

## 41.AWS-B30-Docker-ECR-ECS

--- in this, we will discuss about below sessions.

### Containers:

- Virtual Machines vs Containers
- Installing Docker & Running Container
- Creating Docker Image and Pushing it to ECR.
- Overview on AWS Elastic Container Services(ECS)
- Creating Tasks & Services
- Performing Rolling Update on the application.

## Container registry

--- the famous docker registries are

1. Hub.docker.com
2. AWS elastic container registry (ECR)
3. Azure container registry (ACR)

--- NOTE – in container registry, we have Repository, in Repository we have images.

## AWS ECR (Elastic container registry)

--- go to aws cloud and search for ECR.

The screenshot shows the AWS ECR landing page. At the top, there's a navigation bar with 'aws' and various service icons like EC2, IAM, Route 53, VPC, S3, Certificate Manager, and 'Elastic Container Registry'. Below the navigation, there's a sidebar with 'Containers' and a main content area. The main content features the heading 'Amazon Elastic Container Registry' and the sub-headline 'Share and deploy container software, publicly or privately'. It includes a 'Create a repository' button and a 'Get Started' button. To the right, there's a 'Pricing (US)' section stating 'You pay only for the amount of data you store in your public or private repositories and data transferred to the Internet.' with a 'Learn more' link. Below that is a 'Getting started' section with a 'What is Amazon ECR?' link. At the bottom left, there's a diagram titled 'How it works' showing a flow from 'Write code' to 'Amazon ECR', then 'Compress, encrypt, and control access to Images', followed by 'Version, tag, and manage Images', and finally 'Amazon ECS, Amazon EKS, Amazon Cloud, On-premises' where users can 'Pull Images and run containers anywhere'.

--- Click on get started.

**Create repository**

**General settings**

**Visibility settings** Info  
Choose the visibility setting for the repository.

**Private**  
Access is managed by IAM and repository policy permissions.

**Public**  
Publicly visible and accessible for image pulls.

**Repository name**  
Provide a concise name. A developer should be able to identify the repository contents by the name.

743580576244.dkr.ecr.us-east-1.amazonaws.com/**docker-testing-repository**

25 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

**Tag immutability** Info  
Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.

**Disabled**

Once a repository is created, the visibility setting of the repository can't be changed.

**Image scan settings**

**Deprecation warning**  
ScanOnPush configuration at the repository level is deprecated in favor of registry level scan filters.

**Scan on push**  
Enable scan on push to have each image automatically scanned after being pushed to a repository. If disabled, each image scan must be manually started to get scan results.

**Disabled**

**Encryption settings**

**KMS encryption**  
You can use AWS Key Management Service (KMS) to encrypt images stored in this repository, instead of using the default encryption settings.

**Disabled**

The KMS encryption settings cannot be changed or disabled after the repository is created.

**Create repository**

--- click on create repository.

Successfully created repository docker-testing-repository

Amazon ECR > Repositories

**Private** **Public**

**Private repositories (1)**

Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type	Pull through cache
docker-testing-repository	743580576244.dkr.ecr.us-east-1.amazonaws.com/docker-testing-repository	August 08, 2022, 10:52:04 (UTC+0:5)	Disabled	Continuous	AES-256	Inactive

--- our private repository got created.

## Upload An image to ECR

--- Reference - <https://github.com/prabhu9652/dockertest1.git>

# Build image

```

--- docker build -t prabhu9652/customnginx1:v1 .

# List images

--- docker images

# Create a container from images

--- docker run --rm -d --name nginx01 --publish 9000:80 prabhu9652/customnginx1:v1

# List containers

--- docker ps

```

### Aws cli installation

#### # Unzip installation

```
--- apt update && apt install unzip -y
```

#### # aws cli installation

```
--- curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

```
--- unzip awscliv2.zip
```

```
--- ./aws/install
```

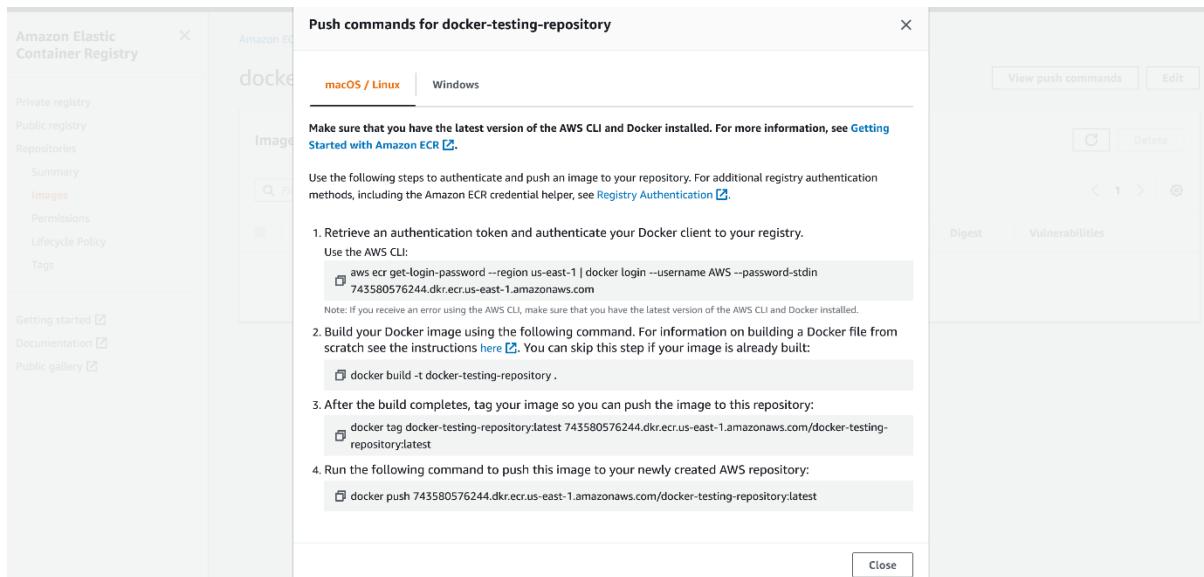
# Configure AWS credentials.

```
--- aws configure
```

--- **Note** – create IAM Role with necessary permissions and attach that role to instance.

## Pushing image to AMAZON ECR

--- **NOTE** – if you want to push an image to ECR then I need to follow proper tagging.



--- these are the commands; we will use IT to push docker image to AWS

--- NOTE – we need to tag the docker images like above.

### # Tag a docker image.

```
--- docker tag prabhu9652/customnginx1:v1 743580576244.dkr.ecr.us-east-1.amazonaws.com/docker-testing-repository:v1
```

### # Login on instance

```
--- aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 743580576244.dkr.ecr.us-east-1.amazonaws.com
```

### # Push image to AMAZON ECR

```
--- docker push 743580576244.dkr.ecr.us-east-1.amazonaws.com/docker-testing-repository:v1
```

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Vulnerabilities
v1	Image	August 08, 2022, 11:50:04 (UTC+05:5)	56.73	<a href="#">Copy URI</a>	sha256:79a1c53cb94caeb...	<a href="#">See findings</a>

--- NOTE – image version V1 is uploaded to amazon ECR.

--- Scenario – I have created a docker image from a project and hosted in an instance. Now I want to make some changes to some html file. This will tell you how I can create an image with new changes and host a docker container form that.

--- here I update html page with new changes.

### # Build a docker images with new changes.

```
--- docker build -t 743580576244.dkr.ecr.us-east-1.amazonaws.com/docker-testing-repository:v2 .
```

### # Create a container form a new image.

```
--- docker run --rm -d --name nginx02 --publish 8100:80 743580576244.dkr.ecr.us-east-1.amazonaws.com/docker-testing-repository:v2
```

# Push image to Amazon ECR.

