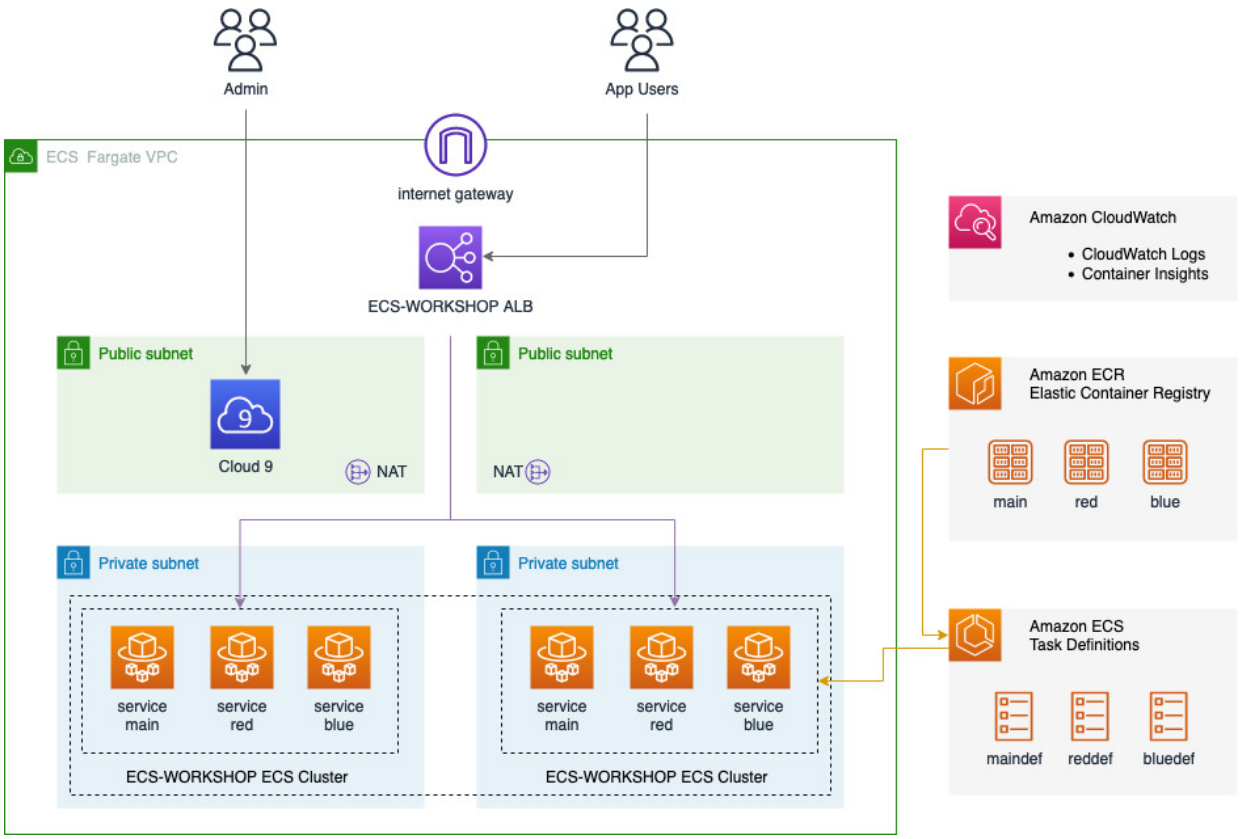
ECS Fargate Immersion Day Workshop

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.

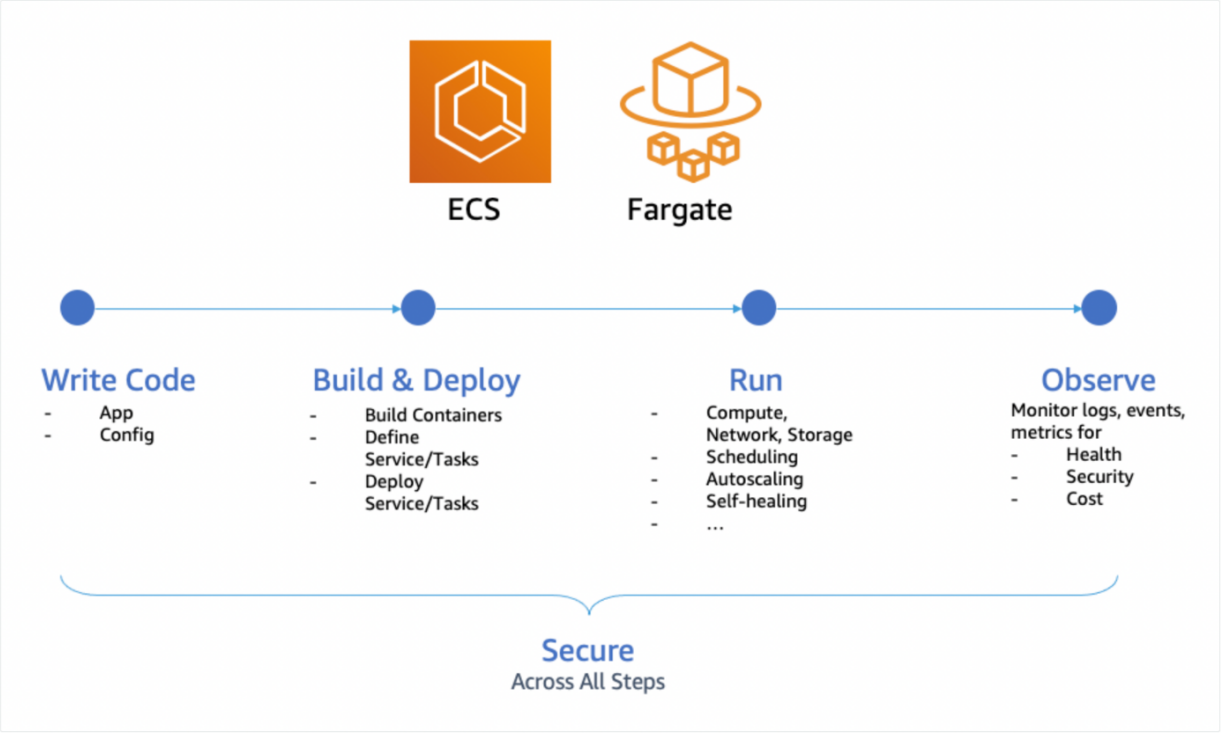
The **Blue Red** application consists of three services: **Main** (landing page), **Blue** (blue page) and **Red** (red page). In this lab, we will use the *AWS Fargate* launch type on a single cluster which we will name **ECS-WORKSHOP**.

