

Introduction to Containers

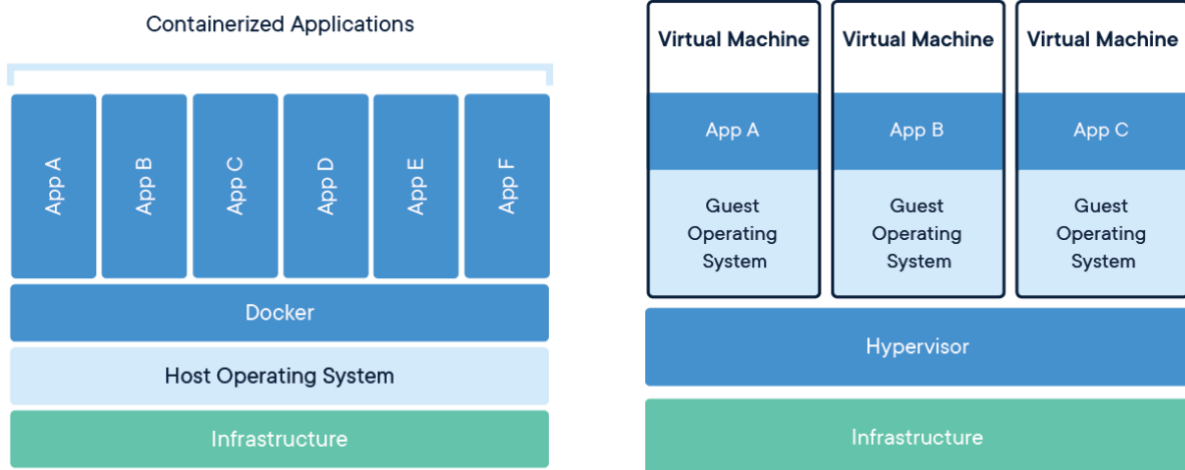


Fig. 1: A comparison of using containers versus virtual machines

Containerization is the standard in modern software development because it allows an application to run in a single package alongside all of its dependencies, libraries, runtime environment, and other configurations. This helps solve an age-old software testing problem, “But it works on my machine!”

Using virtual machines was common for this use case, however, virtual machines include an operating system. This adds to the total size of your application, increasing costs.

A containerization tool runs on the host operating system and manages your containers, making them more portable, and isolated, and reducing your total cost.

But how do we do this?

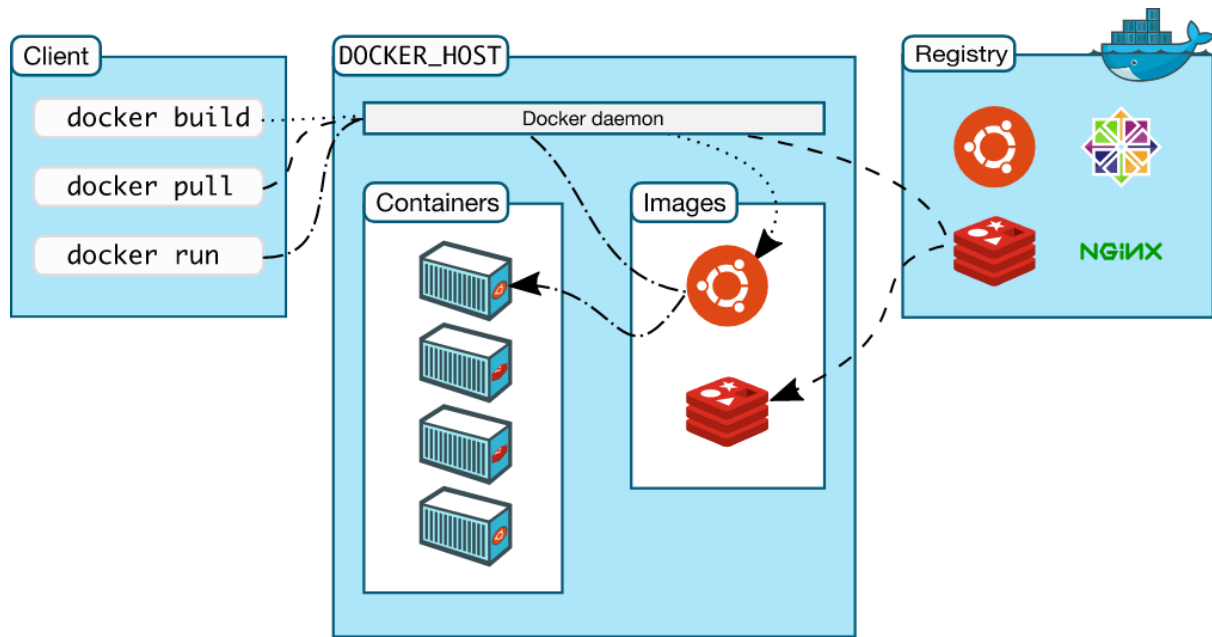


Fig. 2: Docker architecture

Docker is the most popular containerization tool. It allows you to package source code and other files into a Docker image, and then run a container from that image. A container is a **running instance** of that image.

In another perspective, think of a Docker image as the recipe for a cake, and a container as a cake you baked from it. The image spells out what ingredients should go into the cake. You cannot eat if you stop there. The container is the cake.

Docker images can be uploaded online to [Docker Hub](https://hub.docker.com/) or other container registries such as **Amazon ECR** (Elastic Container Registry).

You can use commands on your terminal such as `docker pull` to install a docker image from Dockerhub, `docker build` to package your application into a container image using a **Dockerfile** (a file that specifies how your image will be constructed), and `docker run` to run a container from an image.

The Need for Container Orchestration

Running one container is easy, but what if you need to run ten, a hundred, or even a thousand of the same containers?

This is where container orchestration comes into the picture, which makes managing several containers much easier.