```
--- docker push 743580576244.dkr.ecr.us-east-1.amazonaws.com/docker-testing-repository:v2
```

## AWS ECS (Elastic container service)

--- How to run a containers in AWS...?

1. Using EC2 instances, install docker/k8s and run the container.
2. AWS elastic container service(ECS)

### 3. AWS container as a service – Fargate

### 4. AWS elastic kubernetes service

--- go to aws ECS.

Amazon Elastic Container Service (ECS)

Amazon ECS makes it easy to deploy, manage, and scale Docker containers running applications, services, and batch processes. Amazon ECS places containers across your cluster based on your resource needs and is integrated with familiar features like Elastic Load Balancing, EC2 security groups, EBS volumes and IAM roles.

Get started

Learn more about Amazon ECS

--- Scenario – 1<sup>st</sup> I will create 2 node cluster later, I will increase the cluster size to 3 node cluster.

Clusters

No clusters found

Get Started

--- click on create cluster.

Create Cluster

**Step 1: Select cluster template**

Step 2: Configure cluster

**Select cluster template**

The following cluster templates are available to simplify cluster creation. Additional configuration and integrations can be added later.

**Networking only** For use with either AWS Fargate (Windows/Linux) or with External instance capacity.

**EC2 Linux + Networking**

**EC2 Windows + Networking**

--- **note** – we can use fargate profiles and we can use amazon linux or windows for cluster creating.  
Here I am choosing amazon linux for cluster creating.

The screenshot shows the AWS Step 2: Configure cluster wizard. The top navigation bar includes links for EC2, IAM, Route 53, VPC, Certificate Manager, Elastic Container Registry, and Elastic Container Service. The search bar is set to "Search for services, features, blogs, docs, and more". The region is set to N. Virginia, and the user is Dectran.

**Configure cluster**

**Step 1: Select cluster template** (highlighted in blue)

**Step 2: Configure cluster**

**Cluster name\***: Test-ECS-CLUSTER

Create an empty cluster

**Instance configuration**

**Provisioning Model**:  On-Demand Instance  
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.  
[Learn more](#)

**EC2 instance type\***: t3 medium

Manually enter desired instance type

**Number of instances\***: 2

**EC2 AMI ID\***: Amazon Linux 2 AMI [ami-040d...]

**Root EBS Volume Size (GiB)**: 30

**Key pair**: terraform-key

You will not be able to SSH into your EC2 instances without a key pair. You can create a new key pair in the [EC2 console](#).

**Networking**

Configure the VPC for your container instances to use. A VPC is an isolated portion of the AWS cloud populated by AWS objects, such as Amazon EC2 instances. You can choose an existing VPC, or create a new one with this wizard.

**VPC**: [vpc-0426817f5356554f3...](#)

Check the structure for [vpc-0426817f5356554f3](#) in the Amazon EC2 console.

**Subnets**

subnet-054e48d6a130ee8 48 (10.0.2.0/24)   testing-vpc- subnet2 - us-east-1b assign ipv6 on creation: Disabled	
subnet-07e912bccea8d8e0 08 (10.0.3.0/24)   testing-vpc- subnet3 - us-east-1c assign ipv6 on creation: Disabled	

subnet-03e7f74857cf1fa2d  
(10.0.1.0/24) | testing-vpc-  
subnet1 - us-east-1a  
assign IPv6 on creation: Di-  
abled

Select a subnet...

Auto assign public IP: Enabled

Security group: Create a new security gr...

Security group inbound rules: CIDR block: 0.0.0.0/0  
Port range: 80 Protocol: tcp

**Container instance IAM role**

The Amazon ECS container agent makes calls to the Amazon ECS API actions on your behalf, so container instances that run the agent require the `ecsInstanceRole` IAM policy and role for the service to know that the agent belongs to you. If you do not have the `ecsInstanceRole` already, we can create one for you.

Container instance IAM role: admin\_access

For container instances to receive the new ARN and resource ID format, the root user needs to opt in for the container instance IAM role. Opt in and try again.

**Tags**

Key	Value
Add key	Add value

**CloudWatch Container Insights**

CloudWatch Container Insights is a monitoring and troubleshooting solution for containerized applications and microservices. It collects, aggregates, and summarizes compute utilization such as CPU, memory, disk, and network; and diagnostic information such as container restart failures to help you isolate issues with your clusters and resolve them quickly. [Learn more](#)

CloudWatch Container Insights:  Enable Container Insights

\*Required      Cancel      Previous      **Create**

--- Click on create.

aws Services Search for services, features, blogs, docs, and more [Alt+S]

EC2 IAM Route 53 VPC S3 Certificate Manager Elastic Container Registry Elastic Container Service

**Launch status**

Your container instances are launching, and it may take a few minutes until they are in the running state and ready to access. Usage hours on your new container instances start immediately and continue to accrue until you stop or terminate them.

Back **View Cluster**

ECS status - 3 of 3 complete **Test-ECS-CLUSTER**

- ECS cluster**  
ECS Cluster Test-ECS-CLUSTER successfully created
- ECS Instance IAM Policy**  
IAM Policy for the role `ec2_admin` successfully attached
- CloudFormation Stack**  
CloudFormation stack [EC2ContainerService-Test-ECS-CLUSTER](#) and its resources successfully created

**Cluster Resources**

Instance type	t3.medium
Desired number of instances	2
Key pair	terraform-key
ECS AMI ID	ami-040d909ea4e568f3
VPC	vpc-042681753566543
Subnets	subnet-054e48d6a130ee848, subnet-07e912bcea8d8e008, subnet-03e7f74857cf1fa2d
VPC Availability Zones	us-east-1a, us-east-1b, us-east-1c, us-east-1d, us-east-1e, us-east-1f
Security group	sg-08507dfb6fed29d
Launch configuration	EC2ContainerService-Test-ECS-CLUSTER-EcsInstance1.c-SM2mtm12d7vk
Auto Scaling group	EC2ContainerService-Test-ECS-CLUSTER-EcsInstanceAsg-PPCO7EB2M1W1

--- it will deploy a cloudformation template and create cluster.

**Create application load balancer**

## Target group creating

--- I will create target with out targets.

The screenshot shows the 'Specify group details' step of the AWS Lambda Target Group creation wizard. It includes sections for 'Basic configuration', 'Choose a target type' (with 'Instances' selected), 'Target group name' (set to 'ECS-TG'), 'Protocol' (set to 'HTTP' on port 80), 'VPC' (selecting 'testing-vpc'), 'Protocol version' (selecting 'HTTP1'), 'Health checks' (protocol set to 'HTTP', path '/'), and optional sections for 'Advanced health check settings' and 'Tags - optional'. A 'Next' button is at the bottom right.

Step 1  
Specify group details

Your load balancer routes requests to the targets in a target group and performs health checks on the targets.

**Basic configuration**  
Settings in this section cannot be changed after the target group is created.

**Choose a target type**

Instances

- Supports load balancing to instances within a specific VPC.
- Facilitates the use of [Amazon EC2 Auto Scaling](#) to manage and scale your EC2 capacity.

IP addresses

- Supports load balancing to VPC and on-premises resources.
- Facilitates routing to multiple IP addresses and network interfaces on the same instance.
- Offers flexibility with microservice based architectures, simplifying inter-application communication.
- Supports IPv6 targets, enabling end-to-end IPv6 communication, and IPv4-to-IPv6 NAT.

Lambda function

- Facilitates routing to a single Lambda function.
- Accessible to Application Load Balancers only.

Application Load Balancer

- Offers the flexibility for a Network Load Balancer to accept and route TCP requests within a specific VPC.
- Facilitates using static IP addresses and PrivateLink with an Application Load Balancer.

**Target group name**  
ECS-TG

A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

**Protocol** Port  
HTTP : 80

**VPC**  
Select the VPC with the instances that you want to include in the target group.  
testing-vpc  
vpc-0426817f5356554f3  
IPv4: 10.0.0.0/16

**Protocol version**

**HTTP1**  
Send requests to targets using HTTP/1.1. Supported when the request protocol is HTTP/1.1 or HTTP/2.

**HTTP2**  
Send requests to targets using HTTP/2. Supported when the request protocol is HTTP/2 or gRPC, but gRPC-specific features are not available.

**gRPC**  
Send requests to targets using gRPC. Supported when the request protocol is gRPC.

**Health checks**  
The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.

**Health check protocol**  
HTTP

**Health check path**  
Use the default path of "/" to ping the root, or specify a custom path if preferred.  
/

**Advanced health check settings**

**Tags - optional**  
Consider adding tags to your target group. Tags enable you to categorize your AWS resources so you can more easily manage them.

Cancel **Next**

--- click on next.

Step 1  
Specify group details

Step 2  
Register targets

**Available instances (2)**

Instance ID	Name	State	Security groups	Zone	Subnet ID