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.

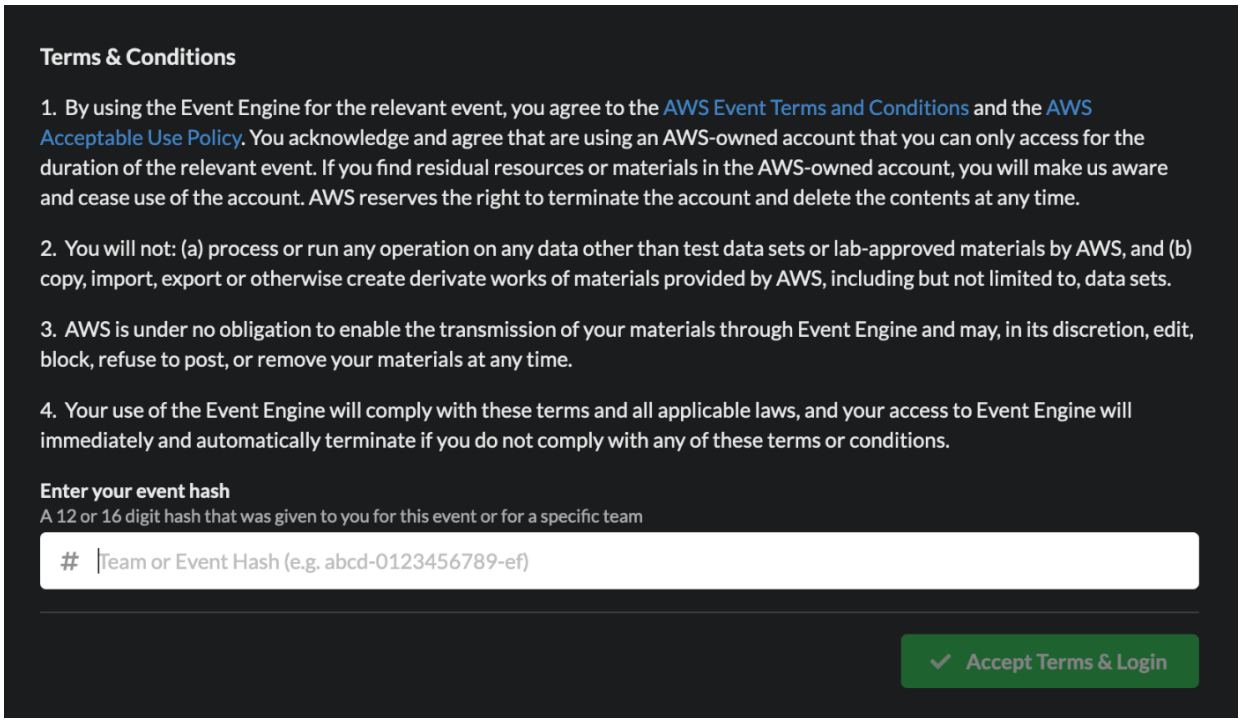
AWS Fargate is built into Amazon ECS. Just define your application's requirements, select Fargate as your launch type in the console or Command Line Interface (CLI), and Fargate takes care of all the scaling and infrastructure management required to run your containers.



