



AMAZON ECS (ELASTIC CONTAINER SERVICE)

Amazon Elastic Container Service (Amazon ECS) is a fully managed container orchestration service provided by Amazon Web Services (AWS). It allows you to easily run, stop, and manage Docker containers on a cluster. Containers are a lightweight and portable way to package, distribute, and run applications.

Key features of Amazon ECS include:

1. **Container Orchestration:** ECS simplifies the process of deploying, managing, and scaling containerized applications. It helps you orchestrate the deployment and scaling of Docker containers across a cluster of Amazon EC2 instances.
2. **Scalability:** ECS enables you to scale your application horizontally by adding or removing containers based on demand. It integrates with other AWS services like Auto Scaling, allowing you to automatically adjust the number of container instances based on defined criteria.
3. **Integration with AWS Services:** Amazon ECS seamlessly integrates with other AWS services, such as Amazon Elastic Load Balancing (ELB) for load balancing, Amazon VPC for networking, and AWS Identity and Access Management (IAM) for access control.
4. **Task Definitions:** ECS uses task definitions to specify the configuration details for a set of containers. This includes information such as which Docker image to use, container resource requirements, and data volumes to mount.
5. **Clusters:** ECS organizes container instances into logical groups called clusters. A cluster is a pool of resources that can be used to run containers. You can have multiple clusters in an AWS account.
6. **Service Definitions:** ECS allows you to define and run long-running applications using services. A service allows you to define the desired number of tasks and how they should be distributed across your cluster.
7. **Fargate:** In addition to running ECS on EC2 instances, AWS offers a service called AWS Fargate that allows you to run containers without managing the underlying infrastructure. With Fargate, you only pay for the vCPU and memory that you use, making it a serverless option for running containers.

Amazon ECS is suitable for a variety of applications, including microservices architectures, monolithic applications, and batch processing workloads. It provides a flexible and scalable platform for containerized applications in the AWS cloud environment.



USE CASES OF AMAZON ECS:

Amazon ECS (Elastic Container Service) is a versatile container orchestration service that can be used for various use cases across different industries. Here are some common use cases for Amazon ECS:

1. **Microservices Architecture:** ECS is well-suited for deploying and managing microservices-based applications. It allows you to break down your application into

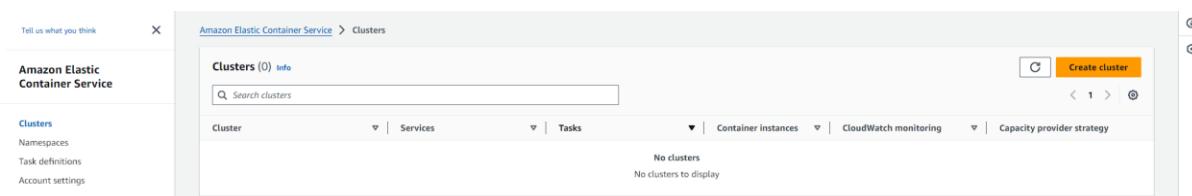
smaller, independent services that can be developed, deployed, and scaled independently.

2. **Web Applications:** Deploying and scaling web applications is a common use case for ECS. By defining services and tasks in ECS, you can easily manage the deployment of web servers, load balancers, and other components of your application.
3. **Batch Processing:** ECS can be used for running batch processing workloads where you need to process large amounts of data periodically. You can define ECS tasks to execute batch jobs, and ECS can scale the number of tasks based on demand.
4. **Continuous Integration/Continuous Deployment (CI/CD):** Integrate ECS into your CI/CD pipeline to automate the building, testing, and deployment of containerized applications. ECS can be used to manage the deployment of new versions of your application, making it easy to achieve continuous delivery.
5. **Machine Learning Inference:** For deploying and scaling machine learning models in production, ECS can be a suitable choice. You can containerize your machine learning models and use ECS to manage the deployment and scaling of inference services.
6. **DevOps Workflows:** ECS can be integrated into DevOps workflows to streamline the process of deploying and managing applications. It facilitates the automation of tasks related to containerized application deployment and updates.
7. **Hybrid Architectures:** For organizations with hybrid cloud architectures, ECS can be used to manage containerized applications both on-premises and in the cloud. This flexibility allows for a seamless integration of container workloads across different environments.
8. **Stateless Applications:** ECS is particularly well-suited for stateless applications where each instance of the application can handle requests independently. Statelessness makes it easier to scale applications horizontally by adding or removing containers.
9. **Multi-Region Deployments:** If your application needs to be deployed across multiple AWS regions for high availability and disaster recovery, ECS can help manage the deployment and scaling of containers in each region.
10. **Cost-Effective Scaling:** ECS, especially when combined with AWS Fargate, provides a cost-effective solution for running containers. With Fargate, you don't need to manage the underlying infrastructure, and you only pay for the resources your containers consume.