Common container orchestration tools include:

- Docker Compose
- Amazon Elastic Container Service (ECS)
- Amazon Elastic Kubernetes Service (EKS) / Kubernetes

In this workshop, I will provide an overview of Amazon ECS, its offerings, and a step-by-step walkthrough of deploying a containerized game on Amazon ECS.

Introducing Amazon ECS

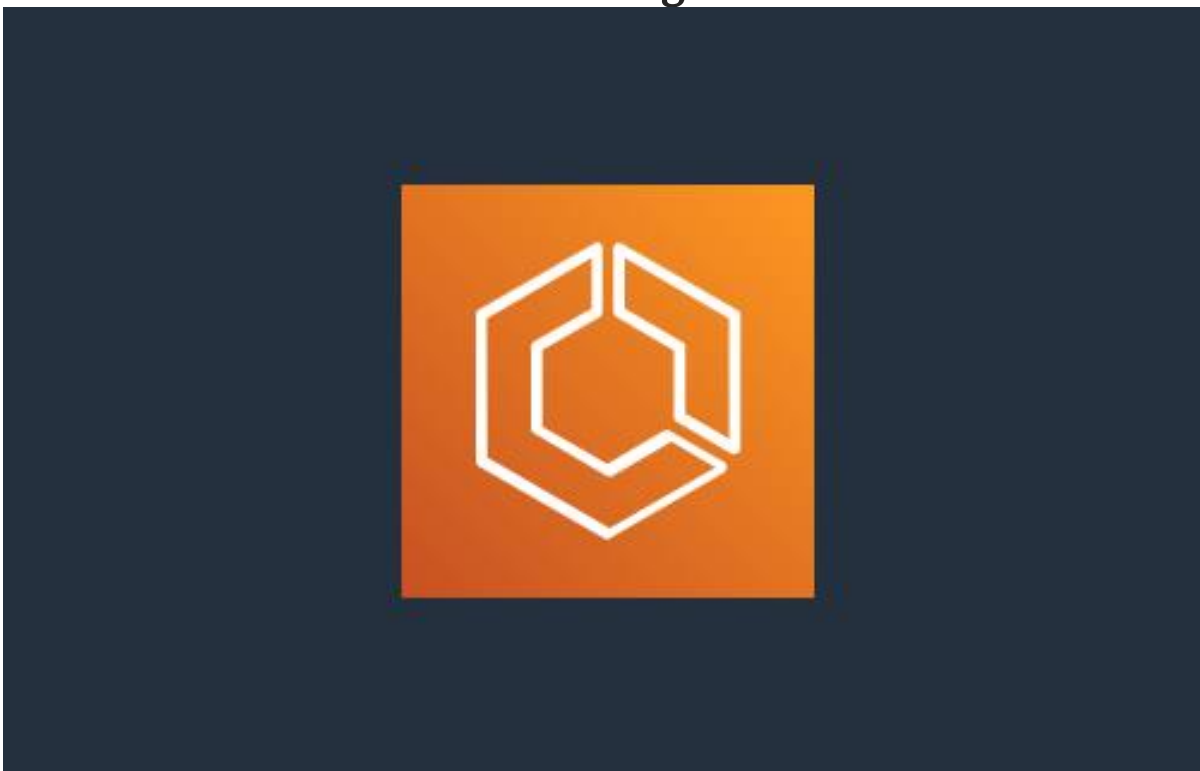


Fig. 3: Amazon ECS Logo

Amazon Elastic Container Service (ECS) is a highly scalable fully-managed container orchestration service that helps you to more efficiently deploy, manage, and scale containerized applications. It deeply integrates with the AWS environment to provide an easy-to-use solution for running container workloads in the cloud that can range from hosting a simple web application to managing a distributed microservices-based architecture.

The benefits of ECS include:

- **Easy and simple deployment** — ECS eliminates the need to set up and maintain the infrastructure of Kubernetes clusters by taking responsibility for these tasks.
- **Scheduling capabilities** — that enable you to schedule services, applications, and batch processes.
- **Managed availability** — ECS is responsible for maintaining application availability and helps you scale up or down as needed to ensure capacity demands are met.
- **Native integration** — with a wide range of features like AWS ELB, Amazon Virtual Private Cloud (Amazon VPC), IAM, and [AWS EBS](#).
- **Integration with existing tools** — ECS provides simple APIs that let you integrate with your CI/CD pipeline and your existing tools.

Core concepts and architecture:

Clusters:

Clusters are logical groups of container instances where tasks (containers) are run. A cluster can use either **EC2 instances** (Amazon EC2 launch type) or **Fargate** (a serverless compute engine) to host tasks.

ECS supports multiple clusters, and each cluster can scale independently, allowing you to manage the workloads of different teams or environments separately (e.g., dev/test/prod).

Tasks and Task Definitions:

Task Definitions are **blueprints** where you can specify how your containerized applications should run. They define important components such as:

- Docker images (Container runtime)
- Allocated CPU and Memory resources
- Networking configs
- Environment variables, storage, and logs
- Number of containers
- Data Volumes

Tasks can be instantiated from Task Definitions. Each task is a running instance of your containerized applications.