i-0bd2cbf38b3147bee	ECS Instance - EC2ContainerService-e-Test-ECS-CLUSTER	running	EC2ContainerService-Test-ECS-CLUSTER-EcsSecurityGroup-E4E60BD5G888	us-east-1b	subnet-054e48d6a130ee848
i-017342947945be532	ECS Instance - EC2ContainerService-e-Test-ECS-CLUSTER	running	EC2ContainerService-Test-ECS-CLUSTER-EcsSecurityGroup-E4E60BD5G888	us-east-1a	subnet-03e7f74857cf1fa2d

**0 selected**

Ports for the selected instances  
Ports for routing traffic to the selected instances.  
80  
1-65535 (separate multiple ports with commas)

Include as pending below

**Review targets**

**Targets (0)**

All	Name	Port	State	Security groups	Zone	Subnet ID
-----	------	------	-------	-----------------	------	-----------

No instances added yet  
Specify instances above, or leave the group empty if you prefer to add targets later.

0 pending

Create target group

--- I will not add any targets here. Click on create target group.

### Application load balancer creating

--- go to aws ec2 instance section.

New EC2 Experience

EC2 Dashboard

EC2 Global View

Events

Tags

Limits

Instances

Instances New

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances New

Dedicated Hosts

Scheduled Instances

Capacity Reservations

Images

AMIs New

AMI Catalog

Elastic Block Store

Volumes New

Snapshots New

Lifecycle Manager New

**Create Load Balancer**

Actions

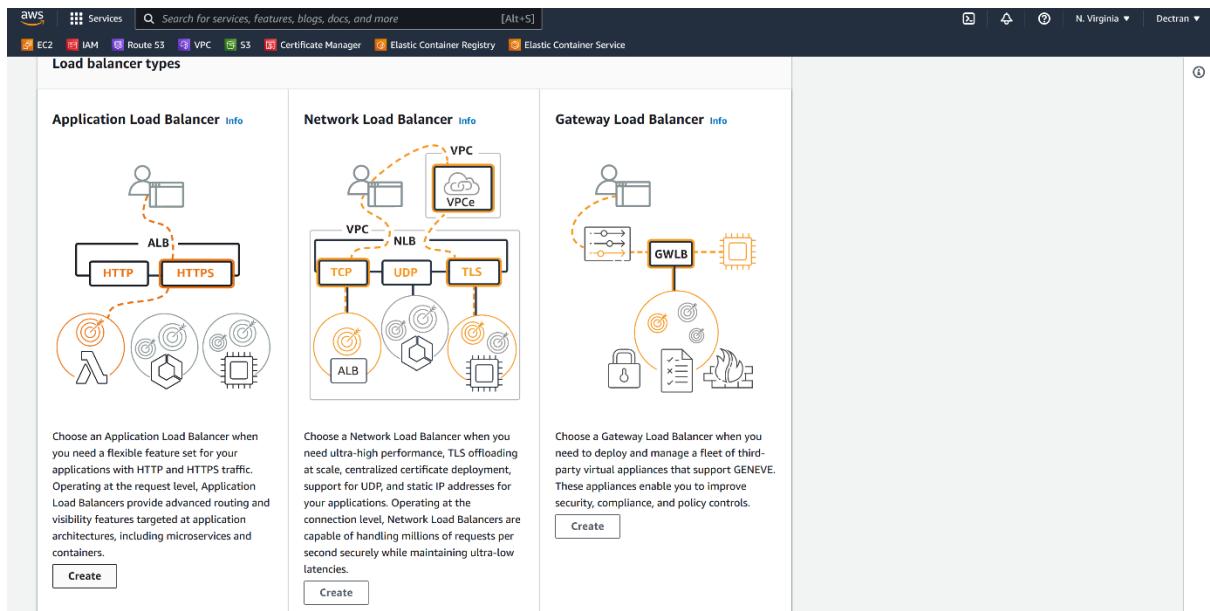
Filter by tags and attributes or search by keyword

Name	DNS name	State	VPC ID	Availability Zones	Type	Created At
------	----------	-------	--------	--------------------	------	------------

You do not have any load balancers in this region.

Select a load balancer

--- click on create load balancer.



--- click on create application load balancer.

The screenshot shows the 'Create Application Load Balancer' wizard in progress:

- Basic configuration** step:
  - Load balancer name:** AWS-ECS-ALB
  - Scheme:** Internet-facing (selected)
  - IP address type:** IPv4 (selected)
- Network mapping** step:
  - VPC:** testing-vpc (selected)
  - Mappings:** us-east-1a (selected)
  - Subnet:** subnet-03e7f74857cf1fa2d
  - IPv4 settings:** Assigned by AWS

**Subnets**

- us-east-1b**
  - Subnet: subnet-054e48d6a130ee848 testing-vpc-subnet2
  - IPv4 settings: Assigned by AWS
- us-east-1c**
  - Subnet: subnet-07e912bcea8d8e008 testing-vpc-subnet3
  - IPv4 settings: Assigned by AWS

**Security groups** Info

A security group is a set of firewall rules that control the traffic to your load balancer.

Select up to 5 security groups

Create new security group

EC2ContainerService-Test-ECS-CLUSTER-EcsSecurityGroup-E4E60BD5G8B8 sg-08507dfb66ffed29d

VPC: vpc-0426817f5356554f3

**Listeners and routing** Info

A listener is a process that checks for connection requests, using the protocol and port you configure. Traffic received by the listener is then routed per your specification. You can specify multiple rules and multiple certificates per listener after the load balancer is created.

▼ Listener HTTP:80

Protocol	Port	Default action	Info
HTTP	: 80 1-65535	Forward to <b>ECS-TG</b> Target type: Instance, IPv4	HTTP <input type="button" value="C"/> <a href="#">Create target group</a>

Add listener

▼ Add-on services - optional

Additional AWS services can be integrated with this load balancer at launch. You can also add these and other services after your load balancer is created by reviewing the "Integrated Services" tab for the selected load balancer.

**AWS Global Accelerator** Info

Create an accelerator to get static IP addresses and improve the performance and availability of your applications. [Additional charges apply](#)

**Tags - optional**

Consider adding tags to your load balancer. Tags enable you to categorize your AWS resources so you can more easily manage them. The 'Key' is required, but 'Value' is optional. For example, you can have Key = production-webserver, or Key = webserver, and Value = production.

**Summary**

Review and confirm your configurations. [Estimate cost](#)

<b>Basic configuration</b> <small>Edit</small>	<b>Security groups</b> <small>Edit</small>	<b>Network mapping</b> <small>Edit</small>	<b>Listeners and routing</b> <small>Edit</small>
AWS-ECS-ALB • Internet-facing • IPv4	• EC2ContainerService-Test-ECS-CLUSTER-EcsSecurityGroup-E4E60BD5G8B8 sg-08507dfb66ffed29d	VPC vpc-0426817f5356554f3 <input type="button" value="C"/> testing-vpc • us-east-1a subnet-01e7f74857cf1fa2d <input type="button" value="C"/> testing-vpc-subnet1 • us-east-1b subnet-054e48d6a130ee848 <input type="button" value="C"/> testing-vpc-subnet2 • us-east-1c subnet-07e912bcea8d8e008 <input type="button" value="C"/> testing-vpc-subnet3	• HTTP:80 defaults to <b>ECS-TG</b>
<b>Add-on services</b> <small>Edit</small>	<b>Tags</b> <small>Edit</small>		
None	None		

**Attributes**

(i) Certain default attributes will be applied to your load balancer. You can view and edit them after creating the load balancer.

--- click on create load balancer.

## Increase ECS 2 nodes cluster to 3 node cluster

--- go to autoscaling group.

The screenshot shows the AWS Management Console with the search bar at the top. The left sidebar is expanded, showing categories like Images, Elastic Block Store, Network & Security, Load Balancing, and Auto Scaling. Under Auto Scaling, 'Auto Scaling Groups' is selected. The main content area displays a table titled 'Auto Scaling groups (1) Info'. The table has columns for Name, Launch template/configuration, Instances, Status, Desired capacity, Min, Max, and Availability Zones. One row is present, showing 'EC2ContainerServ' with 'EC2ContainerService-Test-ECS-CLU...' as the launch template, 2 instances, and a desired capacity of 2. The availability zones listed are 'us-east-1a, us-east-1b'.

--- click on auto scaling groups.

This screenshot shows the same AWS Management Console interface, but the 'Edit' button in the top right corner of the 'Auto Scaling groups (1/1) Info' table is highlighted with a red box. The rest of the interface is identical to the previous screenshot, showing the single Auto Scaling group and its details.

--- select autoscaling group and click on edit option.

--- increase the desired capacity and maximum capacity to 3.

--- click on update.

--- note – now it is updating the capacity.

## How to update our cluster security group

--- go to ECS cluster autoscaling group.

The screenshot shows the AWS EC2 Auto Scaling Groups page. On the left, there's a navigation sidebar with various services like EC2, IAM, VPC, S3, Certificate Manager, and Elastic Container Registry. Under the 'Auto Scaling' section, 'Auto Scaling Groups' is selected. The main content area displays a table titled 'Auto Scaling groups (1)'. The table has columns for Name, Launch template/configuration, Instances, Status, Desired capacity, Min, Max, and Availability Zones. One row is shown for 'EC2ContainerServ' with a status of '2', desired capacity of '2', min of '0', max of '2', and availability zones 'us-east-1a, us-east-1b'.

--- select that autoscaling group and under details section, you will find the security group select it and update it.

The screenshot shows the same EC2 Auto Scaling Groups page after selecting the 'EC2ContainerServ' group. A green banner at the top says 'Auto Scaling group updated successfully'. Below it, the 'Launch configuration' section is expanded. It shows the launch configuration name 'EC2ContainerService-Test-ECS-CLUSTER-EcsInstanceAsg-PPC07EB2M1W1', AMI ID 'ami-040d909ea4e56f8f3', instance type 't3.medium', storage '/dev/xvda', key pair 'terraform-key', and security group 'sg-08507dfb66ffed29d'. The 'Edit' button is visible in the top right corner of this section.

--- click on security group.

--- update the security group with required information. Here I am testing so, I allowed traffic from anywhere.

## AWS ECS task definition

--- Reference - <https://github.com/aws-samples/aws-containers-task-definitions/tree/master/nginx>

--- go to ECS.

Cluster : Test-ECS-CLUSTER

Get a detailed view of the resources on your cluster.

Cluster ARN: arn:aws:ecs:us-east-1:743580576244:cluster/Test-ECS-CLUSTER

Status: ACTIVE

Registered container instances: 3

	Container Instance	ECS Instance	Availability Zone	External Insta...	Agent Connec...	Status	Running tasks...	CPU available	Memory availa...
<input type="checkbox"/>	d411e53f52d74bbb6abe82b...	i-0bd2cbf38b31...	us-east-1b	false	true	ACTIVE	0	2048	3883

--- click on ECS task definition.

--- scenario – when you want to deploy container on docker host, we will give port, image, tag details in a command right. Something like that, when we want to create container ECS cluster we will mention all those details in Task Definition like how much cpu, memory, you want to consume.

--- note - aws ECS nginx task definition github – search by this name on aws console.

(<https://github.com/aws-samples/aws-containers-task-definitions/tree/master/nginx>)

--- nginx\_ec2.json

