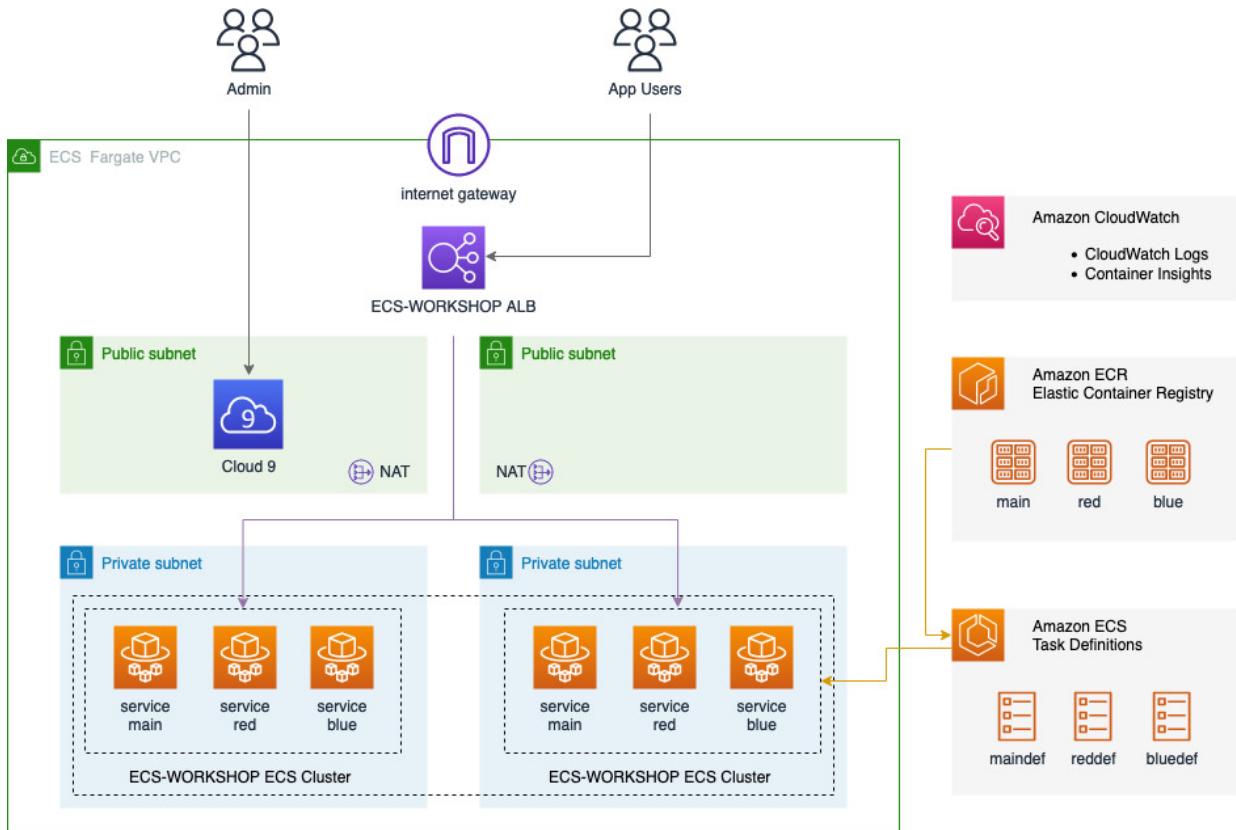


ECS Fargate Immersion Day Workshop

Welcome to the ECS Fargate immersion day. This workshop provides you access to a free AWS account for up to 72 hours and step by step instructions to set up and configure an ECS Fargate cluster and the Blue Red application. The workshop sets up Amazon Container Insights to provide an overview of monitoring and observability capabilities. The intent of this workshop is to provide you a basic setup of an application running on ECS Fargate which you can then use to add and more advance functionalities of ECS Fargate to build up your knowledge.

This workshop is powered by **AWS Event Engine**. Event Engine works best on a personal laptop or PC.

Workshop Introduction



Before beginning the workshop, let's walk through the architecture for this workshop. We will be building the **Blue Red** application and we will use [AWS CloudFormation](#) to automatically deploy a [Virtual Private Cloud](#), and the required public/private subnets and route tables. The majority of this workshop will be done manually **through the AWS Console**, but we will be building Docker images using [AWS Cloud9](#), which is a cloud-based integrated development environment (IDE) that lets you write, run, and debug your code with just a browser.

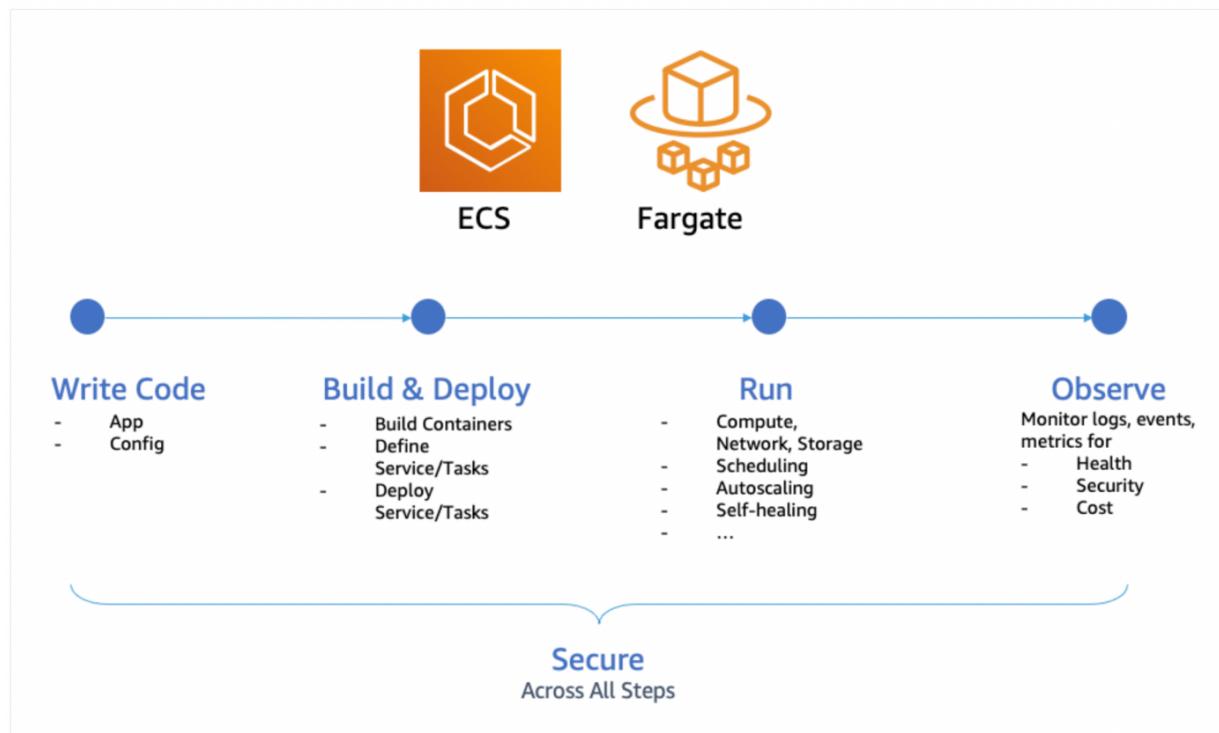
The **Blue Red** application consists of three services: **Main** (landing page), **Blue** (blue page) and **Red** (red page). In this lab, we will use the *AWS Fargate* launch type on a single cluster which we will name **ECS-WORKSHOP**.

Fargate Instances with ECS

AWS Fargate Overview

[AWS Fargate](#) is a technology that you can use with Amazon ECS to run containers without having to manage servers or clusters of Amazon EC2 instances. With Fargate, you no longer have to provision, configure, or scale clusters of virtual machines to run containers. This removes the need to choose server types, decide when to scale your clusters, or optimize cluster packing. When you run your Amazon ECS tasks and services with the Fargate launch type or a Fargate capacity provider, you package your application in containers, specify the Operating System, CPU and memory requirements, define networking and IAM policies, and launch the application. Each Fargate task has its own isolation boundary and does not share the underlying kernel, CPU resources, memory resources, or elastic network interface with another task.

AWS Fargate is built into Amazon ECS. Just define your application's requirements, select Fargate as your launch type in the console or Command Line Interface (CLI), and Fargate takes care of all the scaling and infrastructure management required to run your containers.



Logging into AWS Event Engine Portal

To start your immersion day event, you will need the **Participant Hash** provided upon entry, and your email address to track your unique session.

Connect to the event engine portal by clicking the button or browsing to <https://dashboard.eventengine.run/>. **Note:** You may have been given a URL which includes a has code so you might not see this screen. If you see a screen to login, scroll down to continue with these instructions.

Terms & Conditions

1. By using the Event Engine for the relevant event, you agree to the [AWS Event Terms and Conditions](#) and the [AWS Acceptable Use Policy](#). You acknowledge and agree that are using an AWS-owned account that you can only access for the duration of the relevant event. If you find residual resources or materials in the AWS-owned account, you will make us aware and cease use of the account. AWS reserves the right to terminate the account and delete the contents at any time.

2. You will not: (a) process or run any operation on any data other than test data sets or lab-approved materials by AWS, and (b) copy, import, export or otherwise create derivative works of materials provided by AWS, including but not limited to, data sets.

3. AWS is under no obligation to enable the transmission of your materials through Event Engine and may, in its discretion, edit, block, refuse to post, or remove your materials at any time.

4. Your use of the Event Engine will comply with these terms and all applicable laws, and your access to Event Engine will immediately and automatically terminate if you do not comply with any of these terms or conditions.

Enter your event hash
A 12 or 16 digit hash that was given to you for this event or for a specific team

You will be taken to a screen where they can login with an Email One Time Password (OTP), and thereby get access to the Event Page:

Sign in with
Pick the sign-in method you prefer

Email One-Time Password (OTP)
Enter your personal or corporate email to receive a one-time password

Login with Amazon
Login with your Amazon.com retail account

Amazon Employee
(For Amazon Employees Only) Login with your Amazon Corporate account

[Get help signing in](#)

Click on Email One-Time Password (OTP)

One-time email passcode

Send a passcode to the email below.

Email

[Get help signing in](#)

You can use a personal email address if you are running this workshop on your personal device or use your JPMC email if running on a VDI. Once you have logged in with your one time password, click on **AWS Console**.

Team Dashboard

Event

Event: test_event
 Team Name: (Team Name Not Set Yet)

Event ID:
 Team ID:

Click on **Open Console**, as shown in the highlighted red box below.

AWS Console Login

Use the region specified in the AWS workshop you are running or as instructed by your operator

Login Link

Credentials / CLI Snippets

Mac / Linux Windows

Mac or Linux

```
export AWS_DEFAULT_REGION=us-east-1
export AWS_ACCESS_KEY_ID=
export AWS_SECRET_ACCESS_KEY=
export AWS_SESSION_TOKEN=
```

How do I use the AWS CLI?
 Checkout the AWS CLI documentation here: <https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html>

The AWS Console should open up in your browser. Leave this window open and proceed to next step, preparing for the workshop.