Services:

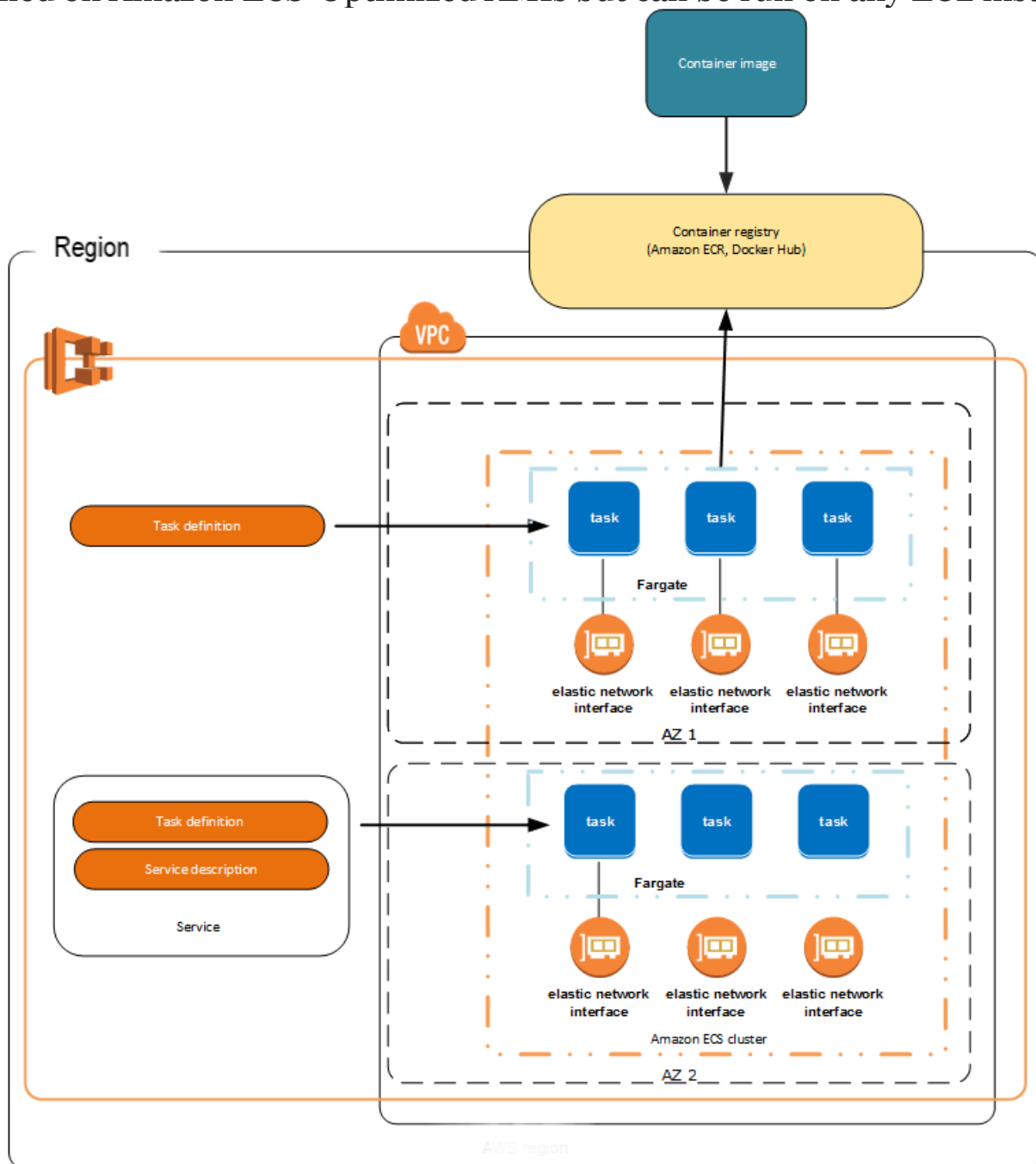
Services ensure that a specific number of tasks are always running and can provide useful functionalities such as load balancing, automatic failover, and auto-scaling.

- ECS integrates with **Elastic Load Balancer** to spread incoming traffic across tasks.

- Services can also be configured for auto-scaling based on policies or application demand.

Container Agent:

The ECS agent runs on EC2 instances to register the instances to an ECS cluster, communicate with the ECS API, and manage the task lifecycle. This agent is pre-installed on Amazon ECS-Optimized AMIs but can be run on any EC2 instance.



ECS Launch Types:

Amazon ECS supports two launch types for deploying and running containers.

With the **EC2 launch type**, you have full control over the underlying EC2 instances (container instances) that host your tasks. This launch type is ideal if you need custom AMIs, SSH access to EC2 instances, or Fine-grained control.

Key Features of ECS on EC2:

- Full control over compute, networking, and IAM.
- Ability to use reserved instances or spot instances for cost savings.
- Manage EC2 instances with auto-scaling groups.

AWS Fargate is a serverless compute engine for containers. It removes the need to manage the EC2 instances that your containers run on. Instead, AWS abstracts away the infrastructure layer, allowing you to focus purely on your application.

Key Features of Fargate:

- No need to provision or manage EC2 instances.
- Auto-scaling at the task level, Fargate automatically scales resources for individual tasks.
- Pay only for the compute resources and storage you use.
- Seamless integration with other AWS services.

Fargate is ideal when you want to **reduce management overhead** and need simple scaling.