```
{
  "requiresCompatibilities": [
    "EC2"
  ],
  "containerDefinitions": [
    {
      "name": "nginx", # if changes this name then please change below.
      "image": "743580576244.dkr.ecr.us-east-1.amazonaws.com/docker-testing-
repository:v2",
      "memory": 256,
      "cpu": 256,
      "essential": true,
      "portMappings": [
        {
          "containerPort": 80,
          "protocol": "tcp"
        }
      ],
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "/aws/container-logs/test-nginx",
          "awslogs-region": "us-east-1",
          "awslogs-stream-prefix": "nginx"
        }
      }
    }
  ]
}
```

```

        "awslogs-group": "awslogs-nginx-ecs", # create a cloudwatch log
group by this name.
        "awslogs-region": "us-east-1",
        "awslogs-stream-prefix": "nginx" # change here
    }
}
],
"volumes": [],
"networkMode": "bridge",
"placementConstraints": [],
"family": "nginx" # change here.
}

```

### Cloud watch log group creating

--- go to the aws cloudwatch

The screenshot shows the AWS CloudWatch interface with the 'Logs' section selected. Under 'Log groups', it says 'Log groups (0)' and 'By default, we only load up to 10000 log groups.' There is a search bar labeled 'Filter log groups or try prefix search' and a checkbox for 'Exact match'. A large button at the bottom right says 'Create log group'.

--- click on log groups and create log group.

The screenshot shows the AWS CloudWatch interface with the 'Logs' section selected. Under 'Log groups', it says 'Log groups (1)' and 'By default, we only load up to 10000 log groups.' There is a search bar labeled 'Filter log groups or try prefix search' and a checkbox for 'Exact match'. A table lists the single log group: 'awslogs-nginx-ecs' with 'Never expire' retention. A large button at the bottom right says 'Create log group'.

--- log groups got created.

--- please click on task definition.

Screenshot of the AWS CloudWatch Task Definitions page. The left sidebar shows navigation options like 'Amazon ECS Clusters' and 'Task Definitions'. The main content area displays a table with one row labeled 'Task Definition' and 'Latest revision status'. A note at the bottom says 'No results'.

--- click on create new task definition.

Screenshot of the 'Create new Task Definition' wizard. Step 1: Select launch type compatibility. It shows three options: FARGATE, EC2, and EXTERNAL. FARGATE is highlighted. Step 2: Configure task and container definitions is shown below.

Launch Type	Description
FARGATE	Price based on task size Requires network mode awsvpc AWS-managed infrastructure, no Amazon EC2 instances to manage
EC2	Price based on resource usage Multiple network modes available Self-managed infrastructure using Amazon EC2 instances
EXTERNAL	Price based on instance-hours and additional charges for other AWS services used Self-managed on-premise infrastructure with ECS Anywhere

--- please select EC2 and click on next.

--- scroll down and click on configure via json.

Screenshot of the 'Configure via JSON' dialog box. It contains a large JSON configuration block with several red highlights underlining specific fields such as 'requiresCompatibilities', 'containerDefinitions', 'name', 'image', 'memory', 'cpu', 'essential', 'portMappings', 'containerPort', 'protocol', 'logConfiguration', 'logDriver', 'options', 'awslogs-group', 'awslogs-region', 'awslogs-stream-prefix', 'volumes', 'networkMode', 'placementConstraints', and 'family'. The 'Cancel' and 'Save' buttons are visible at the bottom right.

