1. Confirmar



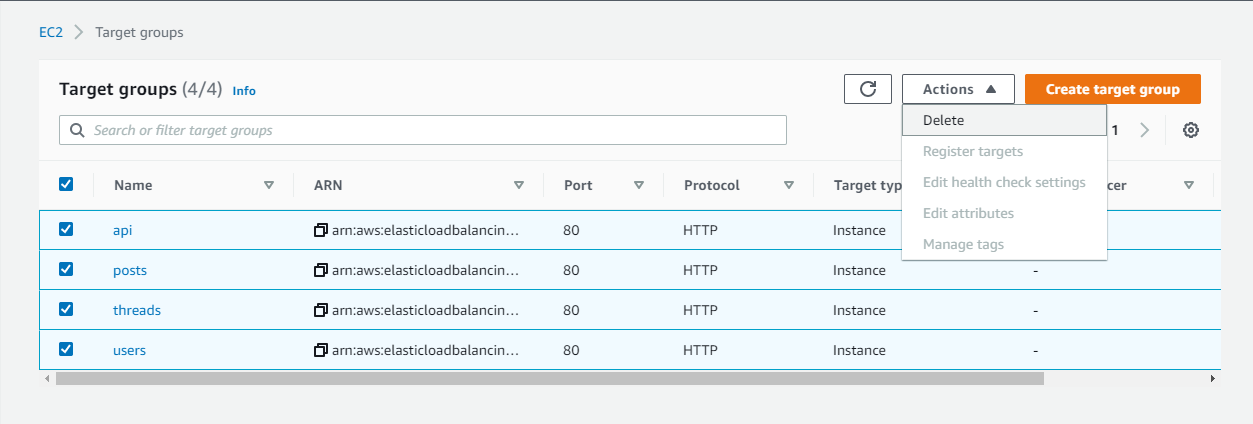
Pasar a la pestaña tasks, seleccionar el boton pararlas todas



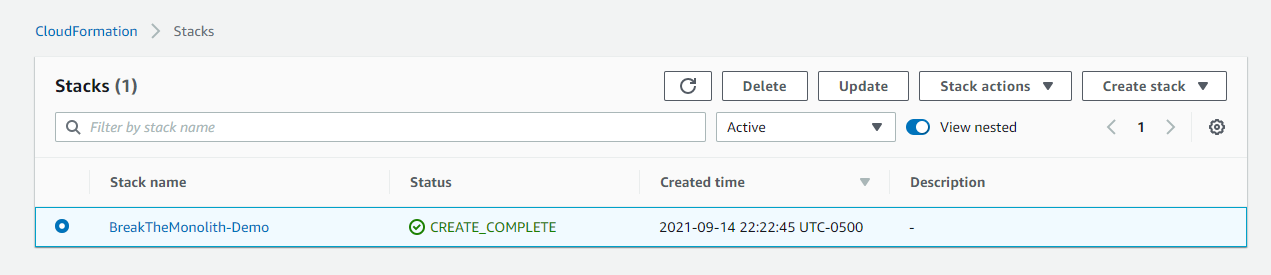
1. Eliminar los listeners



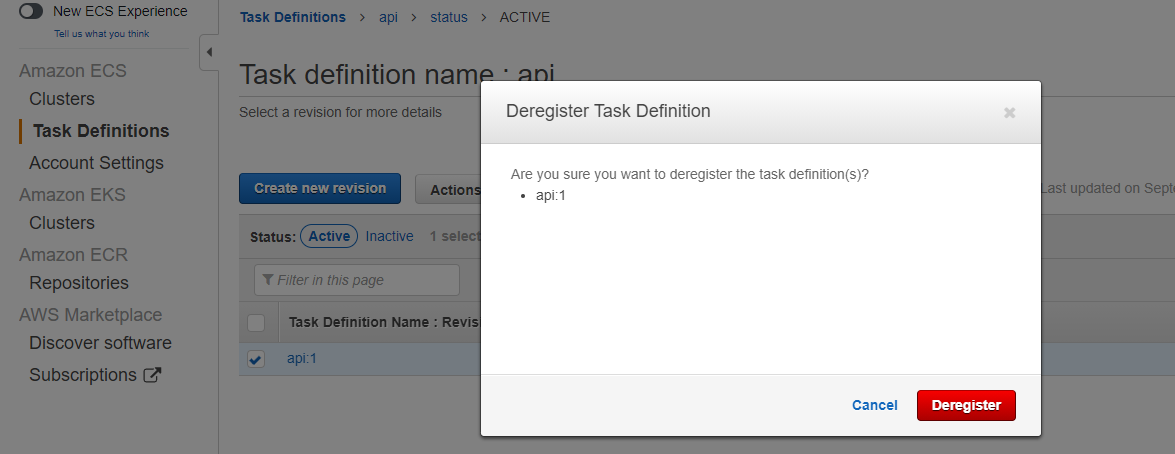
1. Eliminar los target groups



1. Eliminar el stack de AWS Cloudformation



1. Desregistrar las definiciones de tareas



1. Elliminar los repositorios ECR