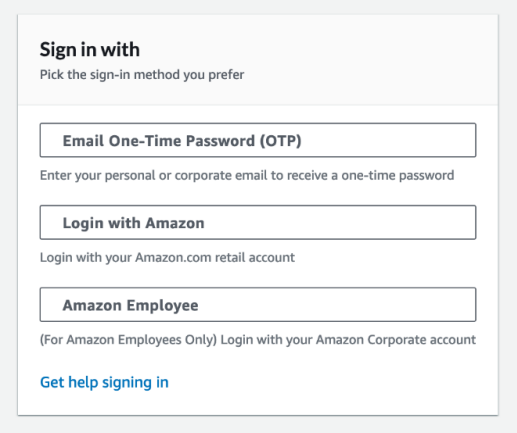
Logging into AWS Event Engine Portal

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

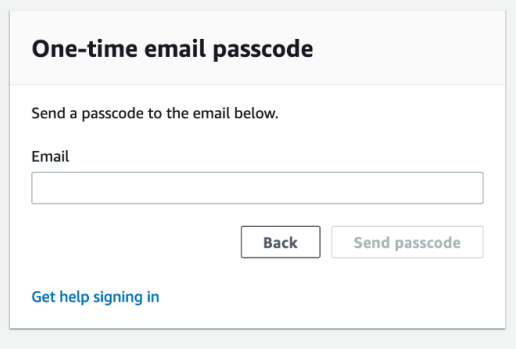
Connect to the event engine portal by clicking the button or browsing to https://dashboard.eventengine.run/. **Note**: You may have been given a URL which includes a has code so you might not see this screen. If you see a screen to login, scroll down to continue with these instructions.



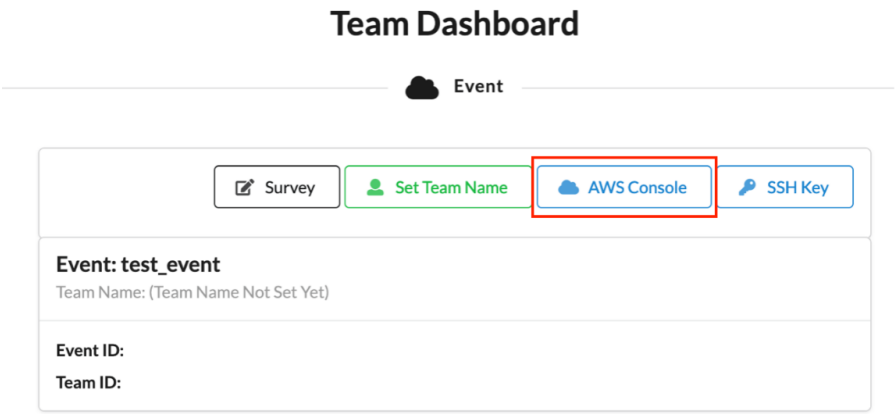
You will be taken to a screen where they can login with an Email One Time Password (OTP), and thereby get access to the Event Page:



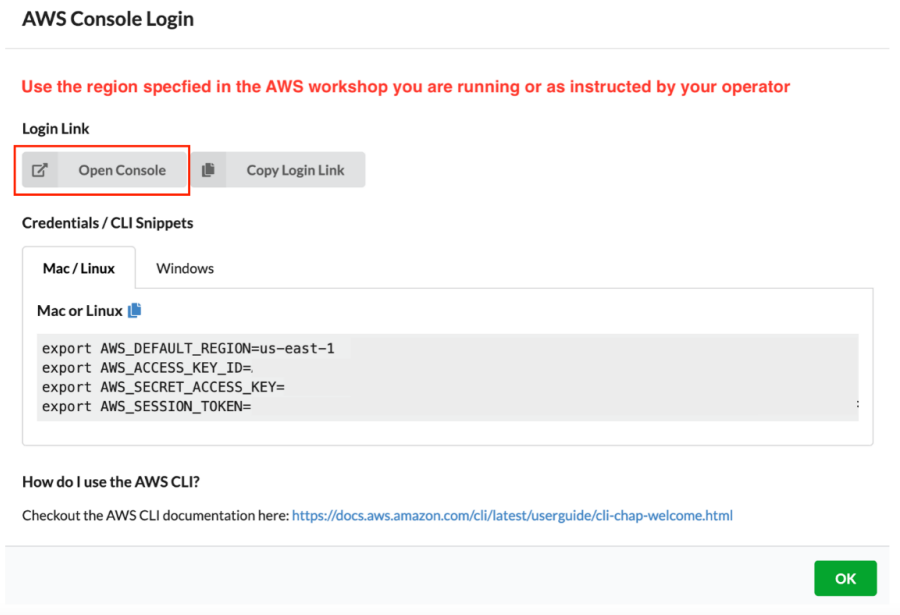
Click on Email One-Time Password (OTP)