```
{
  "requiresCompatibilities": [
    "EC2"
  ],
  "containerDefinitions": [
    {
      "name": "nginx",
      "image": "743560576244.dkr.ecr.us-east-1.amazonaws.com/docker-testing-repository:v2",
      "memory": 256,
      "cpu": 256,
      "essential": true,
      "portMappings": [
        {
          "containerPort": 80,
          "protocol": "tcp"
        }
      ],
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "awslogs-nginx-ecs",
          "awslogs-region": "us-east-1",
          "awslogs-stream-prefix": "nginx"
        }
      }
    }
  ],
  "volumes": [],
  "networkMode": "bridge",
  "placementConstraints": [],
  "family": "nginx"
}
```

--- click on save.

--- everything will be autoloaded from the json file, check before clicking on create.

The screenshot shows the AWS ECS Task Definitions page. A success message 'Created Task Definition successfully' is displayed. The task definition name is 'nginx:1'. Configuration options include 'Task role: None', 'Network mode: Bridge', and 'Compatibilities: EXTERNAL, EC2'. The JSON tab is selected.

--- note – now the task nginx:1 got created.

## Deploy a container using task definition

--- note – here I will deploy our container using task definition.

The screenshot shows the AWS ECS Clusters page. It displays one cluster named 'Test-ECS-CLUSTER'. The cluster details table includes columns for Cluster name, CloudWatch monitoring, Services, Running tasks, Pending tasks, and Container instances. The status for Container instances is 3.

Cluster name	CloudWatch monitoring	Services	Running tasks	Pending tasks	Container instances
Test-ECS-CLUSTER	Container Insights	0	0	0	3

--- click on our cluster.

Cluster : Test-ECS-CLUSTER

Cluster ARN: arn:aws:ecs:us-east-1:743580576244:cluster/Test-ECS-CLUSTER

Status: ACTIVE

Registered container instances: 3

	Task	Task definition	Container ins...	Last status	Desired status	Started at	Started By	Group	Launch type	Platform vers...

No results

--- click on Run New task.

--- Number of tasks = 3 (it is nothing but no of containers you want to deploy).

Run Task

Select the cluster to run your task definition on and the number of copies of that task to run. To apply container overrides or target particular container instances, click Advanced Options.

Launch type: FARGATE (radio button)

Task Definition: Family: nginx (dropdown)

Number of tasks: 3

Task Group: nginx-task-group

VPC and security groups

Task Placement

Placement Templates: AZ Balanced Spread

Advanced Options

Task tagging configuration

Enable ECS managed tags: checked

Propagate tags from: Do not propagate

Tags

Key	Value
Add key	Add value

Cancel Run Task

--- click on Run Task.

## Application testing

--- cluster -> Run a task.

The screenshot shows the AWS CloudWatch Metrics console with the title "Application testing" in red at the top. Below it, there is a message: "Get a detailed view of the resources on your cluster." The main content area displays metrics for a cluster named "Test-ECS-CLUSTER". Key statistics shown include:

- Cluster ARN: arn:aws:ecs:us-east-1:743580576244:cluster/Test-ECS-CLUSTER
- Status: ACTIVE
- Registered container instances: 3
  - Pending tasks count: 0 Fargate, 3 EC2, 0 External
  - Running tasks count: 0 Fargate, 0 EC2, 0 External
  - Active service count: 0 Fargate, 0 EC2, 0 External
  - Draining service count: 0 Fargate, 0 EC2, 0 External

Below this, there is a table with tabs for Services, Tasks, ECS Instances, Metrics, Scheduled Tasks, Tags, and Capacity Providers. The "Tasks" tab is selected, showing a button labeled "Run new Task". The table lists three tasks with the following details:

Task	Task definition	Container ins...	Last status	Desired status	Started at	Started By	Group	Launch type	Platform vers...
28667e3d323e...	nginx:1	d411e53f52d7...	PENDING	RUNNING			nginx-task-group	EC2	--
b649dc759d54...	nginx:1	d5355ab86ff14...	PENDING	RUNNING			nginx-task-group	EC2	--
f3639a4bc845...	nginx:1	f0b33a2ceac0...	PENDING	RUNNING			nginx-task-group	EC2	--

At the bottom right of the table, it says "Last updated on August 8, 2022 3:54:19 PM (0m ago)".

--- under task, click on any task id.

The screenshot shows the AWS CloudWatch Metrics console with the title "Task : 6f856b3d5d614b7ea9cd6fe574a090b". The left sidebar shows the "Clusters" section under "Amazon ECS". The main content area displays the details of the specified task, including:

Cluster	Test-ECS-CLUSTER
Container instance	f0b33a2ceac049e2952fe84fb914fb18
EC2 Instance id	i-012c5d82e7d427218
Launch type	EC2
Task definition	nginx:2
Group	test-ecs-group
Task role	None
Last status	RUNNING
Desired status	RUNNING
Created at	2022-08-08 16:09:08 +0530
Started at	2022-08-08 16:09:11 +0530

Below the table, there is a "Network" section with a "Network mode" set to "bridge". There is also a "Containers" section.

Name	Container Runtime ID	Status	Image	Image Digest	CPU Units	Hard/Soft...	Essential	Resource...
nginx	1d0b433ca4aada9c0f...	RUNNING	743580576244.dkr.ecr.us-east-1.a...	sha256:0c31758c87a380d7e394eb...	256	256/-	true	4cfe460e...

Details

Network bindings

Host Port	Container Port	Protocol	External Link
49154	80	tcp	<a href="http://3.83.69.96:49154">3.83.69.96:49154</a>

Environment Variables - not configured

Environment Files - not configured

Docker labels - not configured

Extra hosts - not configured

Mount Points - not configured

Volumes from - not configured

Ulimits - not configured

Elastic Inference - not configured

Log Configuration

Log driver: awslogs [View logs in CloudWatch](#)

Key	Value
awslogs-group	awslogs-nginx-ecs
awslogs-region	us-east-1

--- click on the external link and you will be able access the application.

## Services

--- ECS -> cluster -> click on our cluster.

--- Scenario - service will recreate the container, due to some reason it stopped working.

Cluster : Test-ECS-CLUSTER

Get a detailed view of the resources on your cluster.

Cluster ARN: arn:aws:ecs:us-east-1:743580576244:cluster/Test-ECS-CLUSTER

Status: ACTIVE

Registered container instances: 3

- Pending tasks count: 0 Fargate, 0 EC2, 0 External
- Running tasks count: 0 Fargate, 3 EC2, 0 External
- Active service count: 0 Fargate, 0 EC2, 0 External
- Draining service count: 0 Fargate, 0 EC2, 0 External

Services    Tasks    ECS Instances    Metrics    Scheduled Tasks    Tags    Capacity Providers

Create    Update    Delete    Actions ▾

Last updated on August 8, 2022 4:19:52 PM (3m ago)

Service Name	Status	Service type	Task Definition	Desired tasks ...	Running task...	Launch type	Platform vers...
No results							

--- click on create.

**Create Service**

**Step 1: Configure service**

Step 2: Configure network  
Step 3: Set Auto Scaling (optional)  
Step 4: Review

### Configure service

A service lets you specify how many copies of your task definition to run and maintain in a cluster. You can optionally use an Elastic Load Balancing load balancer to distribute incoming traffic to containers in your service. Amazon ECS maintains the number of tasks and coordinates task scheduling with the load balancer. You can also optionally use Service Auto Scaling to adjust the number of tasks in your service.

**Launch type**  FARGATE  EC2  EXTERNAL

[Switch to capacity provider strategy](#)