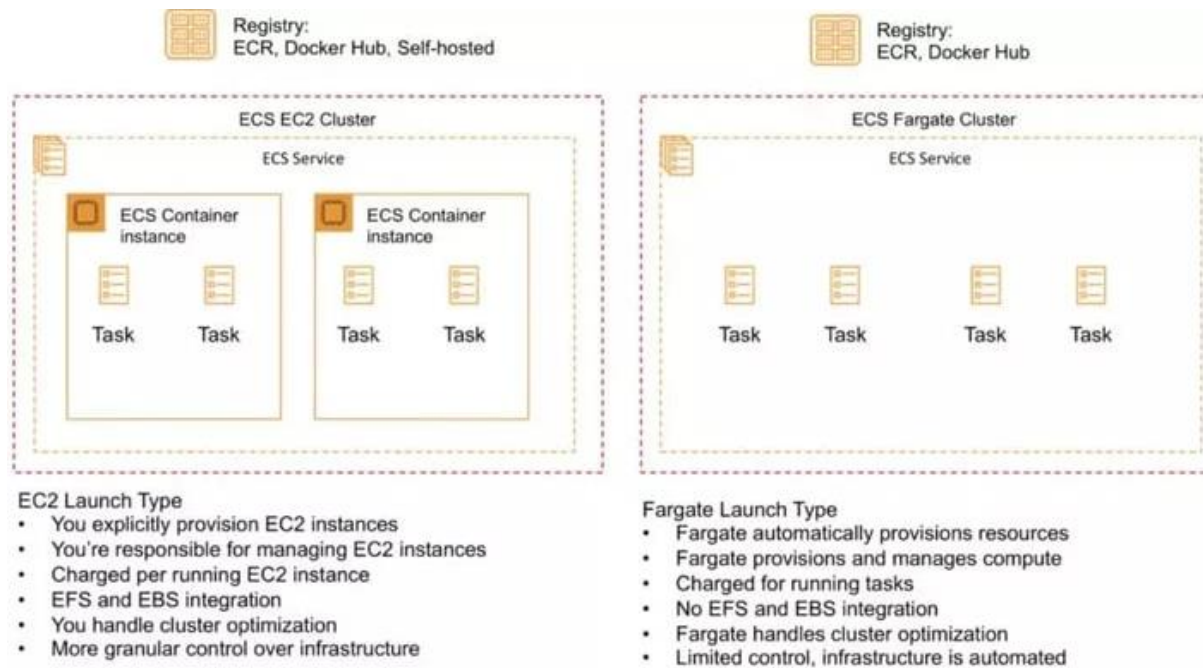


Fig. 5: Comparison between ECS Launch Types

Networking in ECS

Amazon ECS has flexible networking options to make sure your tasks have the right connectivity in your cloud environment.

VPC Integration:

ECS tasks can be launched in a VPC and are fully integrated with the Amazon VPC networking model. They can use all of VPC's features, including Security Groups, NACLs, VPC Peering, PrivateLink, and more.

Task-level networking:

ECS supports task-level networking using `awsvpc` mode, which gives each task its own **Elastic Network Interface (ENI)** and **IP address** in the VPC. This offers better control over network traffic and isolation between components.

Service Discovery and Load Balancing:

ECS services can be integrated with Application Load Balancers or Network Load Balancers to distribute traffic to tasks.

Service Discovery is also supported via Route 53. This allows ECS services to automatically register tasks to a DNS namespace, giving you the ability to discover and connect to your services.

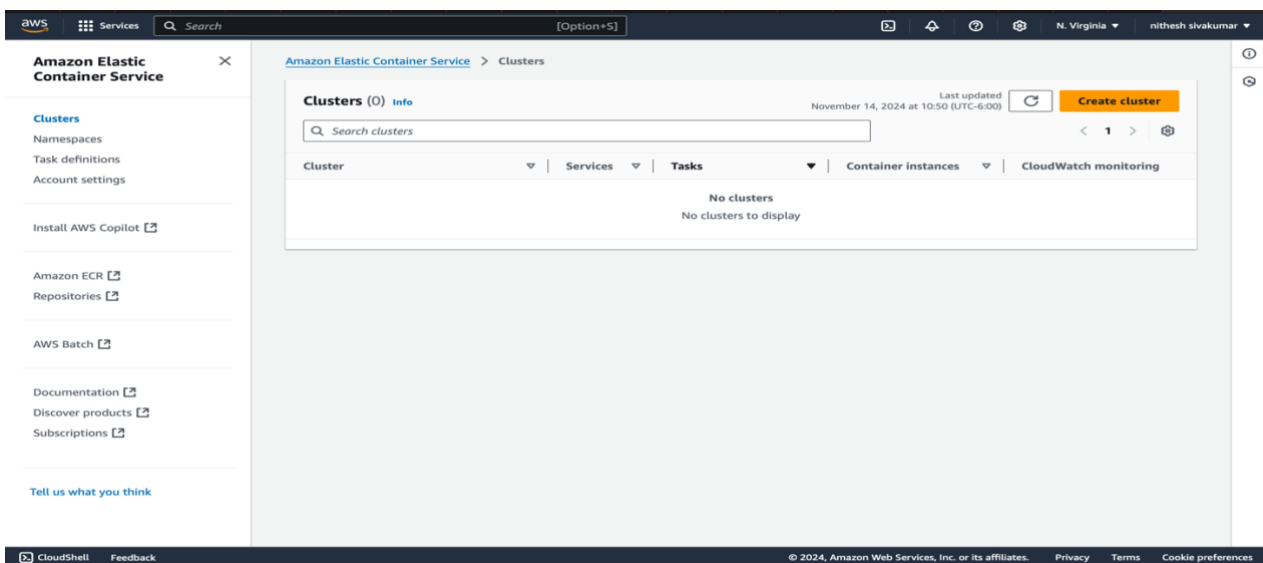
ECS Deployment Demo

In this demo, we are:

- Creating an ECS Cluster for Fargate
- Defining a task definition using this [container image](#)
- Launching a single task in a service, using that task definition

Task 1:

Ensure your region is set to US-East-1. Navigate to the ECS console. In the left menu, select clusters, then click “Create Clusters”.



Input these options and keep the rest as default.

Amazon Elastic Container Service

Amazon Elastic Container Service > Create cluster

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

ecsdemo

Cluster name must be 1 to 255 characters. Valid characters are a-z, A-Z, 0-9, 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.

ecsdemo

▼ Infrastructure [Info](#) Serverless

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

☒ **AWS Fargate (serverless)**

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

☐ **Amazon EC2 instances**

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

[External instances using ECS Anywhere](#) can be registered after cluster creation is complete.