Preparing for the workshop

All source files for CloudFormation and Docker can be found here: <https://github.com/olileach/ecs-workshop>

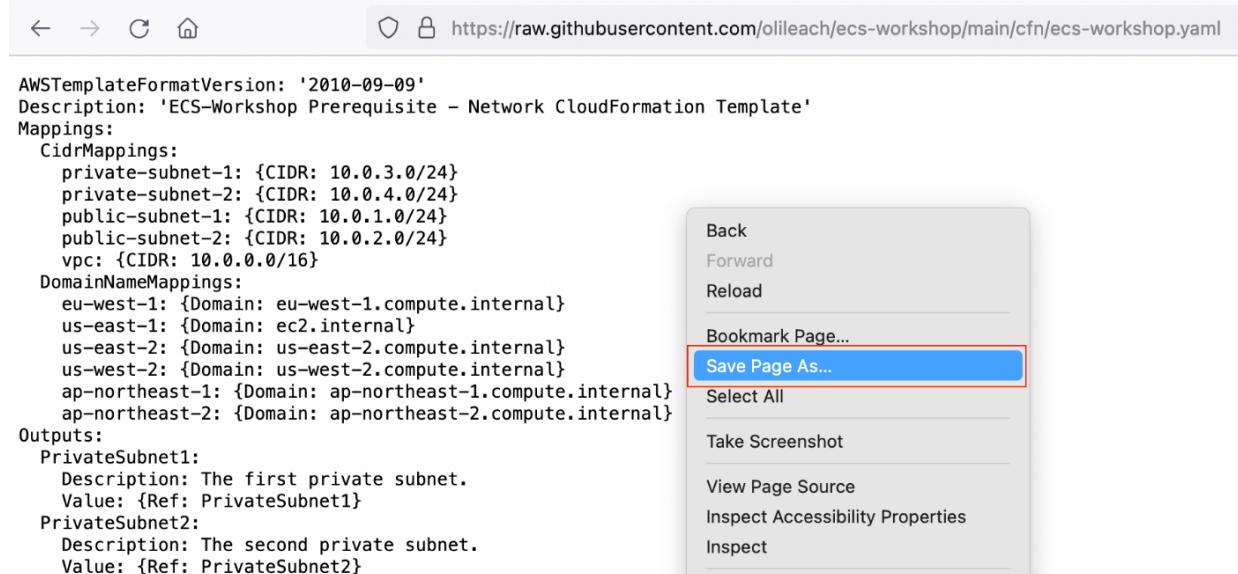
CloudFormation Stack Deployment

Note: This workshop is designed for compatibility in the following regions: **us-east-1 (N.Virginia), us-east-2 (Ohio), us-west-2 (Oregon). For the purpose of this workshop, we will use us-east-1 (N.Virginia).**

1) Download the ECS Workshop CloudFormation template. This CloudFormation template sets up the VPC and Cloud9 environment. You can do download the CloudFormation template a few different ways and it depends if you are using MAC OSX or Windows. The example below uses curl in a terminal or command prompt. Make sure you know where you have downloaded your file.

```
curl -LJO \
https://raw.githubusercontent.com/olileach/ecs-workshop/main/cfn/ecs-workshop.yaml
```

Alternatively, right click on this link <https://raw.githubusercontent.com/olileach/ecs-workshop/main/cfn/ecs-workshop.yaml> and download the contents to a file called **ecs-workshop.yaml**. Or, click on the link to open the web page, right click on the web page and choose **Save Page As...**



2) Make sure you are **logged on to the Event Engine AWS Console**. If you are not, follow the previous steps to open an AWS Console tab in your browser.

3) Click on the following link to launch the CloudFormation template <https://us-east-1.console.aws.amazon.com/cloudformation/home?region=us-east-1>. You will be redirected to **Amazon CloudFormation**.

Click on choose file and browse to the **ecs-workshop.yaml** file you downloaded in step 1

Create stack

Prerequisite - Prepare template

Prepare template

Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

Template is ready

Use a sample template

Create template in Designer

Specify template

A template is a JSON or YAML file that describes your stack's resources and properties.

Template source

Selecting a template generates an Amazon S3 URL where it will be stored.

Amazon S3 URL

Upload a template file

Upload a template file

Choose file

No file chosen

JSON or YAML formatted file

4) Name the stack **ecs-workshop**

Specify stack details

Stack name

Stack name

ecs-workshop

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

5) Click next all the way to the end of the CloudFormation stack setup. At the last section, click on **I acknowledge that AWS CloudFormation might create IAM resources** check box



The following resource(s) require capabilities: [AWS::IAM::Role]

This template contains Identity and Access Management (IAM) resources that might provide entities access to make changes to your AWS account. Check that you want to create each of these resources and that they have the minimum required permissions. [Learn more](#)

I acknowledge that AWS CloudFormation might create IAM resources.

Wait for the CloudFormation template deployment to complete before moving on.



Immersion Day Environment Setup

During the initial CloudFormation setup, the VPC, Subnets, Cloud9 and the ECR **main**, **blue** and **red** repositories with images are all setup for you. Let's check what's been created for you

- 1) Open up the AWS Console and browse to the Cloud9 service. Once in the Cloud9 AWS console page, you should see the **workshop-ide** environment as shown below. Click on **Open** to open up your IDE

A screenshot of the AWS Cloud9 Environments page. The page title is 'AWS Cloud9 > Environments'. It shows a table with one row for the 'workshop-ide' environment. The 'Cloud9 IDE' column contains a link labeled 'Open' which is highlighted with a red box. The table columns are: Name, Cloud9 IDE, Environment type, Connection, Permission, and Owner ARN.

- 2) Once in the IDE, open up a new terminal screen by clicking on the + sign and selecting **New Terminal**. This makes it easier to run commands in a full screen.

A screenshot of the AWS Cloud9 IDE interface. The top navigation bar includes 'File', 'Edit', 'Find', 'View', 'Go', 'Run', 'Tools', 'Window', and 'Support'. A dropdown menu is open over the 'File' icon, showing options like 'New File', 'New Terminal' (which is highlighted with a red box), 'New Run Configuration', etc. On the left, there's a sidebar with tabs for 'Environment' (selected), 'Source Control', and 'AWS'. The main workspace shows a 'Welcome' screen with some code snippets and instructions.

- 3) In the new terminal, run the following command to complete the workshop setup.

```
cd ecsworkshop  
sh setup.sh
```

This should take a couple of minutes and once complete, will output the ECR repositories created during the setup.

- 4) You can confirm that the three repositories and images are in ECR by browsing to the ECR AWS Console. You should see the **latest** tagged image in each repository. Copy and paste the latest image URI into a text editor for reference later.

main						
View push commands Edit						
Images (1)						
<input type="text"/> Find images						
	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
<input type="checkbox"/>	latest	Image	January 07, 2023, 21:09:22 (UTC-00)	139.41	Copy URI	sha256:23f65cb0eaf8316... Complete

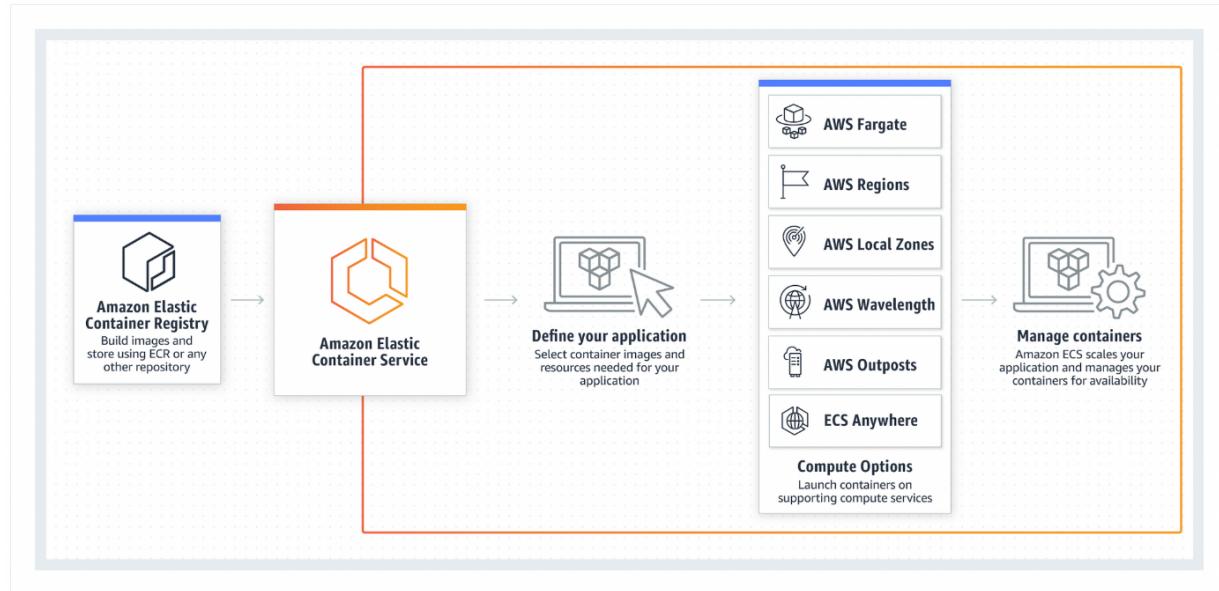
blue						
View push commands Edit						
Images (1)						
<input type="text"/> Find images						
	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
<input type="checkbox"/>	latest	Image	January 07, 2023, 21:09:42 (UTC-00)	139.41	Copy URI	sha256:50922da53223f3... Complete

red						
View push commands Edit						
Images (1)						
<input type="text"/> Find images						
	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
<input type="checkbox"/>	latest	Image	January 07, 2023, 21:10:01 (UTC-00)	139.41	Copy URI	sha256:7be0296bf0b8de... Complete

This completes the workshop setup.

Amazon ECS

[Amazon Elastic Container Service \(ECS\)](#) is a highly scalable and fast container management service. You can use it to run, stop, and manage containers on a cluster. With Amazon ECS, your containers are defined in a task definition that you use to run an individual task or task within a service. ECS enables you to launch thousands of containers across the cloud using your preferred continuous integration and delivery (CI/CD) and automation tools, while also integrating seamlessly with AWS management and governance solutions.



- A **service** is a configuration you can use to run and maintain a specified number of tasks simultaneously in a cluster
- A **task definition** is a blueprint for your application (which are then run as tasks on a given service). Task definitions specify various parameters of your application such as memory, cpu, etc

In this part of the workshop, we will create the following:

- An ECS Cluster, which serves as the logical grouping of tasks or services.
- ECS Task Definitions for our main, blue, and red services
- ECS Services which run our main, blue, and red task definitions
- An [Application Load Balancer](#) (ALB) to handle distribution of traffic across our ECS services

2) Navigate to the [Amazon ECS](#) console. If it is your first time running ECS, you will see the **Get started** screen with an ECS overview video. Click on **Clusters** on the left side navigation bar and **Create Cluster**.

Clusters (0)						Create cluster
<input type="text" value="Search clusters"/>						< 1 > ⚙️
Cluster	Services	Tasks	CloudWatch monitoring	Capacity provider strategy		
No clusters No clusters to display						

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

ecs-workshop

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

- 2) Click on AWS Fargate (serverless) option

▼ Infrastructure Info

Serverless

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.

- 3) Click on **Monitoring - optional** and select **Use Container Insights**, then click on **Create** at the bottom of the page.

▼ Monitoring - optional Info

Container Insights is off by default. When you use Container Insights, there is a cost associated with it.

Use Container Insights

CloudWatch automatically collects metrics for many resources, such as CPU, memory, disk, and network. Container Insights also provides diagnostic information, such as container restart failures, that you use to isolate issues and resolve them quickly. You can also set CloudWatch alarms on metrics that Container Insights collects.

- 4) Refresh the page and soon you will see the **ecs-workshop** cluster you have created.

Clusters (1) Info



Create cluster

< 1 >

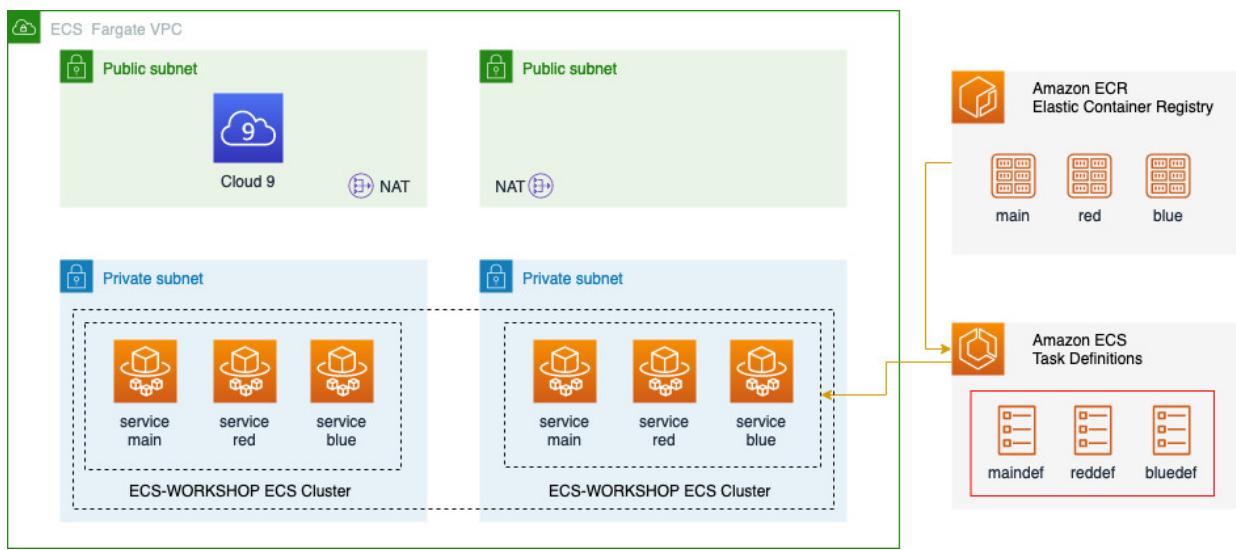
Cluster Services Tasks Registered container instances CloudWatch monitoring Capacity provider strategy

ecs-workshop 0 No tasks running 0 Container Insights No default found

This completes the ECS Fargate cluster setup. Next we will create a task definition for the Red Blue app.