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 you one time password, click on **AWS Console.**



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



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

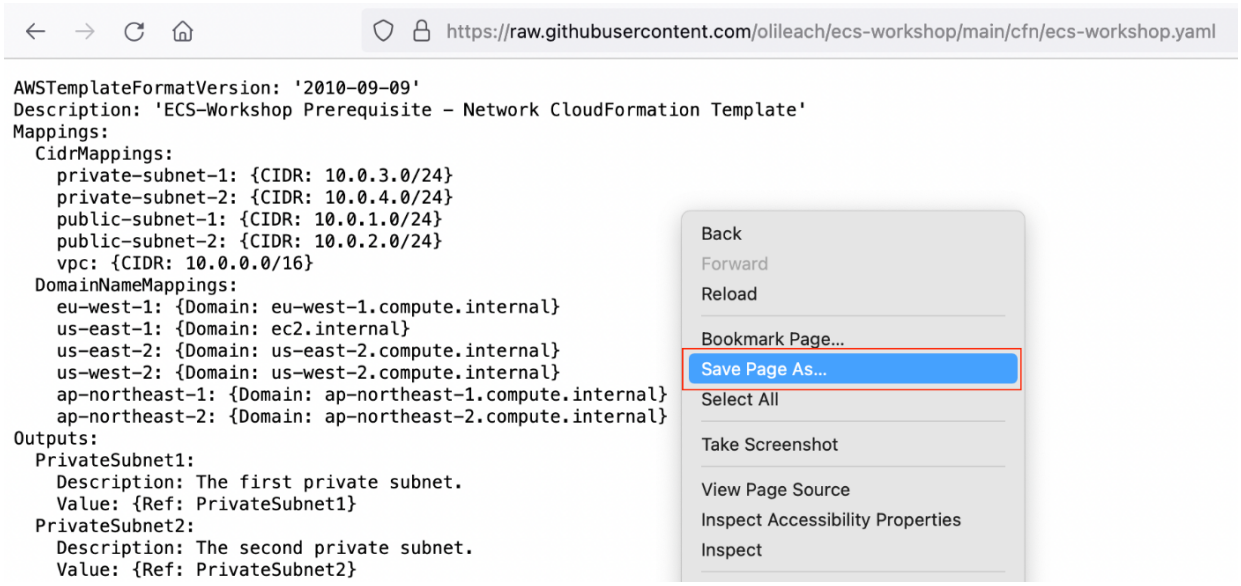
CloudFormation Stack Deployment

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
```

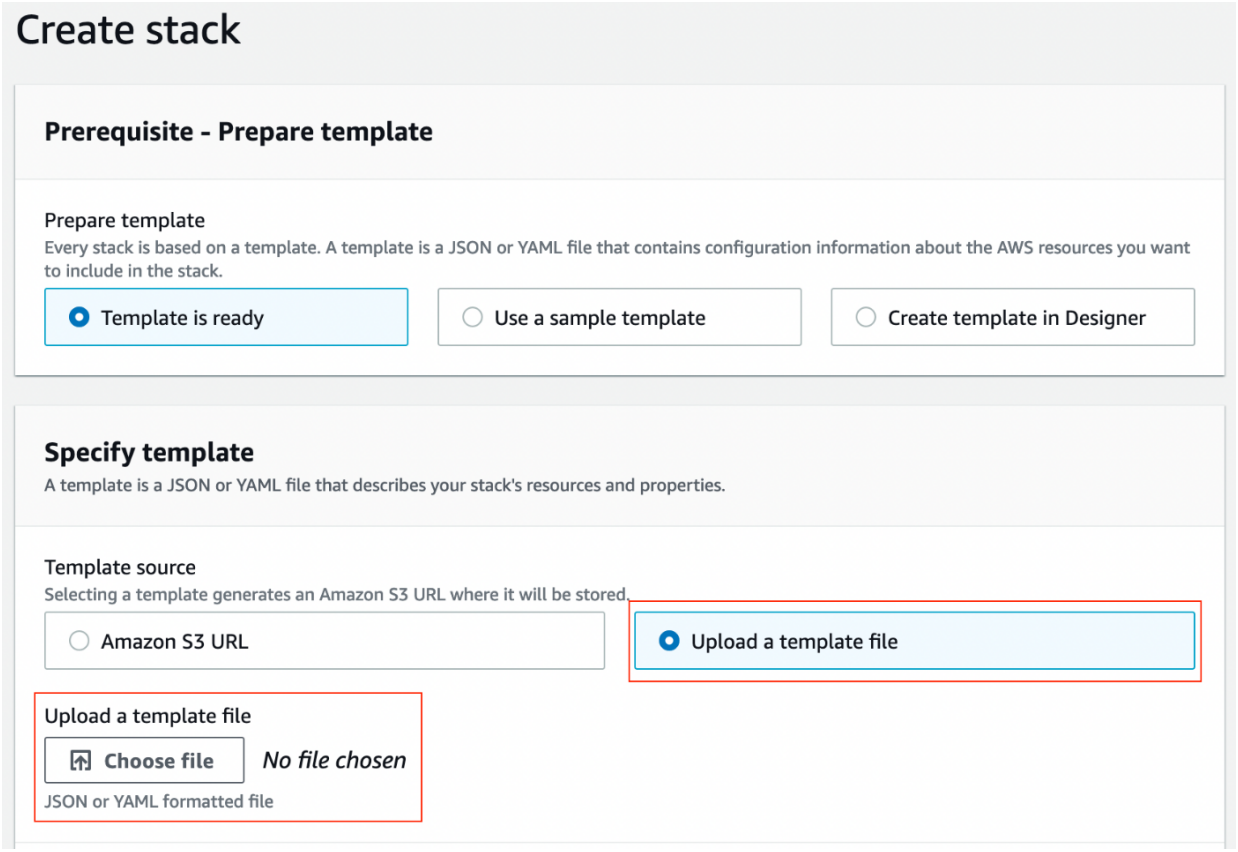
Alternatively, right click on this link https://raw.githubusercontent.com/olileach/ecs-workshop/main/ecs-workshop.yaml and download the contents to a file called **ecs-workshop.yaml.** Or, click on the link to open the web page, right click on the web page and choose **Save Page As...**



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.