**Task Definition** Family: nginx [Enter a value](#)  
Revision: 2 (latest)

**Cluster**: Test-ECS-CLUSTER

**Service name**: mywebapp-service

**Service type\***:  REPLICA  DAEMON

**Number of tasks**: 4

**Minimum healthy percent**: 100

**Maximum percent**: 200

**Deployment circuit breaker**: Disabled

### Deployments

Choose a deployment option for the service.

**Deployment type\***:  Rolling update [Edit](#)  
 Blue/green deployment (powered by AWS CodeDeploy) [Edit](#)  
 This sets AWS CodeDeploy as the deployment controller for the service. A CodeDeploy application and deployment group are created automatically with [default settings](#) for the service. To change the rolling update deployment type after the service has been created, you must re-create the service and select the "rolling update" deployment type.

### Task Placement

Lets you customize how tasks are placed on instances within your cluster. Different placement strategies are available to optimize for availability and efficiency.

**Placement Templates**: AZ Balanced Spread [Edit](#)  
 This template will spread tasks across availability zones and within the availability zone spread tasks across instances. [Learn more](#)  
 Strategy: spread(attribute:ecs.availability-zone), spread(instanceId)

### Task tagging configuration

Enable ECS managed tags [Edit](#)

Propagate tags from: Do not propagate [Edit](#)

#### Tags

Key	Value
Add key	Add value

\*Required [Cancel](#) [Next step](#)

--- click on next.

## Create Service

- Step 1: Configure service
- Step 2: Configure network**
- Step 3: Set Auto Scaling (optional)
- Step 4: Review

### Configure network

#### VPC and security groups

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

#### Health check grace period

If your service's tasks take a while to start and respond to ELB health checks, you can specify a health check grace period of up to 2,147,483,647 seconds during which the ECS service scheduler will ignore ELB health check status. This grace period can prevent the ECS service scheduler from marking tasks as unhealthy and stopping them before they have time to come up. This is only valid if your service is configured to use a load balancer.

Health check grace period  ⓘ

#### Load balancing

An Elastic Load Balancing load balancer distributes incoming traffic across the tasks running in your service. Choose an existing load balancer, or create a new one in the [Amazon EC2 console](#).

##### Load balancer type\*

None

Your service will not use a load balancer.

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.

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.

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  ⓘ

Load balancer name  ⓘ

#### Container to load balance

Container name : port  ⓘ Add to load balancer

--- click on add to load balancer.

Container to load balance  
nginx : 80 Remove ×

Production listener port\*  ⓘ

Production listener protocol\* HTTP

Target group name  ⓘ

Target group protocol HTTP ⓘ

Target type instance ⓘ

Path pattern  Evaluation order default

Health check path  ⓘ

Additional health check options can be configured in the ELB console after you create your service.

#### App Mesh

To use your service with App Mesh, you must

- Ensure your task definition is configured properly. Edit your task definition if you haven't done this.
- Set up your service to use Service Discovery.

#### Service discovery (optional)

Service discovery uses Amazon Route 53 to create a namespace for your service, which allows it to be discoverable via DNS.

Enable service discovery integration

\*Required

Cancel

Previous

Next step

--- click on next.



Create Service

- Step 1: Configure service
- Step 2: Configure network
- Step 3: Set Auto Scaling (optional)**
- Step 4: Review

#### Set Auto Scaling (optional)

Automatically adjust your service's desired count up and down within a specified range in response to CloudWatch alarms. 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

\*Required

[Cancel](#) [Previous](#) [Next step](#)

--- click on next step.



Create Service

- Step 1: Configure service
- Step 2: Configure network
- Step 3: Set Auto Scaling (optional)**
- Step 4: Review**

#### Review

**Cluster:** Test-ECS-CLUSTER

**Launch type:** EC2

**Task Definition:** nginxx:2

**Service name:** mywebapp-service

**Service type:** REPLICAS

**Number of tasks:** 4

**Minimum healthy percent:** 100

**Maximum percent:** 200

**Deployment circuit breaker:** Disabled

[Edit](#)

#### Configure network

**Container Name:** nginx

**Container Port:** 80

**ELB Name:** AWS-ECS-ALB

**Target Group:** ECS-TG

**Health Check Path:** /

**Listener Port:** 80

**Path-pattern:** /

**Service Role:** AWSServiceRoleForECS

[Edit](#)

#### Set Auto Scaling (optional)

not configured

[Edit](#)

[Cancel](#) [Previous](#) [Create Service](#)

--- click on create a service.

## Check the load balancer target group

--- **note** – previously, when I am creating target group, I did not add any target to the target groups. But now while creating service in ECS, it should add containers as targets in load balancer.

Total targets	Healthy	Unhealthy	Unused	Initial	Draining
4	4	0	0	0	0

Instance ID	Name	Port	Zone	Health status	Health status details
i-017342947945be532	ECS Instance - EC2ContainerService-Test-ECS-CLUSTER	49155	us-east-1a	healthy	
i-0bd2cbf38b5147bee	ECS Instance - EC2ContainerService-Test-ECS-CLUSTER	49155	us-east-1b	healthy	
i-012c5d82e7d427218	ECS Instance - EC2ContainerService-Test-ECS-CLUSTER	49155	us-east-1c	healthy	
i-0bd2cbf38b5147bee	ECS Instance - EC2ContainerService-Test-ECS-CLUSTER	49156	us-east-1b	healthy	

--- the containers added as targets in target groups.

--- go to the load balancer and copy the load balancer dns name and test it on google.

## Rolling update on ECS cluster

### # New image building

--- docker build -t 743580576244.dkr.ecr.us-east-1.amazonaws.com/docker-testing-repository:v3 .

### # Push new image

--- docker push 743580576244.dkr.ecr.us-east-1.amazonaws.com/docker-testing-repository:v3

### New task definition version creating

--- **note** – here I will create a new version of task definition and the new version task definition will be used in service.

Task Definition	Latest revision status
nginx	ACTIVE

--- click on our task definition.

--- edit our task definition json file.

Task Definition: nginx:2

View detailed information for your task definition. To modify the task definition, you need to create a new revision and then make the required changes to the task definition.

Create new revision Actions

Builder JSON Tags

Task definition name: nginx

Task role: None

Optional IAM role that tasks can use to make API requests to authorized AWS services. Create an Amazon Elastic Container Service Task Role in the IAM Console.

Network mode: Bridge

If you choose <default>, ECS will start your container using Docker's default networking mode, which is Bridge on Linux and NAT on Windows. Windows tasks support the <default> and awsvpc network modes.

Compatibilities: EXTERNAL, EC2

Requires compatibilities: EC2

--- click on create a new version.

--- after clicking on **create new version** and scroll down.

--- click on configure via json.