ECS Task Definitions for main, blue and red

In the next section you will create the Task Definitions for our **main**, **blue**, and **red** tasks highlighted in the red box below. Once you have created the **maindef** task definition, you will then repeat the instructions to create the **bluedef** and **reddef** task definitions to complete the setup.



Create the Task Definitions for main, blue and red.

In this section, we will create 3 task definitions for main, blue and red.

Note: To complete the following section, you will need your ECR image URL for **main**, **blue** and **red** containers. To obtain this, browse to ECR repository and take note of the Image Uri. Or you can run the following command in your Cloud9 terminal.

```
aws ecr describe-repositories --query "repositories[*].repositoryUri"
```

The output should look similar to the below example where the account_id matches your AWS account ID and us-east-1 is your active AWS region you are working in.

```
[  
    "<account_id>.dkr.ecr.us-east-1.amazonaws.com/red",  
    "<account_id>.dkr.ecr.us-east-1.amazonaws.com/blue",  
    "<account_id>.dkr.ecr.us-east-1.amazonaws.com/main"  
]
```

- 1) Navigate to the [Amazon ECS](#) console and click on **Task Definitions** in the left hand navigation and then click **Create new Task Definition**.

For Task definition family, use **maindef**, click launch type **AWS Fargate**

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.

▼ 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](#)

2) Configure the resources as follows:

CPU: 0.25

Memory: 0.5

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"/>

▼ 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.

3) Configure the container values as follows:

Container - 1 [Info](#)

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
main	565692740138.dkr.ecr.us-east-1.amazonaws.com/main	Yes

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. Any changes to port mappings configuration impacts the associated service connect settings.

Container port	Protocol	Port name	App protocol	Remove
80	TCP	main-80-tcp	HTTP	

[Add more port mappings](#)

4) For Resource allocation limits - conditional, enter the following values

CPU: 0.25

Memory hard limit: 0.5

Memory soft limit: 0.5

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
0.25 in vCPU	1	0.5 in GB	0.5 in GB

Once done, scroll down and click **Create** to create the task definition. You will see a maindef:1 task has been created.

maindef:1

[Deploy](#) [Actions](#) [Create new revision](#)

Overview [Info](#)

ARN  arn:aws:ecs:us-west-1:565692740138:task-definition/maindef:1	Status  ACTIVE	Time created 2023-09-18T18:17:23.449Z	App environment FARGATE
Task role -	Task execution role ecsTaskExecutionRole	Operating system/Architecture Linux/X86_64	Network mode awsvpc

Create the blue and red task definitions

We'll continue to create the blue and red task definitions in the same way as we did for the main task definitions.

- 1) Navigate to the [Amazon ECS](#) console and click on **Task Definitions** in the left hand navigation and then click **Create new Task Definition**.

For **Task definition family**, use **bluedef**, click launch type **AWS Fargate**

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.

▼ 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](#)

- 2) Configure the resources as follows:

CPU: 0.25

Memory: 0.5

Note - you may have to go into the CPU and Memory drop down to see and configure .25 vCPU and .5 GB

Network mode | [Info](#)

awsVpc

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

CPU	Memory
.25 vCPU	.5 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](#).

None

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

3) Configure the bluedef task definition container values as follows:

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
blue	565692740138.dkr.ecr.us-west-1.amazonaws.com/blue	Yes

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. Any changes to port mappings configuration impacts the associated service connect settings.

Container port	Protocol	Port name	App protocol	Remove
80	TCP	blue-80-tcp	HTTP	<input type="button" value="Remove"/>

4) For Resource allocation limits - conditional, enter the following values

CPU: 0.25

Memory hard limit: 0.5

Memory soft limit: 0.5

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
0.25 in vCPU	1	0.5 in GB	0.5 in GB

Once done, scroll down and click **Create** to create the task definition. You will see a bluedef:1 task has been created.

The screenshot shows the AWS CloudWatch Metrics interface for the 'bluedef:1' task definition. At the top, there are three buttons: 'Deploy ▾', 'Actions ▾', and a yellow 'Create new revision ▾' button. Below this is a table with the following data:

Overview Info	
ARN arn:aws:ecs:us-west-1:565692740138:task-definition/bluedef:1	Status ACTIVE
Task role -	Task execution role ecsTaskExecutionRole
Time created 2023-09-18T18:30:36.421Z	Operating system/Architecture Linux/X86_64
App environment FARGATE	Network mode awsvpc

5) Navigate to the [Amazon ECS](#) console and click on **Task Definitions** in the left hand navigation and then click **Create new Task Definition**.

For **Task definition family**, use **reddef**, click launch type **AWS Fargate**

The screenshot shows the 'Create new task definition' wizard. The first step is 'Task definition configuration'.

Task definition family: reddef (highlighted with a red box)

Infrastructure requirements:

- Launch type:** AWS Fargate (selected)
- Amazon EC2 instances:** (unchecked)

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

Operating system/Architecture: Linux/X86_64

6) Configure the resources as follows:

CPU: 0.25

Memory: 0.5

Note - you may have to go into the CPU and Memory drop down to see and configure .25 vCPU and .5 GB

Network mode | [Info](#)

awsVpc

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

CPU
 Memory

▼ 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.

7) Configure the reddef task definition container values as follows

Container - 1 [Info](#)

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 <input type="text" value="red"/>	Image URI <input type="text" value="565692740138.dkr.ecr.us-west-1.amazonaws.com/red"/>	Essential container <input type="text" value="Yes"/>
--	--	---

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. Any changes to port mappings configuration impacts the associated service connect settings.

Container port <input type="text" value="80"/>	Protocol <input type="text" value="TCP"/>	Port name <input type="text" value="red-80-tcp"/>	App protocol <input type="text" value="HTTP"/>	Remove
---	--	--	---	------------------------

[Add more port mappings](#)

8) For Resource allocation limits - conditional, enter the following values

CPU: 025

Memory hard limit: 0.5

Memory soft limit: 0.5

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 0.25 in vCPU	GPU 1	Memory hard limit 0.5 in GB	Memory soft limit 0.5 in GB
------------------------	----------	-----------------------------------	-----------------------------------

reddef:1

Deploy ▾ Actions ▾ Create new revision ▾

Overview [Info](#)

ARN arn:aws:ecs:us-west-1:56569274 0138:task-definition/reddef:1	Status ACTIVE	Time created 2023-09-18T18:37:46.231Z	App environment FARGATE
Task role -	Task execution role ecsTaskExecutionRole	Operating system/Architecture Linux/X86_64	Network mode awsvpc

Once complete, you should see 3 task definitions for main, blue and red.

[Amazon Elastic Container Service](#) > Task definitions

Task definitions (3) [Info](#) Deploy ▾ Create new revision ▾ Create new task definition ▾

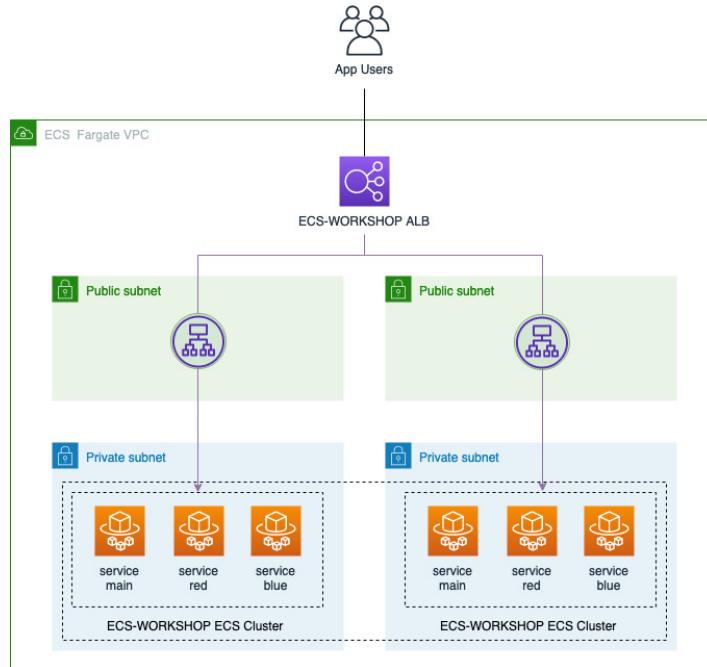
Filter task definitions by property or value 3 matches

Status of last revision = ACTIVE X Clear filters < 1 > ⚙

Task definition	Status of last revision
bluedef	ACTIVE
maindef	ACTIVE
reddef	ACTIVE

Creating the Application Load Balancer for ECS

In this section we will create an Application Load Balancer that will forward and distribute traffic to our **main**, **blue** and **red** web application. Let's recap on the architecture we wish to build for our **Blue Red** application.



We can get ECS to create this load balancer when we create the ECS service but we'll create this ourselves to ensure we have the right security groups and subnet configuration so only our Application Load Balancer is in the public subnet and all our tasks and services run in the private subnet.

- 1) Navigate to the [EC2 Load Balancers](#) console and scroll down until you see the **Load Balancing** section

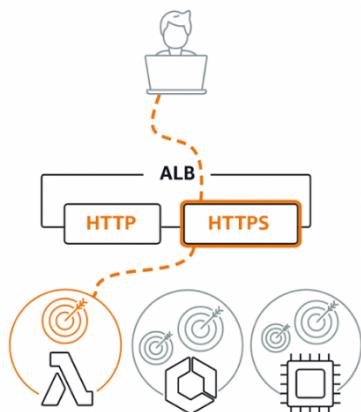
▼ Load Balancing

[Load Balancers](#)

[Target Groups](#)

- 2) Click **Create Load Balancer** and select **Application Load Balancer** and click **Create**

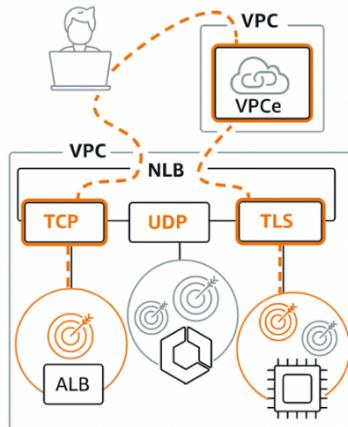
Application Load Balancer [Info](#)



Choose an Application Load Balancer when you need a flexible feature set for your applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including microservices and containers.

[Create](#)

Network Load Balancer [Info](#)



Choose a Network Load Balancer when you need ultra-high performance, TLS offloading at scale, centralized certificate deployment, support for UDP, and static IP addresses for your applications. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second securely while maintaining ultra-low latencies.

[Create](#)

Gateway Load Balancer [Info](#)



Choose a Gateway Load Balancer when you need to deploy and manage a fleet of third-party virtual appliances that support GENEVE. These appliances enable you to improve security, compliance, and policy controls.

[Create](#)

3) On the **Create Application Load Balancer** screen, configure the following

For **Basic configuration**, configure the Load balancer name as **ecs-workshop-alb**. Leave the rest

Basic configuration

Load balancer name

Name must be unique within your AWS account and cannot be changed after the load balancer is created.

ecs-workshop-alb

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

Scheme [Info](#)

Scheme cannot be changed after the load balancer is created.

Internet-facing

An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. [Learn more](#)

Internal

An internal load balancer routes requests from clients to targets using private IP addresses.

IP address type [Info](#)

Select the type of IP addresses that your subnets use.