These use cases highlight the flexibility of Amazon ECS, making it suitable for a wide range of applications and scenarios in the cloud environment.

👉 TO BEGIN WITH THE LAB:

1. Log in to Amazon Console. There you need to search ECS (Elastic Container Service)



2. Now you need to create a cluster. Click on create cluster.
3. Give it a name, then move forward.

Create cluster Info

An Amazon ECS cluster groups together tasks, and services, and allows for shared capacity and common configurations. All of your tasks, services, and capacity must belong to a cluster.

Cluster configuration

Cluster name

demo-ecs-cluster

There can be a maximum of 255 characters. The valid characters are letters (uppercase and lowercase), numbers, hyphens, and underscores.

Default namespace - *optional*

Select the namespace to specify a group of services that make up your application. You can overwrite this value at the service level.

X

4. In the infrastructure part, AWS Fargate will be selected by default so uncheck Fargate. You need to select Amazon EC2 instances only.

▼ Infrastructure Info Customized

Your cluster is automatically configured for AWS Fargate (serverless) with two capacity providers. Add Amazon EC2 instances, or external instances using ECS Anywhere.

- AWS Fargate (serverless)**
Pay as you go. Use if you have tiny, batch, or burst workloads or for zero maintenance overhead. The cluster has Fargate and Fargate Spot capacity providers by default.
- Amazon EC2 instances**
Manual configurations. Use for large workloads with consistent resource demands.

5. As soon as you click on ec2, it will ask you to create an ASG (Auto Scaling Group).
6. Now you need to select On-demand for provisioned model.
7. Then for the operating system select Amazon Linux 2.
8. For the instance type select t2.micro
9. Then in the desired capacity select minimum to be 1 and maximum to be 3.
10. In the last select your key pair. If you don't have a key pair no issues just create new.

Amazon EC2 instances

Manual configurations. Use for large workloads with consistent resource demands.

Auto Scaling group (ASG) | [Info](#)

Use Auto Scaling groups to scale the Amazon EC2 instances in the cluster.

[Create new ASG](#)

Provisioning model

Select a provisioning model for your instances

On-demand

With on-demand instances, you pay for compute capacity by the hour, with no long-term commitments or upfront payments.

Spot

Amazon EC2 Spot instances let you take advantage of unused EC2 capacity in the AWS cloud. Spot instances are available at up to a 90% discount compared to on-demand prices.

Operating system/Architecture

Choose the Windows operating system or Linux architecture for your instance.

[Amazon Linux 2](#)

EC2 instance type

Choose based on the workloads you plan to run on this cluster.

t2.micro

Free tier eligible

i386, x86_64

1 vCPU 1 GiB Memory

Desired capacity

Specify the number of instances to launch in your cluster.

Minimum

1

Maximum

3

SSH Key pair

If you do not specify a key pair, you can't connect to the instances via SSH unless you choose an AMI that is configured to allow users another way to log in.

[demo-user](#)



[Create a new key pair](#)

11. So, when you will scroll down you will see the network configuration leave them to default. And create your cluster.

12. Till your cluster is creating, switch to task definition and create it.

The screenshot shows the 'Task definitions' section of the AWS ECR interface. At the top, there's a header with 'Task definitions (0)' and a 'Create new task definition' button. Below the header, there are filters for 'Filter by status' (set to 'Active') and a search bar. The main table has a single row with the message 'No task definitions' and 'No task definitions to display.' At the bottom of the table, there's a 'Create new task definition' button.