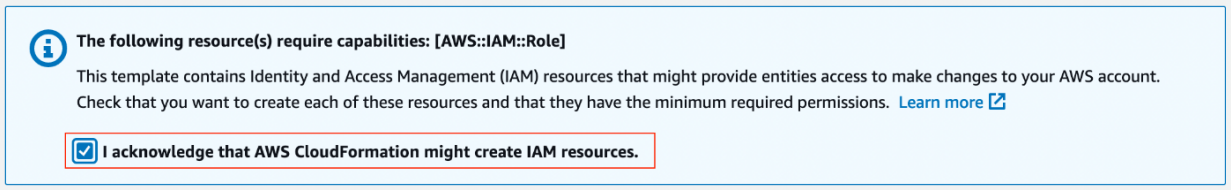
3) Click on the following link to launch the CloudFormation template https://us-east-1.console.aws.amazon.com/cloudformation/home?region=us-east-1. You will be redirected to **Amazon CloudFormation.**

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



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

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

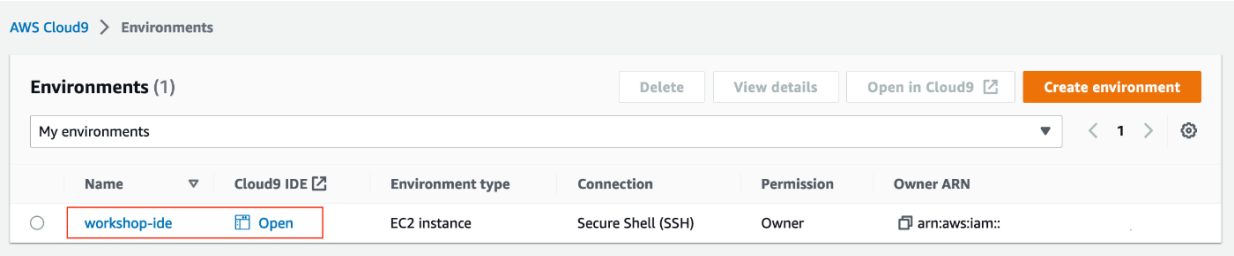


Wait for the CloudFormation template deployment to complete before moving on.

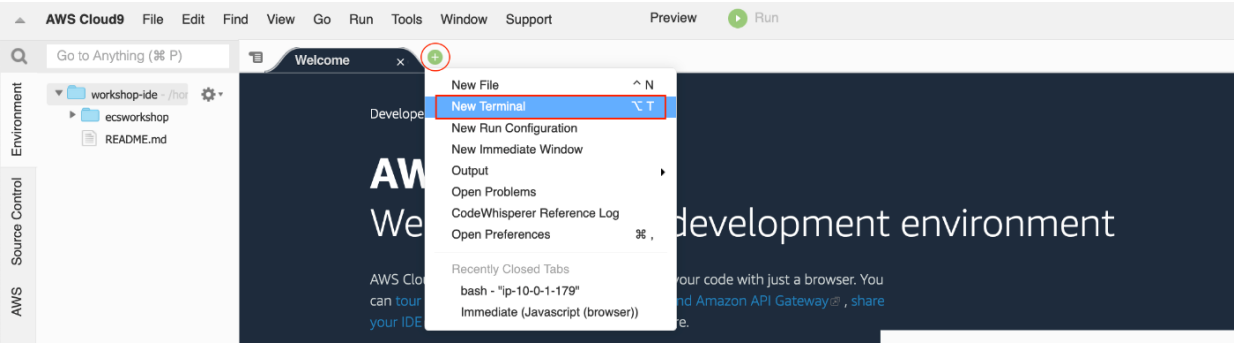
Immersion Day Environment Setup

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



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.

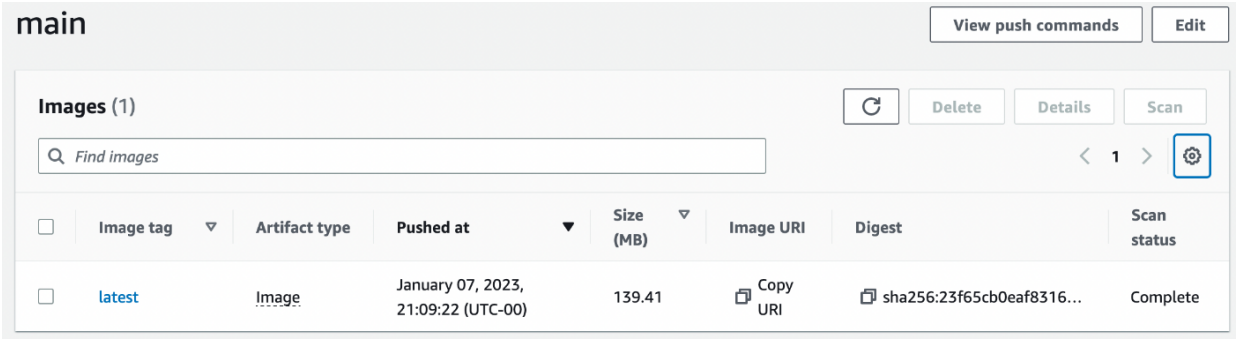


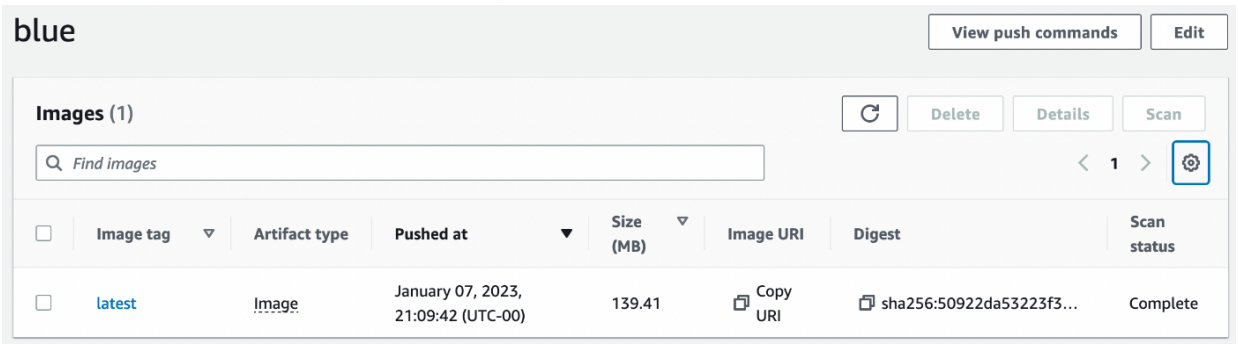
3) In the new terminal, run the following command to complete the workshop setup.