IPv4

Recommended for internal load balancers.

Dualstack

Includes IPv4 and IPv6 addresses.

4) For **Network mapping**, select **ecs-workshop-vpc**

Network mapping Info

The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.

VPC Info

Select the virtual private cloud (VPC) for your targets. Only VPCs with an internet gateway are enabled for selection. The selected VPC cannot be changed after the load balancer is created. To confirm the VPC for your targets, view your [target groups](#).

ecs-workshop-vpc
vpc-0584169744a147e3f
IPv4: 10.0.0.0/16



For **Mappings** select two Availability Zones and use the two **PublicSubnet1** and **PublicSubnet2** subnets.

us-east-1a (use1-az2)

Subnet

subnet-07150aeef04551afe

PublicSubnet1 ▾

IPv4 settings

Assigned by AWS

us-east-1b (use1-az4)

Subnet

subnet-08834ede0e2a51803

PublicSubnet2 ▾

IPv4 settings

Assigned by AWS

5) For **security groups**, select the security group that starts with **ecs-workshop-ALBSG**, for example **ecs-workshop-ALBSG-HNNPZQBUZ5JT**. **Note:** Remove any other Security Group

Security groups Info

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

Security groups

Select up to 5 security groups



Create new security group [\[\]](#)

ecs-workshop-ALBSG-HNNPZQBUZ5JT sg-0b332826bdff56a65 X
VPC: vpc-0584169744a147e3f

6) For **Listeners and routing** click on [Create target group](#)

Listeners and routing [Info](#)

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

▼ Listener HTTP:80 Remove

Protocol	Port	Default action	Info
HTTP	: 80	Forward to	Select a target group
1-65535			
Create target group			

Note: A new tab will open so you can configure a target group

7) Under **Basic configuration**, select **IP address** and type **main** as the **Target Group name**

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

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

Scroll down and click **Next** to navigate to the next screen

8) Click **Remove** to remove the IPv4 address. These addresses are automatically added by ECS so we don't need anything configured here.

Step 2: Specify IPs and define ports
You can manually enter IP addresses from the selected network.

IPv4 address [Remove](#)

Scroll down and click on Create Target Group

Create target group

You should see the Target Group available.

Target groups (1) Info					
Actions Create target group					
<input type="text"/> Search or filter target groups					
<input type="checkbox"/>	Name	ARN	Port	Protocol	Target type
<input type="checkbox"/>	main	arn:aws:elasticloadbalancing:us-east-1:	80	HTTP	IP

9) Now, go back to the tab where you were creating the **Application Load Balancer** and click on the refresh icon to refresh the Target Groups available.

Protocol Port Default action [Info](#)

HTTP : 80 1-65535

Forward to [Select a target group](#) [Create target group](#) [G](#)

⚠ You must select at least one target group.

Once refreshed, you can select the **main target group** you created.

Protocol Port Default action [Info](#)

HTTP : 80 1-65535

Forward to **main** Target type: IP, IPv4 [HTTP](#) [G](#)

[Create target group](#)

Once the **Target Group** is selected, scroll down to the bottom and click on **Create load balancer**.

Create load balancer

10) After a while, the Application Load Balancer status should be **Active** as shown below

Load balancers (1)					
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.					
Actions Create load balancer					
<input type="text"/> Filter by property or value					
<input type="text"/> search: ecs-workshop-alb X Clear filters					
<input type="checkbox"/>	Name	DNS name	State	VPC ID	Availability Zones
<input type="checkbox"/>	ecs-workshop-alb	ecs-workshop-alb-476857044.us-east-1.elb.amazonaws.com	Active	vpc-0584169744a147e3f	2 Availability Zones

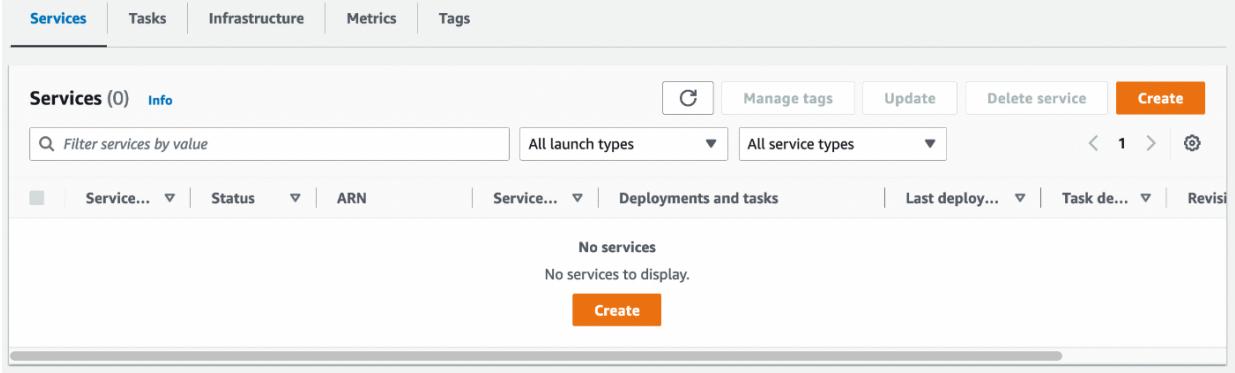
Creating the ECS Services for the Blue Red App

A quick recap before we start the next section

- An **ECS service** is a configuration you can use to run and maintain a specified number of tasks simultaneously in a cluster
- An **ECS task definition** is a blueprint for your application (which are then run as tasks on a given service). Task definitions specify various parameters of your application such as memory, cpu, etc

In the next section you will create the ECS service for **main**, **blue** and **red** that will run tasks based on the task definitions that created in a previous section.

- 1) Navigate to the ECS console and click on the **ecs-workshop** ECS cluster, navigating your way to the Service section as shown below.



The screenshot shows the AWS ECS Services page. At the top, there are tabs for Services, Tasks, Infrastructure, Metrics, and Tags. The Services tab is selected. Below the tabs, there's a search bar labeled "Filter services by value" and several dropdown filters: "All launch types" and "All service types". To the right of these filters are buttons for "Create", "Manage tags", "Update", and "Delete service". Further right are navigation arrows and a refresh icon. The main content area is titled "Services (0)" and contains a table header with columns: Service..., Status, ARN, Service..., Deployments and tasks, Last deploy..., Task de..., and Revision. Below the table, a message says "No services" and "No services to display." A prominent orange "Create" button is located at the bottom center of the page.

- 2) Click on Create and select **Launch type** and make sure **FARGATE** is selected as shown below.

Create Info

Environment

AWS Fargate

Existing cluster
Select an existing cluster. To create a new cluster, go to [Clusters](#).

ecs-workshop

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.

Launch type Info
Select either managed capacity (Fargate), or custom capacity (EC2 or user-managed, External instances). External instances are registered to your cluster using the ECS Anywhere capability.

FARGATE ▾

Platform version Info
Specify the platform version on which to run your service.

LATEST ▾

3) Scroll down to the **Deployment configuration** and add select **Service**, **Family** as **maindef** and the **Desired tasks** as **2**.

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

maindef

Revision

1 (LATEST)

Service name

Assign a unique name for this service.

main

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.

2

- 4) Configure the network as shown below. For the VPC, select the **ecs-workshop-vpc** and the subnets, select **PrivateSubnet1** and **PrivateSubnet2**. Use the existing security group starting with **ecs-workshop-ALB** and make sure **Public IP is Turned off**.

▼ Networking

VPC | [Info](#)

Choose the Virtual Private Cloud to use.

vpc-0843f0506d050d267
ecs-workshop-vpc

Subnets

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

Choose subnets

subnet-012bae79dcc694ee5 X
us-east-1a | PrivateSubnet1

subnet-04fdee2b939a8a230 X
us-east-1b | PrivateSubnet2

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.

sg-0e14b058513f774ba X
ecs-workshop-ALBSG-RTVPJ7NDXB8E

Public IP | [Info](#)

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

Turned off

- 5) Under **Load balancing**, for the **Load balancer** type choose **Application Load Balancer** and configure **ecs-workshop-alb** as the **Load balancer name**. Make sure the **main 80:80** is selected as the container to load balance. Check the **Listener** is configured to use port **80**. Use an existing **Target group** name, select **main** and configure the **Health check grace period** as **15**

▼ Load balancing - optional

Load balancer type | [Info](#)

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

Application Load Balancer

Application Load Balancer

Specify whether to create a new load balancer or choose an existing one.

- Create a new load balancer
- Use an existing load balancer

Load balancer

Select the load balancer you wish to use to distribute incoming traffic across the tasks running in your service.

ecs-workshop-alb

Choose container to load balance

main 80:80

Listener | [Info](#)

Specify the port and protocol that the load balancer will listen for connection requests on.

Port

80

Protocol

HTTP

6) For **Target group name** select **Use an existing target group**, name the **Target group name** as **main**.

Target group | [Info](#)

Specify whether to create a new target group or choose an existing one that the load balancer will use to route requests to the tasks in your service.

- Create new target group
- Use an existing target group

Target group name

main

Health check path

/

Health check protocol

HTTP

Health check grace period | [Info](#)

15

seconds

7) Leave the **Service auto scaling - optional** checkbox as empty and click **Create**.

▼ Service auto scaling - optional

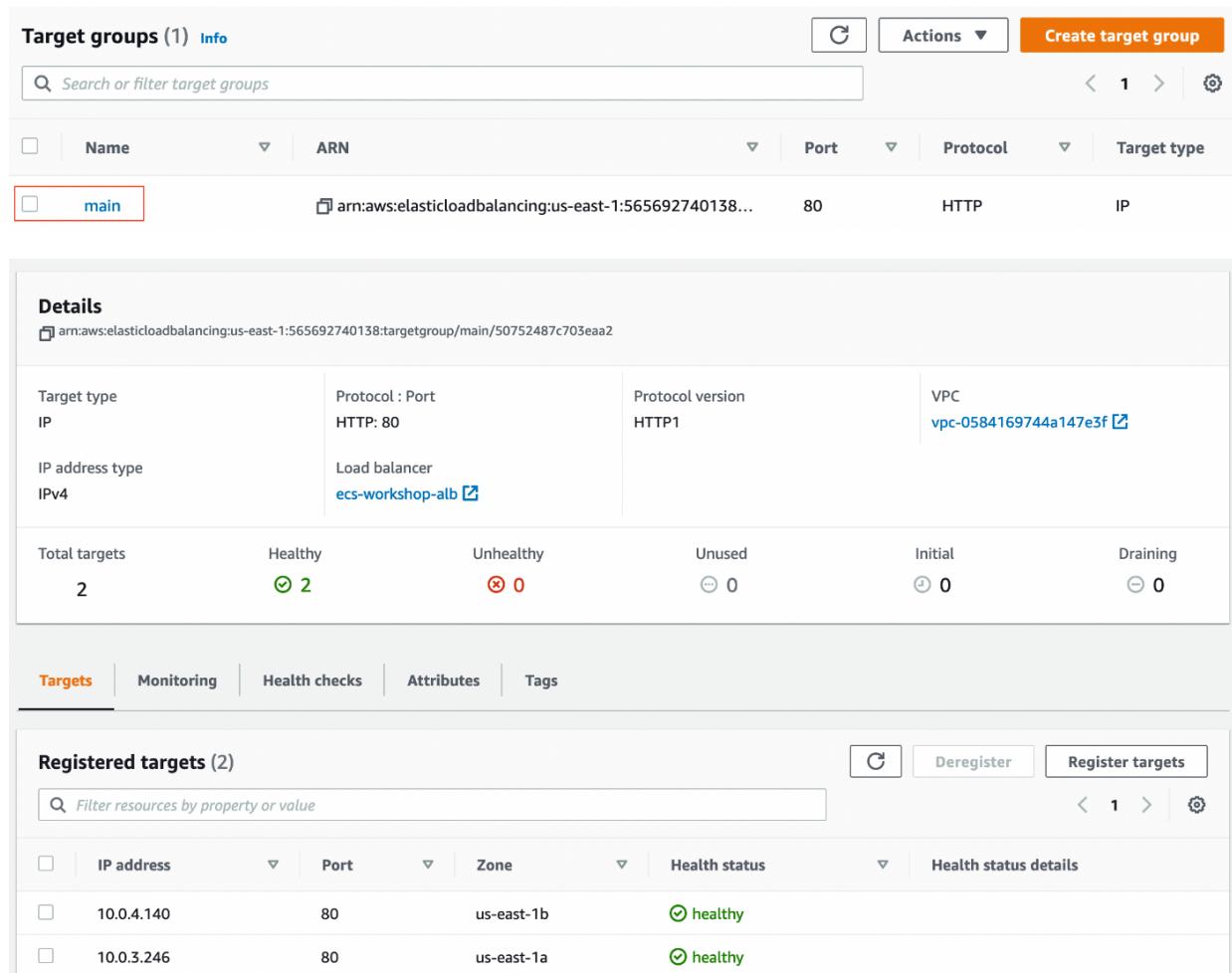
Automatically adjust your service's desired count up and down within a specified range in response to CloudWatch alarms. You can modify your service auto scaling configuration at any time to meet the needs of your application.