CloudShell Feedback

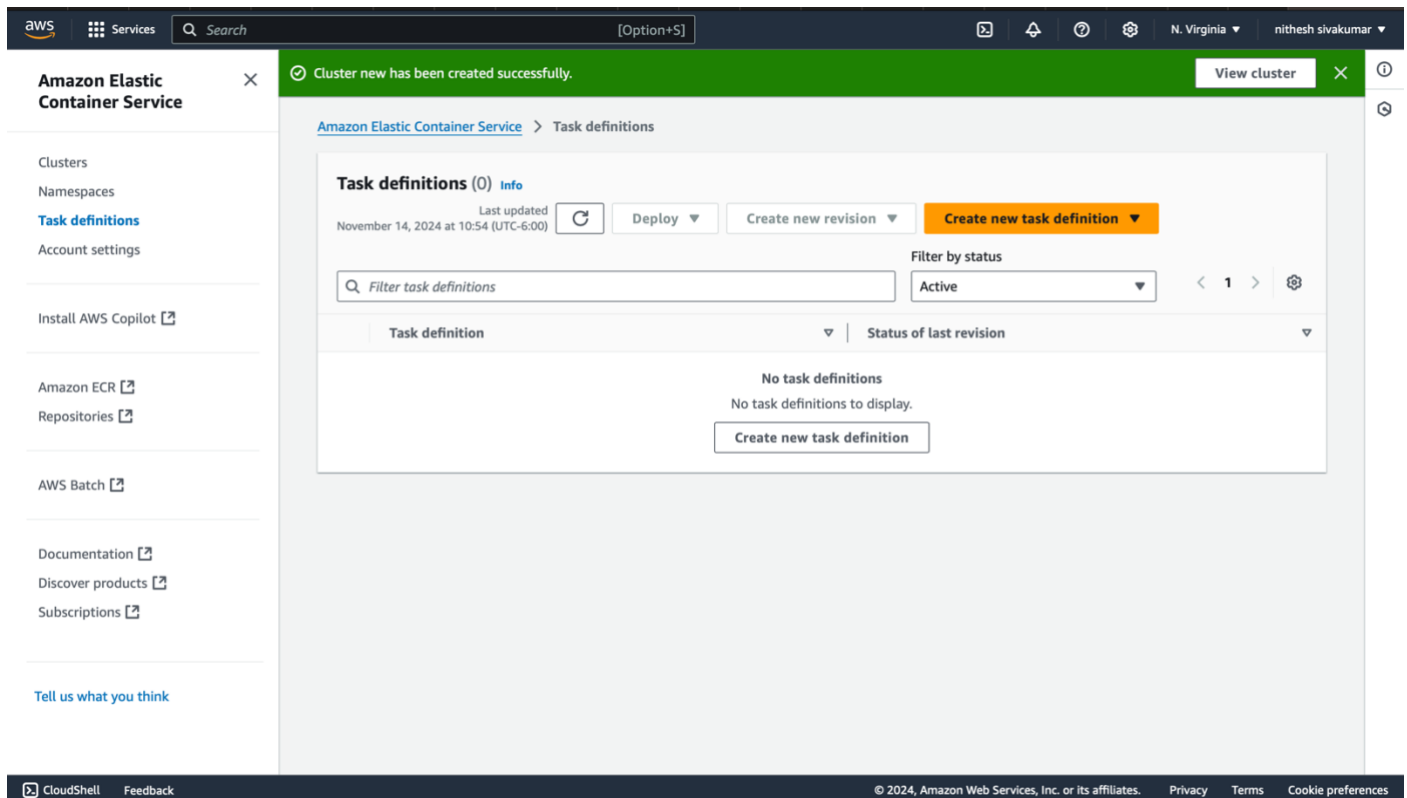
© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Scroll down to the bottom and click “Create”.

You may move on to Task 2 while the cluster is provisioning.

Task 2:

Move over to Task Definitions in the left-hand menu.



Click “Create new task definition”.

Name your task definition anything you would like, for this example, I have used “ecsdemoTD”.

Ensure you have AWS Fargate selected. The containers will run on top of a Linux x86_64 machine. For CPU and Memory, the default options should work fine because the container we run is not large.

Under the Task roles section, leave Task role empty because we will not be making API requests to other AWS services for this demo. Ensure that you have a default IAM role called “ecsTaskExecutionRole”. If not, AWS can create it for you.

▼ Infrastructure requirements

Specify the infrastructure requirements for the task definition.

Launch type | Info

Selection of the launch type will change task definition parameters.

☒ AWS Fargate
Serverless compute for containers.

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

OS, Architecture, Network mode

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

Operating system/Architecture | Info

Linux/X86_64

Network mode | Info

awsvpc

Task size | Info

Specify the amount of CPU and memory to reserve for your task.

CPU

1 vCPU

Memory

3 GB

▼ Task roles - conditional

Task role | Info

A task IAM role allows containers in the task to make API requests to AWS services. You can create a task IAM role from the [IAM console](#).

-

Task execution role | Info

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

ecsTaskExecutionRole

▼ Task placement - optional

Task placement constraints are not supported for AWS Fargate launch type.

Follow the filled out fields for Container-1. This will ensure your container, named “2048-game” (or any other name you may choose) runs the docker image blackicebird/2048. This container is considered an essential container.

In ECS, an essential container is a key component of a task definition. An essential container must be completed for the task to be considered healthy. If an essential container fails or stops for any reason, the entire task is marked as failed. Essential containers are commonly used to run the main application or service within the task.

For port-mappings, the container will listen for HTTP traffic on port 80. For resource allocation, the default will work fine.

▼ 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

2048-game

docker.io/blackicebird/2048

Yes

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

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

Private registry

Info

Store credentials in Secrets Manager, and then use the credentials to reference images in private registries.

☐ Private registry authentication

Port mappings

Info

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

Container port

Protocol

Port name

App protocol

Remove

80

TCP

2048-game-port80

HTTP

Remove

Add port mapping

Read only root file system

Info

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

☐ Read only

Resource allocation limits - conditional

Info

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

CPU

GPU

Memory hard limit

Memory soft limit

1

1

3

1

in vCPU

in GB

in GB

in GB

► Environment variables - optional

▼ Logging - optional

i

CPU and memory allocation for a sidecar

There are logging options that will automatically add a sidecar to your task definition if it does not already exist. AWS provides CPU and memory

Uncheck logging because we will not be using CloudWatch for this demo.