```
{
  "requiresCompatibilities": [
    "EC2"
  ],
  "containerDefinitions": [
    {
      "name": "nginx",
      "image": "743580576244.dkr.ecr.us-east-1.amazonaws.com/docker-testing-
repository:v3",
      "memory": 256,
      "cpu": 256,
      "essential": true,
      "portMappings": [
        {
          "containerPort": 80,
          "protocol": "tcp"
        }
      ],
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
          "awslogs-group": "awslogs-nginx-ecs",
          "awslogs-region": "us-east-1",
          "awslogs-stream-prefix": "nginx"
        }
      }
    }
  ],
}
```

```

    "volumes": [],
    "networkMode": "bridge",
    "placementConstraints": [],
    "family": "nginx"
}

```

--- note – after copying this file, click on create.

Task Definition Name : Revision	Status
nginx:3	Active
nginx:2	Active

--- new task definition version got created.

### Update service with new task definition

--- note – here, I will update the service the new version of task definition.

--- ECS -> cluster -> select service.

Service Name	Status	Service type	Task Definition	Desired tasks ...	Running task...	Launch type	Platform versi...
mywebapp-service	ACTIVE	REPLICAS	nginx:2	4	4	EC2	—

--- select the service.

New ECS Experience  
Tell us what you think

Amazon ECS

- Clusters**
- Task Definitions
- Account Settings
- Amazon EKS
- Clusters
- Amazon ECR
- Repositories
- AWS Marketplace
- Discover software
- Subscriptions

Clusters > Test-ECS-CLUSTER > Service: mywebapp-service

Service : mywebapp-service

Cluster: Test-ECS-CLUSTER

Status: ACTIVE

Task definition: nginx:2

Service type: REPLICA

Launch type: EC2

Service role: AWSServiceRoleForECS

Created By: arn:aws:iam::743580576244:root

Desired count: 4

Pending count: 0

Running count: 4

Update Delete

Details Tasks Events Auto Scaling Deployments Metrics Tags Logs

Load Balancing

Target Group Name	Container Name	Container Port
ECS-TG	nginx	80

Network Access

Health check grace period: 0

--- click on update cluster.

Step 1: Configure service

Step 2: Configure network

Step 3: Set Auto Scaling (optional)

Step 4: Review

Configure service

This service lets you specify how many copies of your task definition to run and maintain in a cluster. You can optionally use an Elastic Load Balancing load balancer to distribute incoming traffic to containers in your service. Amazon ECS maintains that number of tasks and coordinates task scheduling with the load balancer. You can also optionally use Service Auto Scaling to adjust the number of tasks in your service.

Task Definition Family: nginx

Revision: 3 (latest)

Enter a value

Launch type: EC2

Force new deployment:

Switch to capacity provider strategy

Cluster: Test-ECS-CLUSTER

Service name: mywebapp-service

Service type\*: REPLICA

--- select the revision 3(latest) and click on next to complete.

--- go to cluster -> click on task to see rolling deployment.

Pending tasks count: 0 Fargate, 0 EC2, 0 External

Running tasks count: 0 Fargate, 11 EC2, 0 External

Active service count: 0 Fargate, 1 EC2, 0 External

Draining service count: 0 Fargate, 0 EC2, 0 External

Task	Task definition	Container ins...	Last status	Desired status	Started at	Started By	Group	Launch type	Platform vers...
50ca402fcaeb...	nginxx:3	f0b33a2ceac0...	RUNNING	RUNNING	2022-08-08 18:...	ecs-svc/18372...	service:myweb...	EC2	--
6a39ef595d57...	nginxx:3	d411e53f52d7...	RUNNING	RUNNING	2022-08-08 18:...	ecs-svc/18372...	service:myweb...	EC2	--
6f856b3d35d1...	nginxx:2	f0b33a2ceac0...	RUNNING	RUNNING	2022-08-08 16:...		test-ecs-group	EC2	--
7abdf64e4914...	nginxx:2	d5355ab86ff14...	RUNNING	RUNNING	2022-08-08 16:...	ecs-svc/24845...	service:myweb...	EC2	--
97696a1ca805...	nginxx:2	d411e53f52d7...	RUNNING	RUNNING	2022-08-08 16:...	ecs-svc/24845...	service:myweb...	EC2	--
ab04c39da94b...	nginxx:2	d411e53f52d7...	RUNNING	RUNNING	2022-08-08 16:...	ecs-svc/24845...	service:myweb...	EC2	--
b31709141984...	nginxx:2	f0b33a2ceac0...	RUNNING	RUNNING	2022-08-08 16:...	ecs-svc/24845...	service:myweb...	EC2	--
b60794bfac69...	nginxx:2	d5355ab86ff14...	RUNNING	RUNNING	2022-08-08 16:...		test-ecs-group	EC2	--
bd04fa1c32c3...	nginxx:2	d411e53f52d7...	RUNNING	RUNNING	2022-08-08 16:...		test-ecs-group	EC2	--
d00e97c1580b...	nginxx:3	d5355ab86ff14...	RUNNING	RUNNING	2022-08-08 18:...	ecs-svc/18372...	service:myweb...	EC2	--
e42b895adff14...	nginxx:3	d5355ab86ff14...	RUNNING	RUNNING	2022-08-08 18:...	ecs-svc/18372...	service:myweb...	EC2	--

--- I have given 200% in the service while I am creating it. That is why it created 4 new containers. It will delete the old containers once it the new containers are ready.

## Increase replicas using service

--- note – using service, we will increase the containers to 5 on 3 node cluster.

Clusters

An Amazon ECS cluster is a regional grouping of one or more container instances on which you can run task requests. Each account receives a default cluster the first time you use the Amazon ECS service. Clusters may contain more than one Amazon EC2 instance type.

For more information, see the [ECS documentation](#).

Cluster name	CloudWatch monitoring	Services	Running tasks	Pending tasks	Container instances
Test-ECS-CLUSTER	Container Insights	1	7	0	3

--- click on clusters and click on our cluster.

--- click on service.

Service Name	Status	Service type	Task Definition	Desired tasks...	Running tasks...	Launch type	Platform vers...
mywebapp-service	ACTIVE	REPLICAS	nginx:3	4	4	EC2	--

--- click on update.

--- click on update.

The screenshot shows the 'Task Placement' section of the AWS ECS Task Definition configuration. It includes fields for 'Number of tasks' (set to 4), 'Minimum healthy percent' (set to 100), 'Maximum percent' (set to 200), and 'Deployment circuit breaker' (set to 'Disabled'). Below this is the 'Task Placement' section, which lets you customize how tasks are placed on instances within your cluster. It includes a 'Placement Templates' dropdown set to 'Custom', an 'Edit' button, and a 'Strategy' section. The 'Strategy' section contains two 'Spread' strategies based on 'attribute:ecs.a...' and 'instancetype'. A link to 'Add strategy' is also present.

--- Number of tasks =5 – please enter your desired number of tasks here, if you give 5 tasks then it will create 5 replicas.

## Deamonset creating ECS cluster

--- deamonset will create single pod on each node. It is like 1:1.

--- **scenario** – some times you need to run monitoring agent or logging agent on each server then we need to install single container on each node.

--- **note** – you need to have your own task definition of your monitoring agent. So that you can use in new service. In my case I do not have any agent now so I will use already existed task definition.

The screenshot shows the 'Cluster : Test-ECS-CLUSTER' details page. It displays the Cluster ARN (arn:aws:ecs:us-east-1:743580576244:cluster/Test-ECS-CLUSTER), Status (ACTIVE), and Registered container instances (3). Below this, it shows Pending tasks count (0 Fargate, 0 EC2, 0 External), Running tasks count (0 Fargate, 8 EC2, 0 External), Active service count (0 Fargate, 1 EC2, 0 External), and Draining service count (0 Fargate, 0 EC2, 0 External). At the bottom, there are tabs for Services, Tasks, ECS Instances, Metrics, Scheduled Tasks, Tags, and Capacity Providers. The Services tab is selected, showing a table with columns: Service Name, Status, Service type, Task Definition, Desired tasks ..., Running task..., Launch type, and Platform vers... The table contains one row for 'mywebapp-service' with values: ACTIVE, REPLICAS, nginx.3, 5, 5, EC2, and --.