13. Click on create new task definition.

14. In there now you need to give it a name.

Create new task definition [Info](#)

Task definition configuration

Task definition family [Info](#)
Specify a unique task definition family name.

Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.

15. Then in infrastructure requirements, select launching type as ec2 instances.
16. The OS should be Linux.
17. Then the select host in the network mode.
18. Now in the task size you need to select CPU as 0.25 and memory as 0.5gb, so that it is compatible with our instance type that is t2.micro.

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

Launch type [Info](#)
Selection of the launch type will change task definition parameters.

AWS Fargate
Serverless compute for containers.

Amazon EC2 instances
Self-managed infrastructure using Amazon EC2 instances.

OS, Architecture, Network mode
Network mode is used for tasks and is dependent on the compute type selected.

Operating system/Architecture [Info](#)

Network mode [Info](#)

Task size [Info](#)
Specify the amount of CPU and memory to reserve for your task.

CPU	Memory
<input type="text" value=".25 vCPU"/>	<input type="text" value=".5 GB"/>

19. And within here you'll have to specify the details related to what are the container images that you'd like. You'll be using a sample nginx container image.

▼ Container - 1 [Info](#) Essential container [Remove](#)

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

Name	Image URI	Essential container
<input type="text" value="nginx"/>	<input type="text" value="nginx:latest"/>	<input checked="" type="checkbox" value="Yes"/>

20. So, now just scroll down to the bottom to create your task definition.

21. If you go back to your cluster and open it, you'll see that it is empty. There is no service running.
22. So, quickly click on create, to create a service.

The screenshot shows the AWS ECS Cluster Overview page. At the top, the cluster name 'demo-ecs-cluster' is displayed with an ASG badge. Below the cluster overview table, the 'Services' tab is selected. A single service named 'Draining' is listed as active. At the bottom of the services section, there is a prominent orange 'Create' button.

23. In here you need to select compute options as Launch type.
24. Then select Launch type as EC2.

The screenshot shows the 'Create' dialog for a new service. In the 'Compute configuration (advanced)' section, the 'Launch type' option is selected and highlighted with a blue border. The 'Capacity provider strategy' option is also present but not selected. Other sections like 'Environment' and 'Deployment configuration' are visible but not the focus.

25. In the deployment configuration select Service as application type.
26. In the family select your task definition that you created.
27. Give a service name to it.
28. In the desired task select it to 1.
29. Then keep everything to default and click on create.

Deployment configuration

Application type | [Info](#)

Specify what type of application you want to run.

Service

Launch a group of tasks handling a long-running computing work that can be stopped and restarted. For example, a web application.

Task

Launch a standalone task that runs and terminates. For example, a batch job.

Task definition

Select an existing task definition. To create a new task definition, go to [Task definitions](#).

Specify the revision manually

Manually input the revision instead of choosing from the 100 most recent revisions for the selected task definition family.

Family

demo-task-definition

Revision

1 (LATEST)

Service name

Assign a unique name for this service.

nginx-service

Service type | [Info](#)

Specify the service type that the service scheduler will follow.

Replica

Place and maintain a desired number of tasks across your cluster.

Daemon

Place and maintain one copy of your task on each container instance.

Desired tasks

Specify the number of tasks to launch.

1

30. After some time, you'll see that your service is in active state.

Services | Tasks | Infrastructure | Metrics | Scheduled tasks | Tags

Services (1) [Info](#)

Filter services by value Any launch type Any service type

Service name	ARN	Status	Service type	Deployments and tasks	Last deployment	Task definition
nginx-service	arn:aws:lambda:...	Active	REPLICA	1/1 Tasks running	Completed	demo-task-defin...

31. Now if go to your ec2 instance, there you have to copy your public IP address and paste it in the new tab or browser.

Not secure 34.220.88.111

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](#).
Commercial support is available at [nginx.com](#).

Thank you for using nginx.