Use service auto scaling

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

⌚ main deployment is in progress. It takes a few minutes.

You can also check the Registered targets **in the Application Load Balancer** to make sure the targets have registered successfully. Navigate to the EC2 console and click on Target Groups. Click on the **main** Target Group and check the Targets are **Healthy**.



The screenshot shows the AWS Lambda console with the 'Service auto scaling' section. A blue banner at the top indicates a 'main deployment is in progress. It takes a few minutes.' Below this, there is a checkbox labeled 'Use service auto scaling' with a descriptive text below it. The main content area displays a table of target groups, with one entry for 'main' highlighted. The 'Details' tab for the 'main' target group is selected, showing the following information:

Target type	Protocol : Port	Protocol version	VPC
IP	HTTP: 80	HTTP1	vpc-0584169744a147e3f
IP address type	Load balancer		
IPv4	ecs-workshop-alb		

Below this, a summary table shows the status of targets:

Total targets	Healthy	Unhealthy	Unused	Initial	Draining
2	2	0	0	0	0

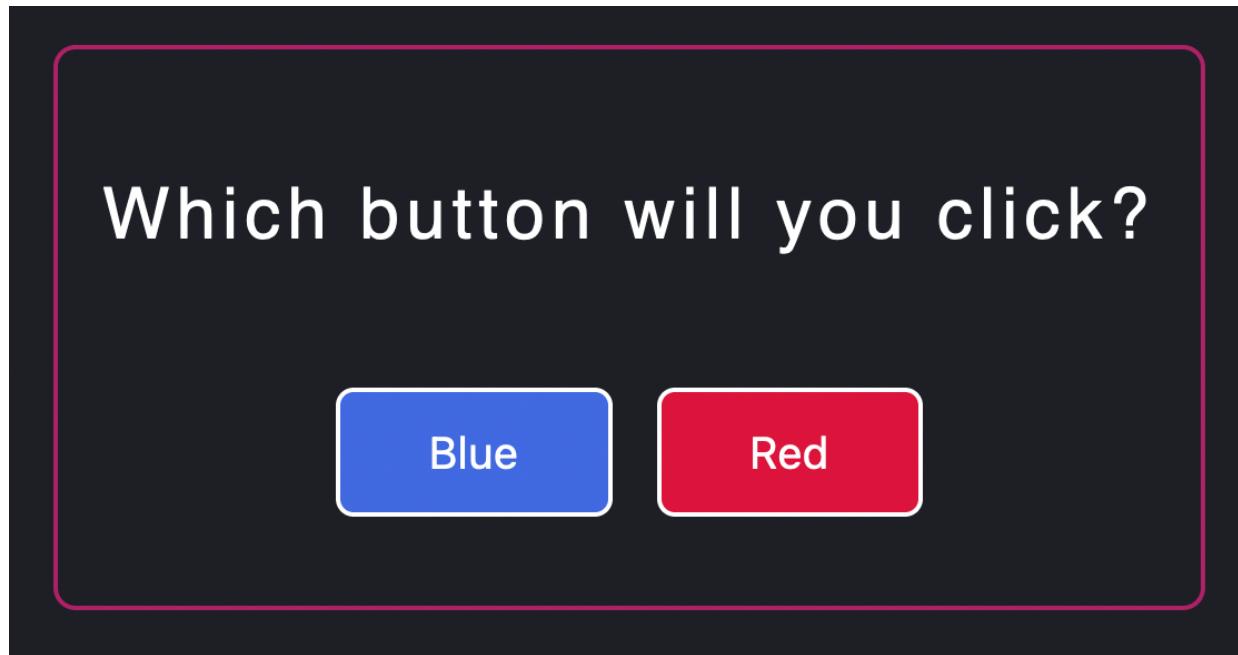
At the bottom, tabs for 'Targets', 'Monitoring', 'Health checks', 'Attributes', and 'Tags' are visible, with 'Targets' being the active tab. The 'Registered targets' section shows two entries:

IP address	Port	Zone	Health status	Health status details
10.0.4.140	80	us-east-1b	healthy	
10.0.3.246	80	us-east-1a	healthy	

Whilst still in the EC2 console, click on Load Balancers and click on the **ecs-workshop-alb** load balancer. Click and copy the **DNS name** and copy the name into a new browser tab. Note this DNS name down as we'll use it later.

ecs-workshop-alb			
▼ Details C Actions ▾			
<small>arn:aws:elasticloadbalancing:us-east-1:565692740138:loadbalancer/app/ecs-workshop-alb/87b3d960665d036f</small>			
Load balancer type Application	DNS name ecs-workshop-alb-476857044.us-east-1.elb.amazonaws.com (A Record)	Status Active	VPC vpc-0584169744a147e3f
IP address type IPv4	Scheme Internet-facing	Availability Zones subnet-07150aeef04551afe us-east-1a (use1-az2) subnet-08834ede0e2a51803 us-east-1b (use1-az4)	Hosted Zone Z35SXDOTRQ7X7K
Created At January 7, 2023, 23:10 (UTC+00:00)			

The following page should load in the browser.



Congratulations! This completes setting up the main ECS service.

Create the Blue Service

Now we need to set up the **blue** service. We'll reuse the same steps to create the **red** service once we've completed the **blue** service setup.

- 1) Navigate to the [Amazon ECS](#) console and select **ECS-WORKSHOP** cluster. Go to **Services** tab and click **Create**.

The screenshot shows the AWS ECS Services page. At the top, there are buttons for 'Create', 'Manage tags', 'Update', and 'Delete service'. Below that is a search bar labeled 'Filter services by value' and dropdown menus for 'All launch types' and 'All service types'. The main table lists one service: 'main' (Status: Active, ARN: arn:aws:ecs..., Deployment: REPLICA, Last deployment: Completed). A progress bar indicates 2/2 Tasks running.

2) Click on Create and select **Launch type** and make sure **FARGATE** is selected as shown below.

For the **Launch Type**, click on **Fargate**

The screenshot shows the 'Compute configuration (advanced)' section. It includes fields for 'Existing cluster' (set to 'ecs-workshop'), 'Compute options' (set to 'FARGATE'), and 'Platform version' (set to 'LATEST'). The 'Launch type' section is highlighted with a red border, showing 'FARGATE' selected. Other options like 'Capacity provider strategy' are also shown.

3) Scroll down to the **Deployment configuration** and add select **Service**, **Family** as **bluedef**, the **Service name** as **blue** and the **Desired tasks** as **2**.

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

bluedef

Revision

1 (LATEST)

Service name

Assign a unique name for this service.

blue

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.

2

- 4) Configure the network as shown below. For the VPC, select the **ecs-workshop-vpc** and the subnets, select **PrivateSubnet1** and **PrivateSubnet2**. Use the existing security group starting with **ecs-workshop-ALB** and make sure **Public IP is Turned off**.

▼ Networking

VPC | [Info](#)

Choose the Virtual Private Cloud to use.

vpc-0843f0506d050d267

ecs-workshop-vpc

Subnets

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

Choose subnets

subnet-012bae79dcc694ee5 X
us-east-1a | PrivateSubnet1

subnet-04fdee2b939a8a230 X
us-east-1b | PrivateSubnet2

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.

sg-0e14b058513f774ba X
ecs-workshop-ALBSG-RTVPJ7NDXB8E

Public IP | [Info](#)

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

Turned off

- 5) Under **Load balancing**, for the **Load balancer** type choose **Application Load Balancer** and configure **ecs-workshop-alb** as the **Load balancer name**. Make sure the **blue** **80:80** is selected as the container to load balance. Check the **Listener** is configured to use port **80**. Use an existing **Target group** name, select **blue** and configure the **Health check grace period** as **15**

▼ Load balancing - optional

Load balancer type | [Info](#)

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

Application Load Balancer

Application Load Balancer

Specify whether to create a new load balancer or choose an existing one.

- Create a new load balancer
- Use an existing load balancer

Load balancer

Select the load balancer you wish to use to distribute incoming traffic across the tasks running in your service.

ecs-workshop-alb

Choose container to load balance

blue 80:80

Listener | [Info](#)

Specify the port and protocol that the load balancer will listen for connection requests on.

- Create new listener
- Use an existing listener

Listener

80:HTTP

6) For **Target group name** select **create new** name the **Target group name** as **blue**, **Path pattern** as **/blue/** the **Evaluation order** as **1**, the **Health check path** as **/blue/index.html** and the **Health check grace period** as **15**.

IMPORTANT NOTE: You must configure the **Path pattern** as **/blue/** and **Health check path** as **/blue/index.html** otherwise the checks will fail

Target group | [Info](#)

Specify whether to create a new target group or choose an existing one that the load balancer will use to route requests to the tasks in your service.

- Create new target group
- Use an existing target group

Target group name

blue

Protocol

HTTP

Path pattern

/blue/

Evaluation order

1

Max size: 128 characters.

1 - 50000

< 1 >

Evaluation order

▼

Rule path

▼

Target group

▼

default

/

main

Health check path | [Info](#)

/blue/index.html

Health check protocol

HTTP

Health check grace period | [Info](#)

15

▼

seconds

7) Leave the **Service auto scaling - optional** checkbox as empty and click **Create**.

▼ Service auto scaling - optional

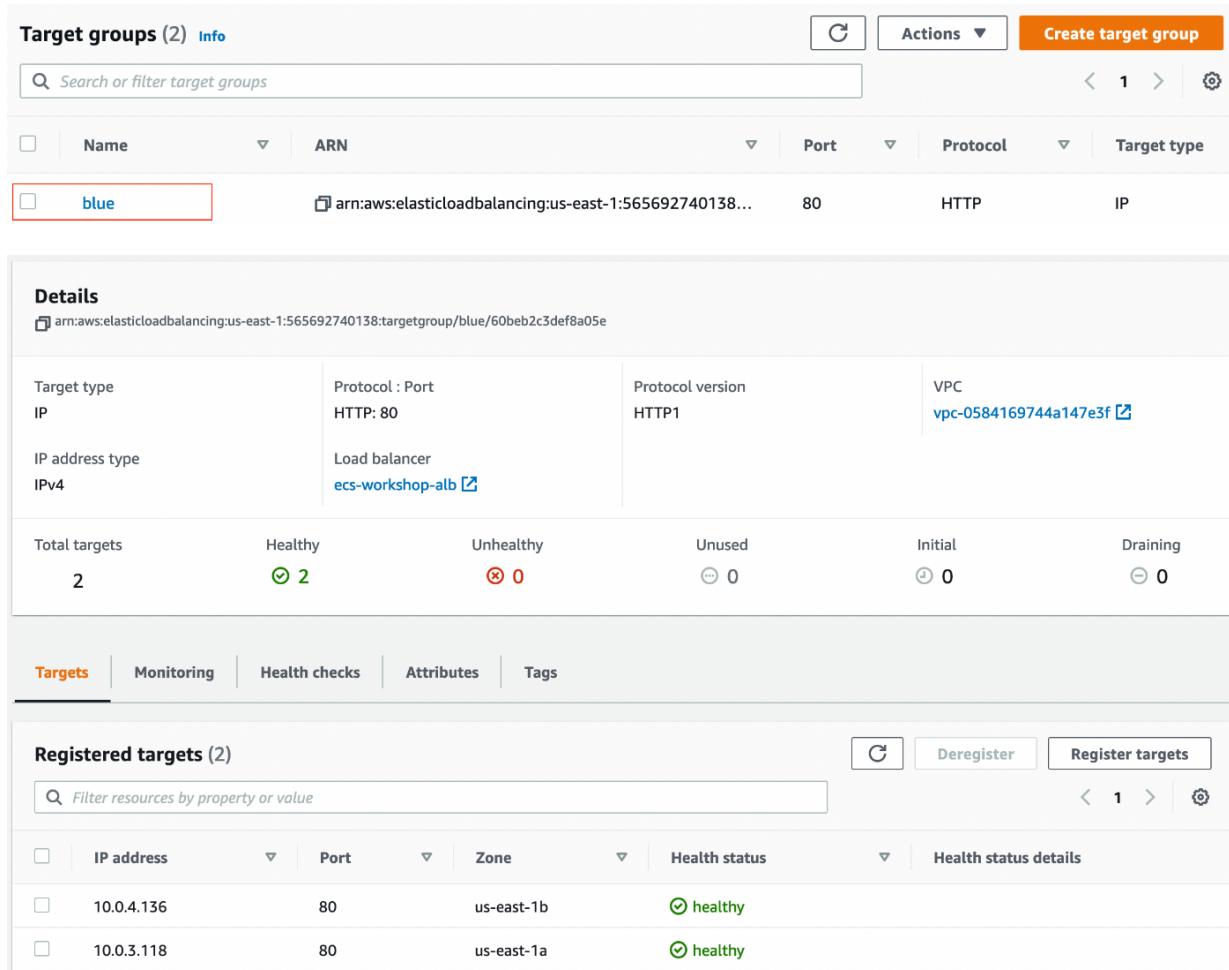
Automatically adjust your service's desired count up and down within a specified range in response to CloudWatch alarms. You can modify your service auto scaling configuration at any time to meet the needs of your application.