--- click on our ECS cluster to see above service page.

--- click on create.



## Create Service

### Step 1: Configure service

- Step 2: Configure network
- Step 3: Set Auto Scaling (optional)
- Step 4: Review

#### Configure service

A service lets you specify how many copies of your task definition to run and maintain in a cluster. You can optionally use an Elastic Load Balancing load balancer to distribute incoming traffic to containers in your service. Amazon ECS maintains that number of tasks and coordinates task scheduling with the load balancer. You can also optionally use Service Auto Scaling to adjust the number of tasks in your service.

Launch type  FARGATE  EC2  EXTERNAL

Switch to capacity provider strategy [i](#)

Task Definition Family: nginx [Enter a value](#)

Revision: 3 (latest) [i](#)

Cluster: Test-ECS-CLUSTER [i](#)

Service name: my-daemon [i](#)

Service type\*  REPLICA  DAEMON [i](#)

--- here select daemon.

Number of tasks: Automatic [i](#)

Minimum healthy percent: 0 [i](#)

Maximum percent: 100 [i](#)

Deployment circuit breaker: Disabled [i](#)

#### Deployments

Choose a deployment option for the service.

Deployment type\*  Rolling update [i](#)

Blue/green deployment (powered by AWS CodeDeploy) [i](#)  
This sets AWS CodeDeploy as the deployment controller for the service. A CodeDeploy application and deployment group are created automatically with **default settings** for the service. To change to the rolling update deployment type after the service has been created, you must re-create the service and select the "rolling update" deployment type.

#### Task Placement

Lets you customize how tasks are placed on instances within your cluster. Different placement strategies are available to optimize for availability and efficiency.

Placement Templates: Custom [Edit](#)

Customize how tasks are placed by applying strategies and constraints. [Learn more](#)

Strategy: spread(attribute:ecs.availability-zone), spread(instanceId)

#### Task tagging configuration

Enable ECS managed tags [i](#)

Propagate tags from: Do not propagate [i](#)

#### Tags

Key	Value
Add key	Add value

\*Required

[Cancel](#) [Next step](#)

--- Click on next

--- please select none and click on next

The screenshot shows the AWS Services console with the search bar at the top. Below it, several services are listed: EC2, IAM, Route 53, VPC, S3, Certificate Manager, Elastic Container Registry, Elastic Container Service, and CloudWatch. The 'Elastic Container Service' service is highlighted. The main content area is titled 'Create Service' and is on 'Step 3: Set Auto Scaling (optional)'. A note says: 'Automatically adjust your service's desired count up and down within a specified range in response to CloudWatch alarms. You can modify your Service Auto Scaling configuration at any time to meet the needs of your application.' A warning message states: 'Service Auto Scaling Daemon services are not compatible with Auto Scaling.' At the bottom, there are buttons for 'Cancel', 'Previous', and 'Next step'.

Create Service

- Step 1: Configure service
- Step 2: Configure network
- Step 3: Set Auto Scaling (optional)**
- Step 4: Review

#### Set Auto Scaling (optional)

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

**Service Auto Scaling** Daemon services are not compatible with Auto Scaling.

\*Required

[Cancel](#) [Previous](#) [Next step](#)

--- click on next step.

The screenshot shows the AWS Services console with the search bar at the top. Below it, several services are listed: EC2, IAM, Route 53, VPC, S3, Certificate Manager, Elastic Container Registry, Elastic Container Service, and CloudWatch. The 'Elastic Container Service' service is highlighted. The main content area is titled 'Review' and shows the configuration for the service. It includes fields for Cluster (Test-ECS-CLUSTER), Launch type (EC2), Task Definition (nginx:3), Service name (my-deamon), Service type (DAEMON), Minimum healthy percent (0), Maximum percent (100), and Deployment circuit breaker (Disabled). Below this, sections for 'Configure network' and 'Set Auto Scaling (optional)' are shown, both with 'not configured' status. At the bottom, there are buttons for 'Cancel', 'Previous', and 'Create Service'.

--- click on create service.

The screenshot shows the AWS Services console with the search bar at the top. Below it, several services are listed: EC2, IAM, Route 53, VPC, S3, Certificate Manager, Elastic Container Registry, Elastic Container Service, and CloudWatch. The 'Amazon ECS' service is highlighted. The main content area shows the 'Clusters' section with 'Test-ECS-CLUSTER' selected. It displays the 'Service : my-deamon' details, including Cluster (Test-ECS-CLUSTER), Status (ACTIVE), Desired count (3 Automatic), Pending count (0), and Running count (3). The 'Task definition' is nginx:3, 'Service type' is DAEMON, 'Launch type' is EC2, and 'Created By' is am:aws:iam::743580576244:root. Below this, a table shows task details:

Task	Task Definition	Last status	Desired status	Group	Launch type
890de299891a4092b9b4156...	nginx:3	RUNNING	RUNNING	service:my-deamon	EC2
c360a89364f94a34b2b8244e...	nginx:3	RUNNING	RUNNING	service:my-deamon	EC2
feaf154b00e24d66925763bf2...	nginx:3	RUNNING	RUNNING	service:my-deamon	EC2

--- we have only 3 nodes and it only created containers.

Check services in our cluster

Cluster : Test-ECS-CLUSTER

Cluster ARN: arn:aws:ecs:us-east-1:743580576244:cluster/Test-ECS-CLUSTER  
Status: ACTIVE

Registered container instances: 3

Pending tasks count	Running tasks count	Active service count	Draining service count
0 Fargate, 0 EC2, 0 External	0 Fargate, 11 EC2, 0 External	0 Fargate, 2 EC2, 0 External	0 Fargate, 0 EC2, 0 External

Services | Tasks | ECS Instances | Metrics | Scheduled Tasks | Tags | Capacity Providers

Last updated on August 8, 2022 6:47:50 PM (0m ago)

Service Name	Status	Service type	Task Definition	Desired tasks ...	Running task...	Launch type	Platform versi...
mywebapp-service	ACTIVE	REPLICA	nginx:3	5	5	EC2	—
my-deamon	ACTIVE	DAEMON	nginx:3	3	3	EC2	—

--- **note** – here, you can see I have created 2 services, 1<sup>st</sup> service is for application, 2<sup>nd</sup> service is for log monitoring.

## Delete cluster

Delete Cluster

Deleting the cluster also deletes the CloudFormation stack **EC2ContainerService-Test-ECS-CLUSTER**.

Are you sure you want to delete the cluster **Test-ECS-CLUSTER** and all the ECS resources within it?

Deleting ECS tasks and services... **COMPLETE**

Deleting cluster resources. This may take a few minutes...

AWS::AutoScaling::AutoScalingGroup...

Enter the phrase "delete me" into the field below to confirm deletion.

delete me

Last updated on August 8, 2022 6:54:44 PM (4m ago)

Service Name	Status	Service type	Task Definition	Desired tasks ...	Running task...	Launch type	Platform versi...
mywebapp-service	ACTIVE	REPLICA	nginx:3	5	5	EC2	—
my-deamon	ACTIVE	DAEMON	nginx:3	3	3	EC2	—

--- **note** – go to the ECS and click on the AMAZON ECS clusters and click on the delete cluster.

--- **important** – the main problem with ECS cluster node is, we need to manage the nodes, we need to security patch, we need to restart the amazon agent on the nodes. If you do not want to manages nodes and their security then go for the fargates.

---