▼ Logging - optional

i

CPU and memory allocation for a sidecar

There are logging options that will automatically add a sidecar to your task definition if it does not already exist. AWS provides CPU and memory adjustment recommendations based on the selected options.

1

We recommend that you use log collection for tasks running on AWS Fargate. Learn more about [log collection](#).

Log collection

Info

Configure your task to send container logs to a logging destination using a default configuration. See pricing information on [Amazon CloudWatch](#).

☐ Use log collection

► Restart policy - optional, new

Configure a container restart policy to restart individual containers locally without restarting the whole task.

► HealthCheck - optional

► Startup dependency ordering - optional

► Container timeouts - optional

► Container network settings - optional

► Docker configuration - optional

► Resource limits (Ulimits) - optional

► Docker labels - optional

Leave the rest as default, scroll to the bottom, and click Create.

Amazon Elastic Container Service > Task definitions > ecsdemoTD > Revision 1 > Containers

ecsdemoTD:1 Deploy Actions Create new revision

Overview Info

ARN arn:aws:ecs:us-east-1:472595282557:task-definition/ecsdemoTD:1	Status ACTIVE	Time created October 26, 2024 at 07:09 (UTC-5:00)	App environment Fargate
Task role -	Task execution role ecsTaskExecutionRole	Operating system/Architecture Linux/X86_64	Network mode awsvpc

[Containers](#) | [JSON](#) | [Task placement](#) | [Volumes \(0\)](#) | [Requires attributes](#) | [Tags](#)

Task size

Task CPU
1024 units (1 vCPU)

Task CPU maximum allocation for containers

CPU (unit)

☒ 2048-game ☐ Shared task CPU

Task memory
3072 MiB (3 GB)

Task memory maximum allocation for container memory reservation

Memory (MiB)

☒ 2048-game ☐ Shared task memory

Containers Info

Container name	Image	Private registry	Essential	CPU	Memory hard/soft limit	GPU
2048-game	docker.io/blackicebird/2048-game	-	Yes	0	-/-	-

Task 3:

Navigate to Clusters in the left-hand menu.

Amazon Elastic Container Service > Clusters

Clusters (1) Info October 26, 2024 at 07:14 (UTC-5:00) Create cluster

[Clusters](#) | [Services](#) | [Tasks](#) | [Container instances](#) | [CloudWatch monitoring](#) | [Capacity provider strategy](#)

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

It's time to run our tasks!

Click ecsdemo and then navigate to the services section and click Create.

Amazon Elastic Container Service > Clusters > ecsdemo > Services

ecsdemo October 26, 2024 at 07:16 (UTC-5:00) Update cluster Delete cluster

Cluster overview

ARN arn:aws:ecs:us-east-1:472595282557:cluster/ecsdemo	Status Active	CloudWatch monitoring Default	Registered container instances -
Services Draining -	Active -	Tasks Pending -	Running -
Encryption Managed storage -	Fargate ephemeral storage -		

[Services](#) | [Tasks](#) | [Infrastructure](#) | [Metrics](#) | [Scheduled tasks](#) | [Tags](#)

Services (0) Info Manage tags Update Delete service Create

Any launch type Any service type

Service name	ARN	Status	Service type	Deployments and tasks	Last deployment	Task definition	Launc...
No services No services to display. Create							

Leave environment as default:

Environment

AWS Fargate

Existing cluster

ecsdemo

▼ Compute configuration *(advanced)*

Compute options | [Info](#)
To ensure task distribution across your compute types, use appropriate compute options.

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

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

Capacity provider strategy | [Info](#)
Select either your cluster default capacity provider strategy or select the custom option to configure a different strategy.

☐ Use cluster default
No default capacity provider strategy configured for this cluster.

☒ Use custom (Advanced)

Capacity provider

Base | [Info](#)Weight | [Info](#)

FARGATE ▼01

Add capacity provider

Platform version | [Info](#)
Specify the platform version on which to run your service.

LATEST ▼

For deployment configuration, ensure you have selected service for application type and set the values to what is listed below. If you named your task definition different from mine, then use yours.

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

ecsdemoTD ▼

Revision

1 (LATEST) ▼

Service name
Assign a service name that is unique for this cluster.

ecsdemoservice

Up to 255 letters (uppercase and lowercase), numbers, underscores, and hyphens are allowed. Service names must be unique within a cluster.

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

► Deployment options

► Deployment failure detection [Info](#)

Expand networking:

▼ Networking

VPC | [Info](#)
Select a VPC to use for your Amazon ECS resources.

vpc-0c906cd5b5957872a
default

↺

Create a new VPC ↗

Subnets
Choose the subnets within the VPC that the task scheduler should consider for placement.

Choose subnets

Clear current selection

subnet-0d3bd46a7478083f0
us-east-1c 172.31.16.0/20

subnet-0870455352f7760ee
us-east-1d 172.31.32.0/20

subnet-01b98931223e608d3
us-east-1f 172.31.64.0/20

subnet-0bde849e4f6781297
us-east-1e 172.31.48.0/20

subnet-0e59335367f8e83cf
us-east-1a 172.31.0.0/20

subnet-02fae2c675be5ec98
us-east-1b 172.31.80.0/20

Security group | [Info](#)
Choose an existing security group or create a new security group.

☒ Use an existing security group
☐ Create a new security group

Security group name
Choose an existing security group.

Choose security groups

sg-01609fe34a22691dc
default

Public IP | [Info](#)
Choose whether to auto-assign a public IP to the task's elastic network interface (ENI).

☒ Turned on

Ensure you have a default VPC, or create a VPC. Under “Security group”, select “Create a new security group”. Enter a name for your security group and a fitting description. Ensure that you have an inbound rule that accepts HTTP traffic on port 80 from any source.

Make sure your Public IP is turned on because we will be using this to connect to the task.

Security group [Info](#)

Choose an existing security group or create a new security group.