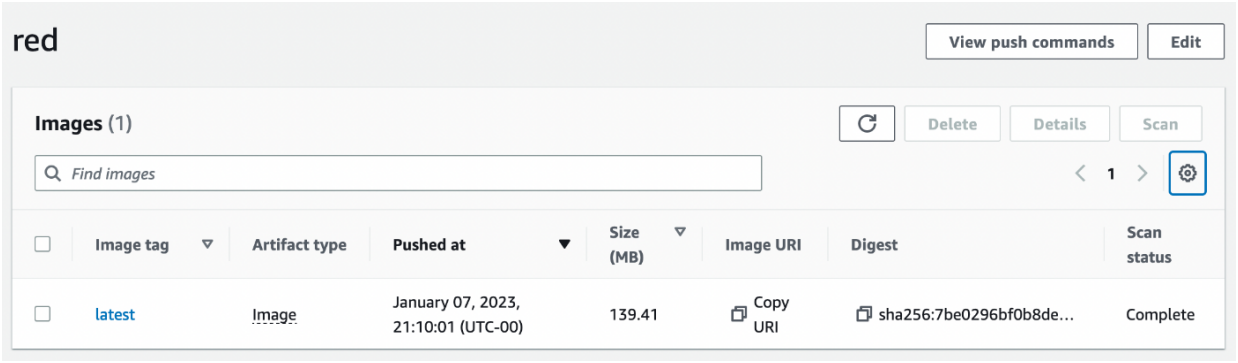
```
cd ecsworkshop
sh setup.sh
```

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.



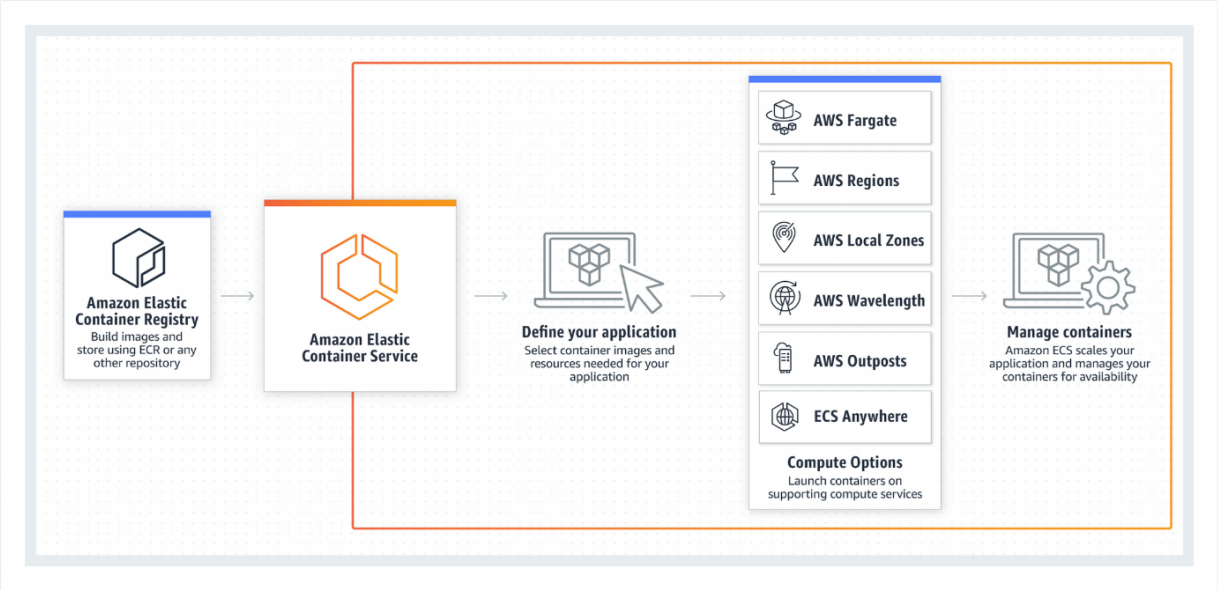




This completes the workshop setup.