Use service auto scaling

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

 main deployment is in progress. It takes a few minutes.

As you did previously for the **main** service, you can also check the Registered targets in the Application Load Balancer to make sure the **blue** targets have registered successfully. Navigate to the **Load Balancers** in the EC2 console and click on **Target Groups**. Click on the **blue** Target Group and check the Targets are **Healthy**.



The screenshot shows the AWS Lambda console with the 'Targets' tab selected for the 'blue' target group. The table displays the following information:

Name	ARN	Port	Protocol	Target type
blue	arn:aws:elasticloadbalancing:us-east-1:565692740138:targetgroup/blue/60beb2c3def8a05e	80	HTTP	IP

Details section:

Target type IP	Protocol : Port HTTP: 80	Protocol version HTTP1	VPC vpc-0584169744a147e3f		
IP address type IPv4	Load balancer ecs-workshop-alb				
Total targets 2	Healthy 	Unhealthy 	Unused 	Initial 	Draining

Registered targets (2) section:

IP address	Port	Zone	Health status	Health status details
10.0.4.136	80	us-east-1b	healthy	
10.0.3.118	80	us-east-1a	healthy	

Now the **blue** service is up and running and the Health status is **healthy**, we need to repeat the steps for the **red** service.

Create the Red Service

Now we need to set up the **blue** service. We'll reuse the same steps to create the **red** service once we've completed the **blue** service setup.

- 1) Navigate to the [Amazon ECS](#) console and select **ECS-WORKSHOP** cluster. Go to **Services** tab and click **Create**.

Services (2) Info						
C		Manage tags		Update		Create
<input type="text"/> Filter services by value		All launch types ▼		All service types ▼		◀ 1 ▶ ⚙️
Service...	Status ▼	ARN	Service... ▼	Deployments and tasks	Last deployment	▼
<input type="checkbox"/> main	Active	arn:aws:ecs...	REPLICA	<div style="width: 100%;">2/2 Tasks running</div>	Completed	⚙️
<input type="checkbox"/> blue	Active	arn:aws:ecs...	REPLICA	<div style="width: 100%;">2/2 Tasks running</div>	Completed	⚙️

2) Click on Create and select **Launch type** and make sure **FARGATE** is selected as shown below.

For the **Launch Type**, click on **Fargate**

Environment

AWS Fargate

Existing cluster

Select an existing cluster. To create a new cluster, go to [Clusters](#).

ecs-workshop

▼ 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.

Launch type [Info](#)

Select either managed capacity (Fargate), or custom capacity (EC2 or user-managed, External instances). External instances are registered to your cluster using the ECS Anywhere capability.

FARGATE [▼](#)

Platform version [Info](#)

Specify the platform version on which to run your service.

LATEST [▼](#)

3) Scroll down to the **Deployment configuration** and add select **Service**, **Family** as **reddef**, the **Service name** as **red** and the **Desired tasks** as **2**.

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

reddef

Revision

1 (LATEST)

Service name

Assign a unique name for this service.

red

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.

2

- 4) Configure the network as shown below. For the VPC, select the **ecs-workshop-vpc** and the subnets, select **PrivateSubnet1** and **PrivateSubnet2**. Use the existing security group starting with **ecs-workshop-ALB** and make sure **Public IP is Turned off**.

▼ Networking

VPC | [Info](#)

Choose the Virtual Private Cloud to use.

vpc-0843f0506d050d267
ecs-workshop-vpc

Subnets

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

Choose subnets

subnet-012bae79dcc694ee5 X
us-east-1a | PrivateSubnet1

subnet-04fdee2b939a8a230 X
us-east-1b | PrivateSubnet2

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.

sg-0e14b058513f774ba X
ecs-workshop-ALBSG-RTVPJ7NDXB8E

Public IP | [Info](#)

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

Turned off

- 5) Under **Load balancing**, for the **Load balancer** type choose **Application Load Balancer** and configure **ecs-workshop-alb** as the **Load balancer name**. Make sure the **red 80:80** is selected as the container to load balance. Check the **Listener** is configured to use port **80**. Use an existing **Target group** name, select **red** and configure the **Health check grace period** as **15**

▼ Load balancing - optional

Load balancer type | [Info](#)

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

Application Load Balancer

Application Load Balancer

Specify whether to create a new load balancer or choose an existing one.

- Create a new load balancer
- Use an existing load balancer

Load balancer

Select the load balancer you wish to use to distribute incoming traffic across the tasks running in your service.

ecs-workshop-alb

Choose container to load balance

red 80:80

Listener | [Info](#)

Specify the port and protocol that the load balancer will listen for connection requests on.

- Create new listener
- Use an existing listener

Listener

80:HTTP

6) For **Target group name** select **create new** name the **Target group name** as **red**, **Path pattern** as **/red/**, the **Evaluation order** as **2**, the **Health check path** as **/red/index.html** and the **Health check grace period** as **15**.

IMPORTANT NOTE: You must configure the **Path pattern** as **/red/** and health check **/red/index.html** otherwise the checks will fail.

Target group [Info](#)

Specify whether to create a new target group or choose an existing one that the load balancer will use to route requests to the tasks in your service.

Create new target group

Use an existing target group

Target group name

red

Protocol

HTTP

Path pattern

/red/

Evaluation order

1

Max size: 128 characters.

1 - 50000

< 1 >

Evaluation order

Rule path

Target group

default

/

main

Health check path [Info](#)

/red/index.html

Health check protocol

HTTP

Health check grace period [Info](#)

15

seconds

7) Leave the **Service auto scaling - optional** checkbox as empty and click **Create**.

▼ Service 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.

Use service auto scaling

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

⌚ main deployment is in progress. It takes a few minutes.

Once the service has created, check that the 3 services are listed in the **ECS Console**

Services	Tasks	Infrastructure	Metrics	Tags	
Services (3) Info					
<input type="button" value="C"/>	<input type="button" value="Manage tags"/>	<input type="button" value="Update"/>	<input type="button" value="Delete service"/>	<input type="button" value="Create"/>	
<input type="text"/> Filter services by value	All launch types	All service types	< 1 >	<input type="button" value=""/>	
Service...	Status	ARN	Service...	Deployments and tasks	Last deployment
red	Active	arn:aws:ecs...	REPLICA	<div style="width: 100%;">2/2 Tasks running</div>	Completed
main	Active	arn:aws:ecs...	REPLICA	<div style="width: 100%;">2/2 Tasks running</div>	Completed
blue	Active	arn:aws:ecs...	REPLICA	<div style="width: 100%;">2/2 Tasks running</div>	Completed

As you did previously for the **blue** and **main** service, you can also check the Registered targets in the **Application Load Balancer** to make sure the **red** targets have registered successfully. Navigate to the **Load Balancers** in the EC2 console and click on **Target Groups**. Click on the **red** Target Group and check the Targets are **Healthy**.

Target groups (1) Info					
<input type="button" value="C"/> Actions <input type="button" value="Create target group"/> <input type="text"/> Search or filter target groups					
<input type="button" value="search: red"/> <input type="button" value="Clear filters"/>					
Name	ARN	Port	Protocol	Target type	
red	arn:aws:elasticloadbalancing:us-east-1:565692740138:targetgroup/blue/60beb2c3def8a05e	80	HTTP	IP	

Details

arn:aws:elasticloadbalancing:us-east-1:565692740138:targetgroup/blue/60beb2c3def8a05e

Target type IP	Protocol : Port HTTP: 80	Protocol version HTTP1	VPC vpc-0584169744a147e3f		
IP address type IPv4	Load balancer ecs-workshop-alb				
Total targets 2	Healthy ②	Unhealthy ① 0	Unused ① 0	Initial ① 0	Draining ① 0

Targets	Monitoring	Health checks	Attributes	Tags
Registered targets (2)				
<input type="button" value="C"/>	<input type="button" value="Deregister"/>	<input type="button" value="Register targets"/>		
<input type="text"/> Filter resources by property or value	< 1 >	<input type="button" value=""/>		
IP address	Port	Zone	Health status	Health status details
10.0.4.136	80	us-east-1b	healthy	
10.0.3.118	80	us-east-1a	healthy	

This completes the setup for the **main**, **red** and **blue** ECS services. Now we can check if the application is running successfully.

Navigate to the **Load Balancers** in the EC2 console and click on **ecs-workshop-alb** application load balancer to get the DNS name. Click and copy the **DNS name** and copy the name into a new browser tab.

The screenshot shows the AWS Load Balancers console with the following details:

Load balancer type	DNS name	Status	VPC
Application	ecs-workshop-alb-476857044.us-east-1.elb.amazonaws.com (A Record)	Active	vpc-0584169744a147e3f
IP address type	Scheme	Availability Zones	Hosted Zone
IPv4	Internet-facing	subnet-07150aeef04551afe us-east-1a (use1-az2) subnet-08834ede0e2a51803 us-east-1b (use1-az4)	Z35SXDOTRQ7X7K
Created At			
January 7, 2023, 23:10 (UTC+00:00)			

Container Insights

CloudWatch Container Insights to collect, aggregate, and summarize metrics and logs from your containerized applications and microservices. CloudWatch automatically collects metrics for many resources, such as CPU, memory, disk, and network. Container Insights also provides diagnostic information, such as container restart failures, to help you isolate issues and resolve them quickly. You can also set CloudWatch alarms on metrics that Container Insights collects.

See the following link for more information on Container Insights, <https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/ContainerInsights.html>

Container Insights was enabled when we created the ECS cluster. Let's take a look at what is configured.