- ☐ Use an existing security group
- ☒ Create a new security group

Security group details

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

Security group name

ecsdemo-sg

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

Security group description

ECS Demo SG, HTTP from anywhere on port 80

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

Inbound rules for security groups

Add one or more ingress rules for your security group.

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

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

Add rule

Public IP [Info](#)

Choose whether to auto-assign a public IP to the task's elastic network interface (ENI).

- ☒ Turned on

Review your configurations to make sure they are correct and scroll down.

For this demo, we will not be using a load balancer or auto-scaling because we will only be deploying a single task, so leave them unchecked and click Create.

▼ **Load balancing - optional** [Info](#)

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

Load balancer type [Info](#)

Configure a load balancer to distribute incoming traffic across the tasks running in your service.

None

▼ **Service auto scaling - optional** [Info](#)

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.

☐ Use service auto scaling

Configure service auto scaling to adjust your service's desired count

► **Volume - optional** [Info](#)

Configure a data volume to provide additional storage for the containers in the task.

► **Tags - optional** [Info](#)

Tags help you to identify and organize your resources.

Cancel **Create**

Wait until your service is deployed before moving onto Task 4.

Task 4:

ecsdemo

October 26, 2024 at 07:36 (UTC-5:00)

Last updated

Update cluster

Delete cluster

Cluster overview

ARN
arn:aws:ecs:us-east-1:472595282557:cluster/ecsdemo

Status
Active

CloudWatch monitoring
Default

Registered container instances
-

Services
Draining
-

Active
1

Tasks
Pending
-

Running
1

Encryption
Managed storage
-

Fargate ephemeral storage
-

Services

Tasks

Infrastructure

Metrics

Scheduled tasks

Tags

Services (1) Info

Filter services by value

Filter launch type
Any launch type

Filter service type
Any service type

Manage tags

Update

Delete service

Create

Service name	ARN	Status	Service type	Deployments and tasks	Last deployment	Task definition	Launc...
ecsdemoservice	arn:aws.ecs...	Active	REPLICA	1/1 Tasks running	Completed	ecsdemoTD:1	-

Your service should have Active status and 1/1 Tasks running. Click your service and move to the Tasks tab.

Amazon Elastic Container Service

Clusters

ecsdemo

Services

ecsdemoservice

Tasks

ecsdemoservice Info

October 26, 2024 at 07:37 (UTC-5:00)

Last updated

Update service

Delete service

Health and metrics

Tasks

Logs

Deployments

Events

Configuration and networking

Tags

Tasks (1/1)

Filter tasks by property or value

Filter desired status
Running

Filter launch type
Any launch type

Stop

Task	Last status	Desired status	Task definit...	Health status	Started at	Container instan...	Launch type	Platform version	CPU	Memory
81ca32afc...	Running	Running	ecsdemoTD:1	Unknown	8 minutes ago	-	FARGATE	1.4.0	1 vCPU	3 GB

Containers for task 81ca32afc16549d1b25719efb53f96c8

Containers (1)

Filter containers

Container name	Container runtime ID	Image URI	Image Digest	Status	Health status	CPU	Memory hard/soft limit
2048-game	81ca32afc16549d...	docker.io/...	sha256:04...	Running	Unknown	0	- / -

Here you can see your task's details, like Status, Health, CPU, Memory, Launch type, Image URI, etc.

Click your task to view its configurations.

Amazon Elastic Container Service > Clusters > ecsdemo > Services > ecscdemo > Tasks > 81ca32afc16549d1b25719efb53f96c8 > Configuration

81ca32afc16549d1b25719efb53f96c8 Last updated October 26, 2024 at 07:40 (UTC-5:00) [Refresh](#) [Stop](#)

[Configuration](#) | [Logs](#) | [Networking](#) | [Volumes \(0\)](#) | [Tags](#)

Task overview

ARN arn:aws:ecs:us-east-1:472595282557:task/ecsdemo/81ca32afc16549d1b25719efb53f96c8	Last status Running	Desired status Running	Started/Created at October 26, 2024 at 07:29 (UTC-5:00) October 26, 2024 at 07:29 (UTC-5:00)
---	-------------------------------------	--	--

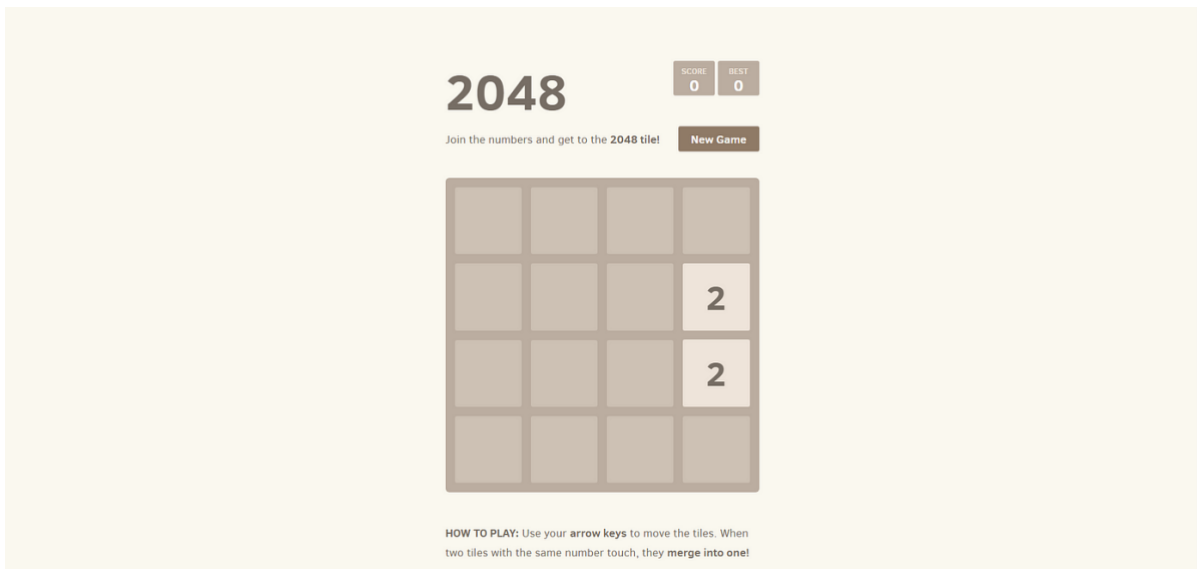
Fargate ephemeral storage

Encryption Info Default AWS Fargate encryption	Size (GiB) 20
---	------------------