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.

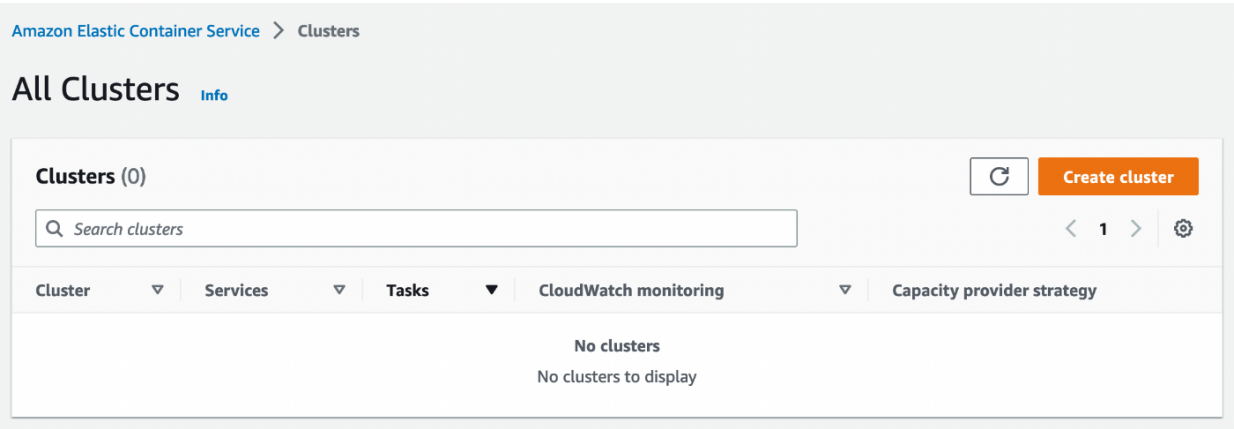


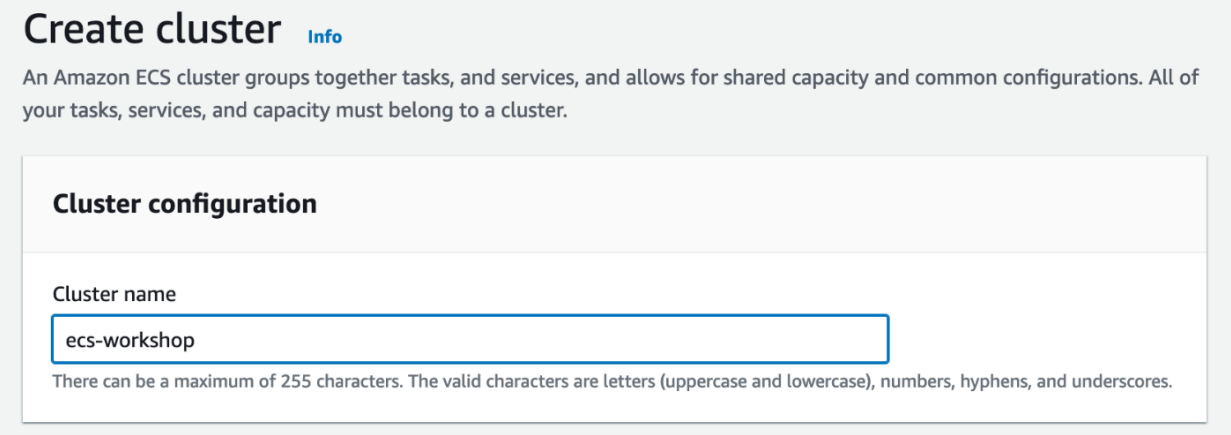
- A **service** is a configuration you can use to run and maintain a specified number of tasks simultaneously in a cluster
- A **task definition** is a blueprint for your application (which are then run as tasks on a given service). Task definitions specify various parameters of your application such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

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

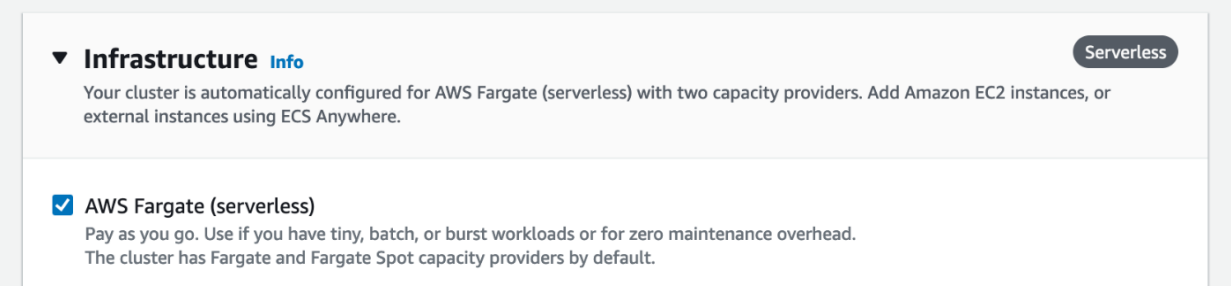
- An ECS Cluster, which serves as the logical grouping of tasks or services.