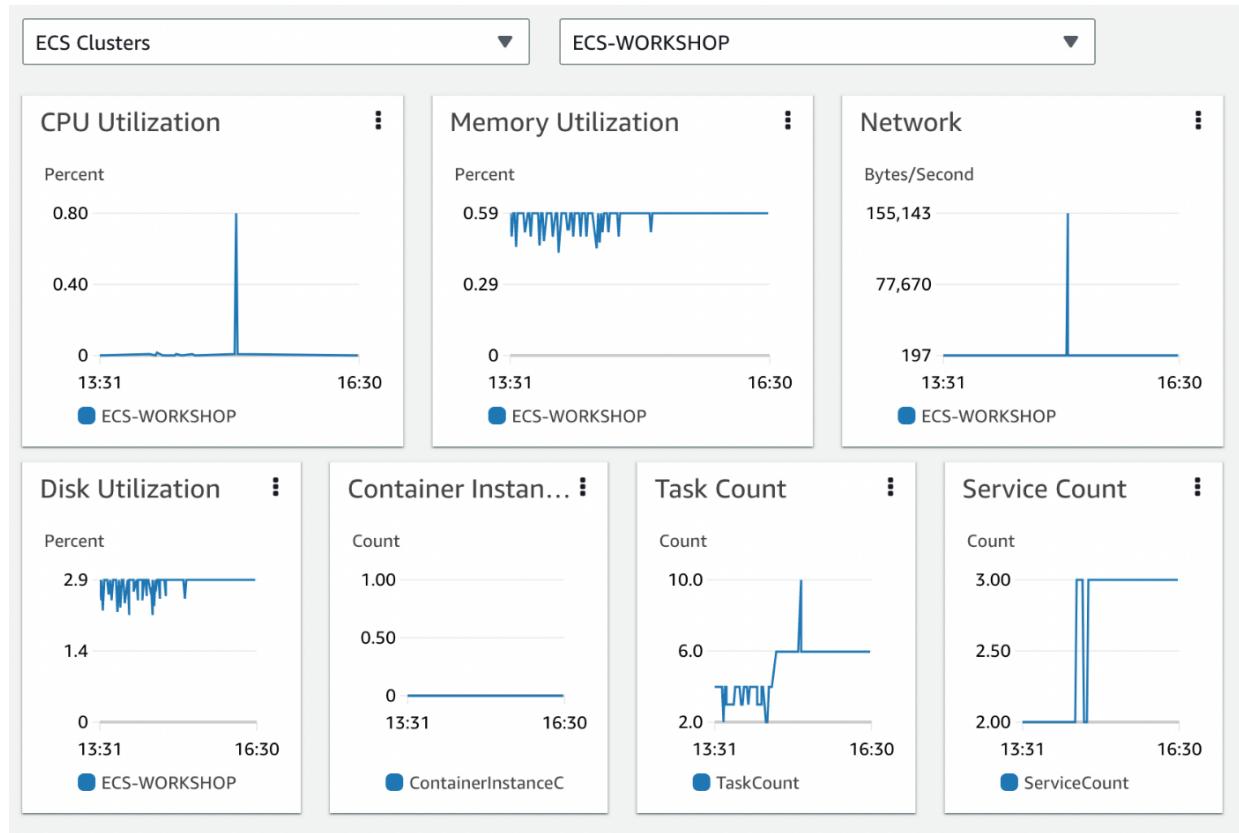
Monitoring Amazon ECS with Container Insights

1. Navigate to the [Amazon CloudWatch](#) console and select **Performance monitoring** under **Container Insights**. This will let you view your ECS metrics on an automatically configured CloudWatch Dashboard.
2. Click on the **ECS Clusters** and choose **ecs-workshop**

The screenshot shows the AWS CloudWatch Metrics console with the following dropdowns:

- Performance monitoring
- ECS Clusters: ecs-workshop

The following metrics for your **ECS Cluster** will be displayed



Click on the ECS Cluster drop down and select ECS Services for more information on ECS resources.

ECS Services ▲

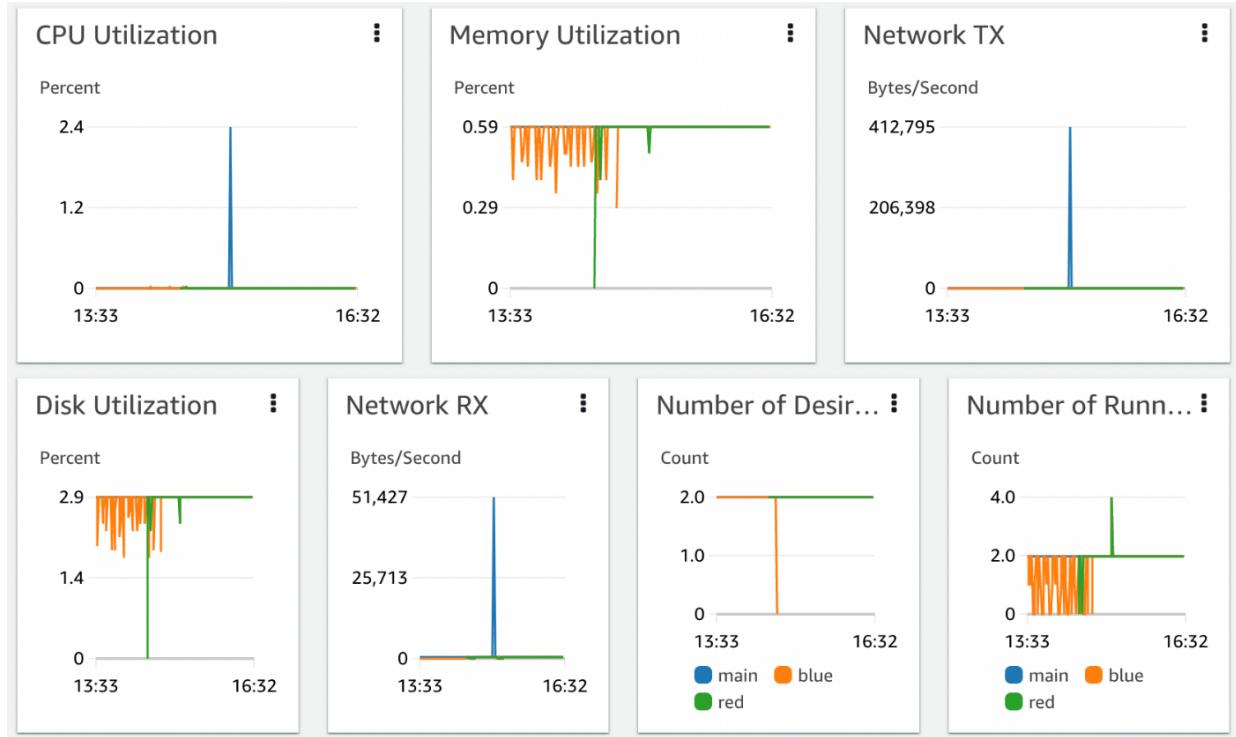
ECS Clusters

ECS Instances

ECS Services ✓

ECS Tasks

The following metrics for your **ECS Services** will be displayed



In the left hand pane, click on Log Groups to see the log groups available for the ECS Cluster, services and container insights.

The CloudWatch Log Groups interface shows the following details:

- Log groups (10):** By default, we only load up to 10000 log groups.
- Search bar:** ecs (5 matches, Exact match)
- Actions:** Actions ▾, View in Logs Insights, Create log group
- Table Headers:** Log group, Data protection, S..., Retention, Metric filters, Contr...
- Log Groups:**
 - /ecs/reddef: Data protection: Inactive, Retention: Never expire
 - /ecs/maindef: Data protection: Inactive, Retention: Never expire
 - /ecs/bluedef: Data protection: Inactive, Retention: Never expire
 - /aws/ecs/containerinsights/ECS-WORKSHOP/performance: Data protection: Inactive, Retention: 1 day

CloudWatch Alarms

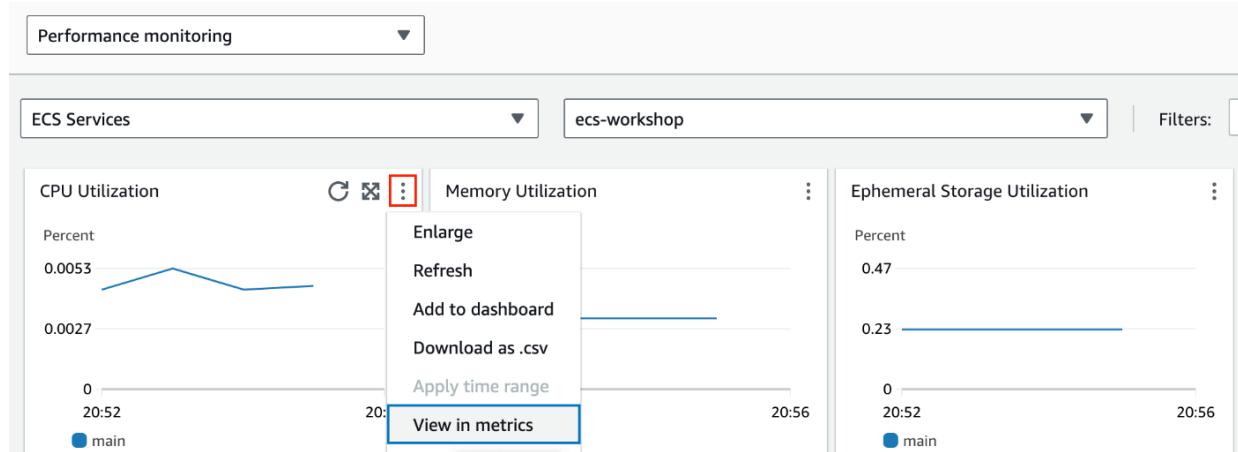
The metrics captured through Container Insights can be used to setup Alarms to get notified of any anomalies in the environment behavior.

Note: In the section **Configure Service Auto Scaling in ECS**, we will create load on the main ECS Service that will trigger the CloudWatch alarm. Complete the **CloudWatch alarm setup** and then follow the **Configure Service Auto Scaling in ECS** to trigger the alarm.

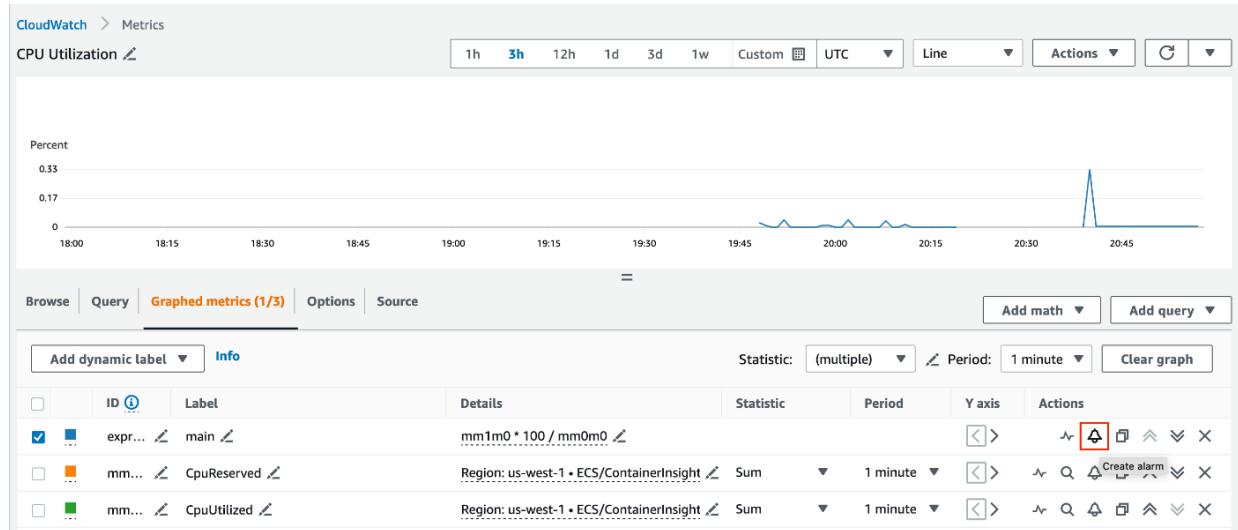
CloudWatch alarm setup

In CloudWatch Container Insights, we're going to drill down to create an alarm using CloudWatch for CPU Utilization for our application.

Select **ECS Services** and click on the three vertical dots in the upper right of the CPU Utilization box. And select **View in Metrics**.