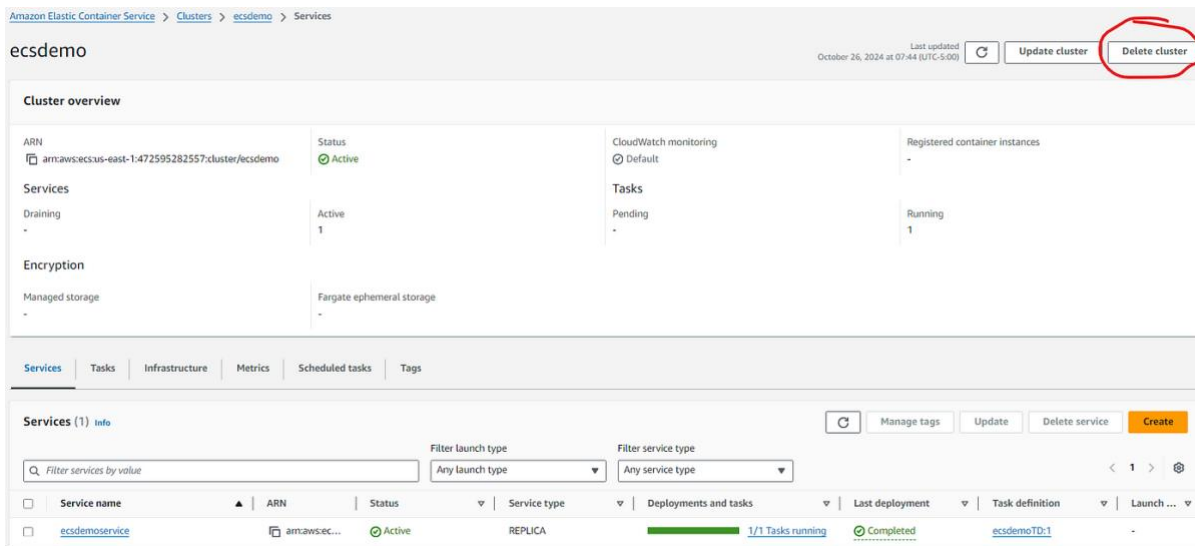
Configuration

Operating system/Architecture Linux/X86_64	Capacity provider FARGATE	ENI ID eni-0c898b62def23137d	Public IP 44.203.96.42 open address
CPU Memory 1 vCPU 3 GB	Launch type FARGATE	Network mode awsvpc	Private IP 172.31.84.52
Platform version 1.4.0	Container instance ID *	Subnet ID subnet-02fae2c675be5ec98	MAC address 12:0f:29:6c:25:a1
Task definition: revision ecsdemoTD:1			

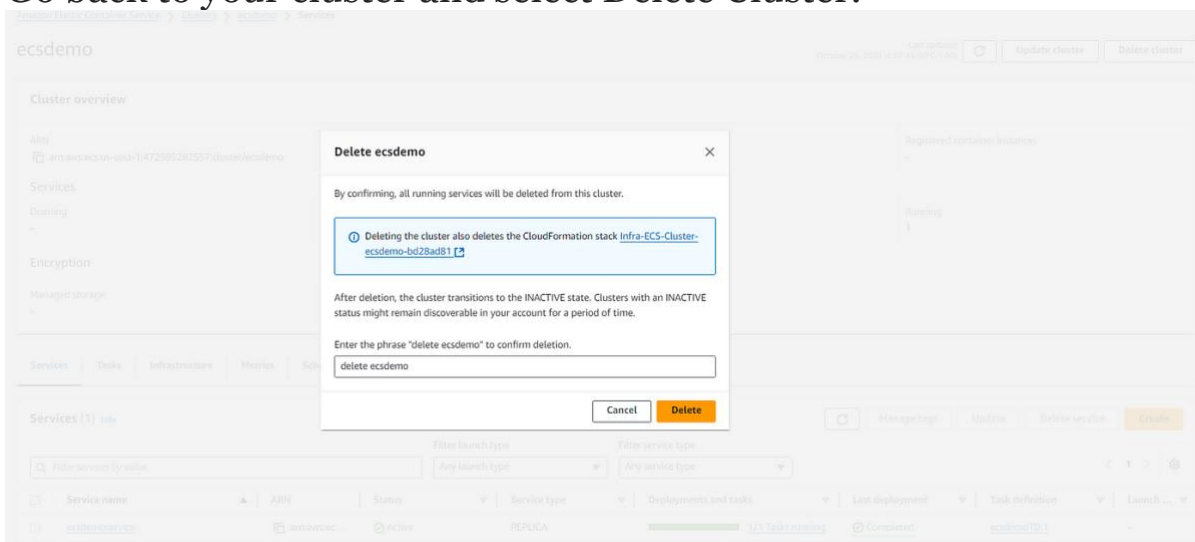
Copy the public IP into your clipboard, and paste it into a browser. You should see this:



You can play the game however long as you wish, but, you should delete your changes as soon as possible to keep your AWS bill low.



Go back to your cluster and select Delete Cluster.



Follow the instructions and click Delete. This concludes the demo, congrats!

Stopping the cluster will prevent you from being further charged. You don't need to delete your Task Definition because it won't charge you. However, you may do so if you desire.

Conclusion

Following through this article, you have understood the basics of Docker containerization, the need for container orchestration, and a practical understanding of getting started with Amazon ECS.