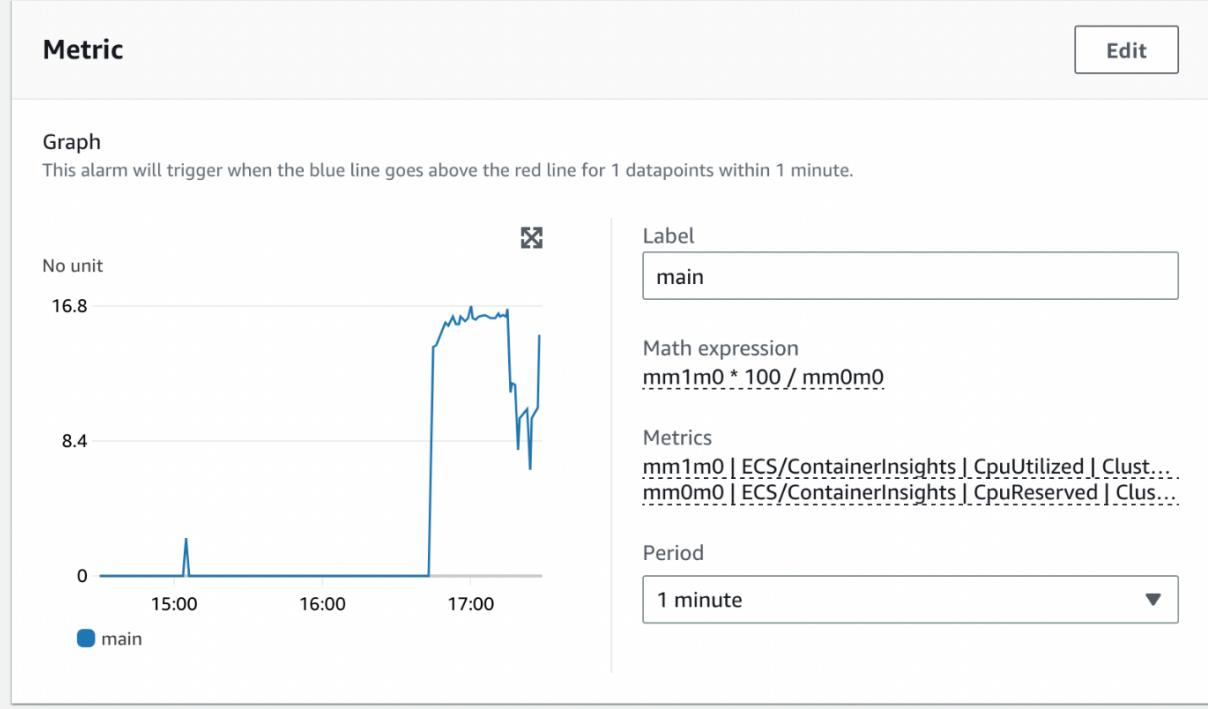
This will take you to a screen like the one below where you can configure an alarm



In the Graph metrics (x/y) section below the graph, click on the alarm bell under actions.

The following **Specify metric conditions** screen will appear.

Specify metric and conditions



Leave everything as default and enter **15** in the **Define the threshold value** screen. By doing this, we are setting the CPU Utilization threshold for the alarm to be 15%. Select **Next**

Conditions

Threshold type

Static

Use a value as a threshold

Anomaly detection

Use a band as a threshold

Whenever main is...

Define the alarm condition.

Greater

> threshold

Greater/Equal

\geq threshold

Lower/Equal

\leq threshold

Lower

< threshold

than...

Define the threshold value.

15



Must be a number

► Additional configuration

Click **Next**

In the **Configure actions** screen, click Remove to remove the default action.

Configure actions

Notification

Alarm state trigger

Define the alarm state that will trigger this action.

Remove

In alarm

The metric or expression is outside of the defined threshold.

OK

The metric or expression is within the defined threshold.

Insufficient data

The alarm has just started or not enough data is available.

Send a notification to the following SNS topic

Define the SNS (Simple Notification Service) topic that will receive the notification.

Select an existing SNS topic

Create new topic

Use topic ARN to notify other accounts

Send a notification to...

Select an email list

Only email lists for this account are available.

Add notification

In the **Add a description** screen, enter a name for the alarm and select **Next**

Add name and description

Name and description

Alarm name

ecs-main-cpu-alert

In the review screen, select **Create alarm** to create the alarm. Once complete, you should be able to see a screen such as the one below.

Configure Service Auto Scaling in ECS

You can increase or decrease your desired task count by integrating Amazon ECS on Fargate with Amazon CloudWatch alarms and [Application Auto Scaling](#). Then, you can use CloudWatch metrics to configure your CloudWatch alarms.

When your CloudWatch alarms trigger an Auto Scaling policy, Application Auto Scaling decides the new desired count based on the configured scaling policy. Then, Application Auto Scaling makes the **UpdateService** API call to Amazon ECS with the new desired count value. The Amazon ECS [service scheduler](#) launches or shuts down tasks to meet the new desired count. Your scaling activity remains in the **InProgress** state until the desired count and the running count are same.

Navigate to the [Amazon ECS](#) console and select our **ECS-WORKSHOP** cluster. Click on the **main** service and then **Update**.

The screenshot shows the AWS ECS Services list. There are three services listed: 'blue', 'main', and 'red'. The 'main' service is selected, indicated by a checked checkbox in the first column. The 'Update' button in the top right corner is also highlighted with a red box. The table columns include Service name, Status, ARN, Service type, and Deployments and tasks.

	Service name	Status	ARN	Service...	Deployments and tasks
<input type="checkbox"/>	blue	Active	arn:aws:ecs...	REPLICA	<div style="width: 100%;"><div style="width: 100%;">2/2 Tasks ru...</div></div>
<input checked="" type="checkbox"/>	main	Active	arn:aws:ecs...	REPLICA	<div style="width: 100%;"><div style="width: 100%;">2/2 Tasks ru...</div></div>
<input type="checkbox"/>	red	Active	arn:aws:ecs...	REPLICA	<div style="width: 100%;"><div style="width: 100%;">2/2 Tasks ru...</div></div>

Scroll down to the **Set Auto Scaling (optional)** and click on **Configure Service Auto Scaling** to adjust your service's desired count, as shown below

Set the **Minimum number of tasks** to 2, the **Desired number of tasks** to 2 and the **Maximum number of tasks** to 10.

The screenshot shows the 'Service auto scaling - optional' configuration section. It includes a description of how auto scaling adjusts the service's desired count based on CloudWatch alarms. Below this, there is a checkbox labeled 'Use service auto scaling' which is checked and highlighted with a red box. To the right of the checkbox is a description: 'Configure service auto scaling to adjust your service's desired count'. Below this are two input fields: 'Minimum number of tasks' (set to 2) and 'Maximum number of tasks' (set to 10), both of which are also highlighted with red boxes.

▼ Service 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.

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

Minimum number of tasks
The lower boundary to which service auto scaling can adjust the desired count of the service.

Maximum number of tasks
The upper boundary to which service auto scaling can adjust the desired count of the service.

Click on **Add scaling policy**

Configure the **Policy name** as **CPU** and select **ECSServiceAverageCPUUtilization** in the **ECS service metric**

Scaling policy type | [Info](#)
Create either a target tracking or step scaling policy.

Target tracking

Increase or decrease the number of tasks that your service runs based on a target value for a specific metric.

Step scaling

Increase or decrease the number of tasks that your service runs based on a set of scaling adjustments, known as step adjustments, that vary based on the size of the alarm breach.

Policy name

CPU

ECS service metric

ECSServiceAverageCPUUtilization

Target value

10

Scale-out cooldown period

30

Scale-in cooldown period

30

Turn off scale-in

We will add an aggressive **Target value** of 10 to demonstrate the ECS Autoscaling functionality and **Scale-out & Scale-in cooldown period** of 30. Click on **Save** and click **Next** and **Update service** to complete the configuration.

Navigate to the [AWS Cloud9 ecs-workshop IDE](#)

Install the Siege stress tool on your **ecs-workshop** IDE tool

```
sudo yum -y install siege
```

Navigate to the [EC2 Load Balancers](#). console and copy the DNS name of your ALB into **[Your ALB DNS name]** in the command below

```
siege -c 500 -i [Your ALB DNS name] > /dev/null
```

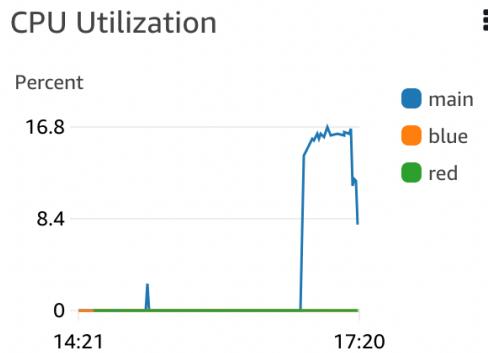
Note: Make sure that '/' is at the end of the ALB DNS name, because the **web** service is bound to the default path (/). Without it, the traffic burst will not hit our web application.

You should see the following message saying the service is now under siege.

```
\oli:~/environment/ecsworkshop (main) $siege -c 500 -i http://ecs-workshop-alb-476857044.us-east-1.elb.amazonaws.com/ > /dev/null
** SIEGE 3.0.8
** Preparing 500 concurrent users for battle.
The server is now under siege...
```

Let the script run in the background and move onto the next section.

To check the service CPU utilisation, go back to CloudWatch and analyse the CloudWatch ECS Service CPU metric graph for the main service. In the example below, the blue line represents the main service,



After a while, you will notice additional **maindef** tasks are being started, as shown below one is in **Pending** status.

Tasks (1/3)							
Task		Last status	Desired st...	Task defi...	Revis...	Health sta...	Started at
<input checked="" type="radio"/>	6e9671cc454c495...	Running	Running	maindef	1	Unknown	46 minutes ago
<input type="radio"/>	8b4efb2ebab945a...	Pending	Running	maindef	1	Unknown	-
<input type="radio"/>	ec7c65219304467...	Running	Running	maindef	1	Unknown	46 minutes ago

Click on the Events tab to confirm the Autoscaling event triggered

If you do not see any tasks scale up, check the CPU percentage hits the threshold configured in the ECS Autoscaling policy.

Health and metrics	Tasks	Logs	Deployments	Events	Configuration	Networking	Tags
Events (7)							
Started at	Message	Event ID					
September 18, 2023 at 21:26 (UTC+1:00)	Message: Successfully set desired count to 5. Waiting for change to be fulfilled by ecs. Cause: monitor alarm TargetTracking-service/ecs-workshop/main-AlarmHigh-ee211104-18b6-4141-88c3-c9c61f845bd2 in state ALARM triggered policy CPU	5bff400e-c8b9-4994-a6aa-66df3a736217					
September 18, 2023 at 21:25 (UTC+1:00)	service main has started 1 tasks: task 8b4efb2ebab945ae8f77c94a9db22aa6 .	b27030cb-2f05-4a80-466faf02b964					
September 18, 2023 at 21:25 (UTC+1:00)	Message: Successfully set desired count to 3. Found it was later changed to 2. Cause: monitor alarm TargetTracking-service/ecs-workshop/main-AlarmHigh-ee211104-18b6-4141-88c3-c9c61f845bd2 in state ALARM triggered policy CPU	630d5d26-4b5f-4e9d-af11-93e271562119					

ECS Fargate Workshop Conclusion

Congratulations! You have successfully containerized the ECS Blue Red application, deployed an ECS cluster, configured the application load balancer, added the Blue Red application ECS task definitions and services, then auto scaled the app and configured a CloudWatch CPU metric using Container Insights.

In this workshop, we have:

- Created a simple containerized application **ECS Blue Red application** on Amazon ECS
- Created, Pushed, and managed Docker images for the **ECS Blue Red application** to Amazon ECR.
- Created an **ECS Fargate** cluster with **Container Insights** enabled and IAM permissions configured.
- Launched an ECS Fargate task definition for the **ECS Blue Red application**
- Launched an ECS Fargate based service for the **ECS Blue Red application**
- Configured **Amazon CloudWatch** and Amazon Container Insight
- Monitored our ECS cluster, services, and tasks with Amazon CloudWatch **Container Insights**.
- Set up ECS service Auto Scaling to automatically adjust the resources to help ensure predictable performance in the most cost-optimized manner.

Advanced ECS Fargate configurations

Now you have a base ECS Fargate cluster up and running, you can do more advanced set ups. Why not try:

AWS App Mesh with ECS Fargate

<https://docs.aws.amazon.com/app-mesh/latest/userguide/getting-started-ecs.html>

<https://github.com/aws/aws-app-mesh-examples>

Using Amazon EFS file systems with Amazon ECS using the classic console

<https://docs.aws.amazon.com/AmazonECS/latest/developerguide/tutorial-efs-volumes.html>

Creating a service using a blue/green deployment

<https://docs.aws.amazon.com/AmazonECS/latest/developerguide/create-blue-green.html>

Canary deployments with Amazon ECS using AWS App Mesh

<https://aws.amazon.com/blogs/containers/create-a-pipeline-with-canary-deployments-for-amazon-ecs-using-aws-app-mesh/>

Listening for Amazon ECS CloudWatch Events

https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ecs_cwet.html