- ECS Task Definitions for our main, blue, and red services
- ECS Services which run our main, blue, and red task definitions
- An Application Load Balancer (ALB) to handle distribution of traffic across our ECS services

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.**

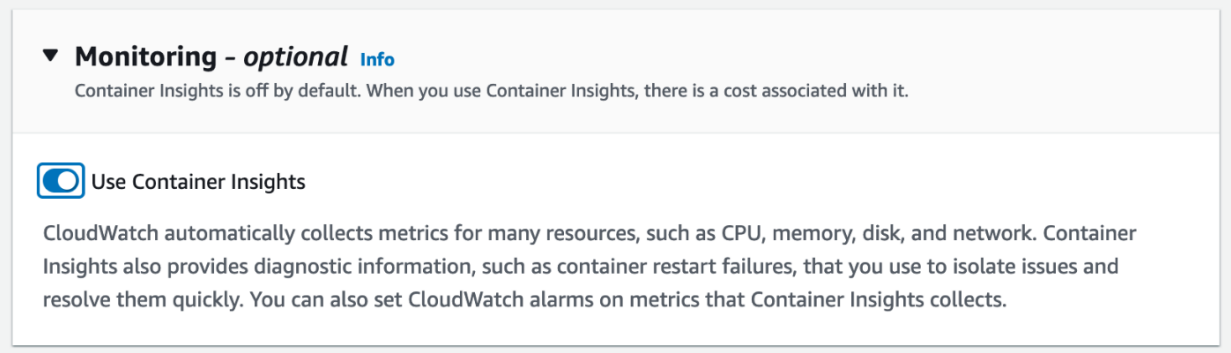




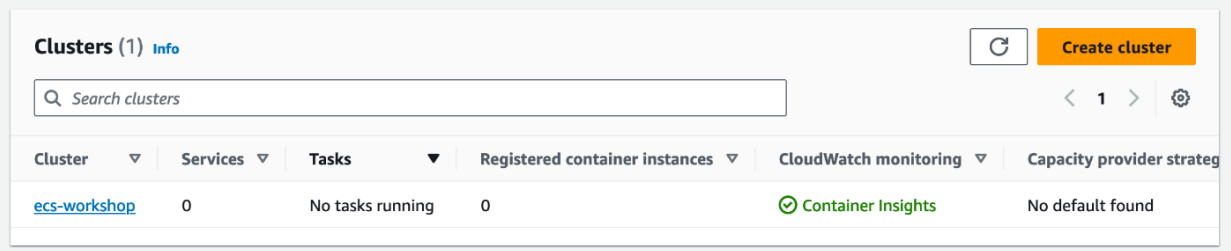
2) Click on AWS Fargate (serverless) option



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



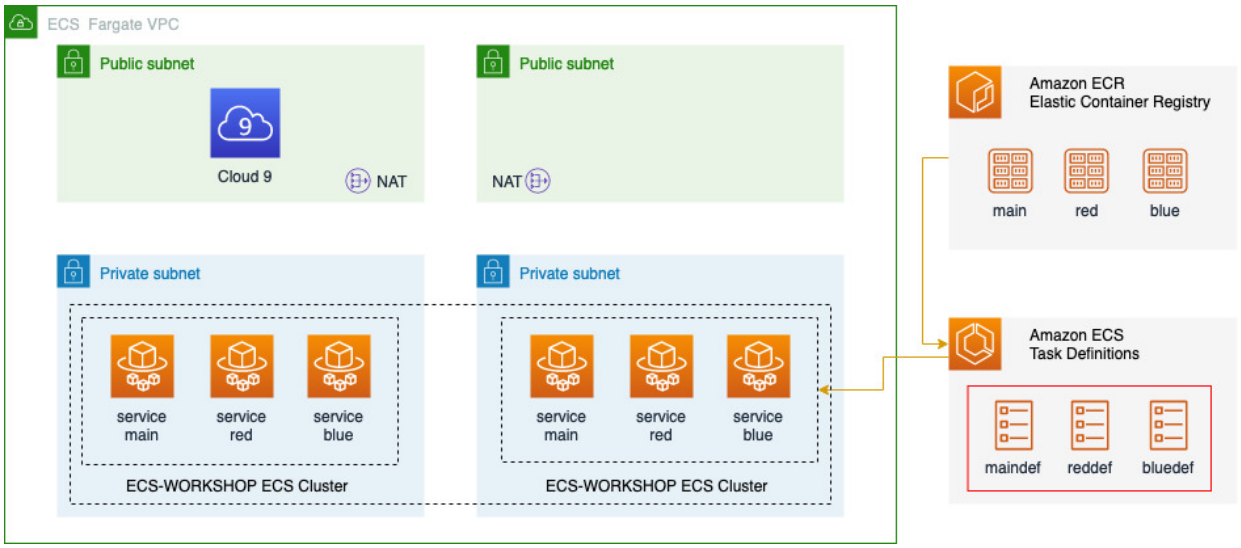
4) Refresh the page and soon you will see the **ecs-workshop** cluster you have created.



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 folllowing 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"
```

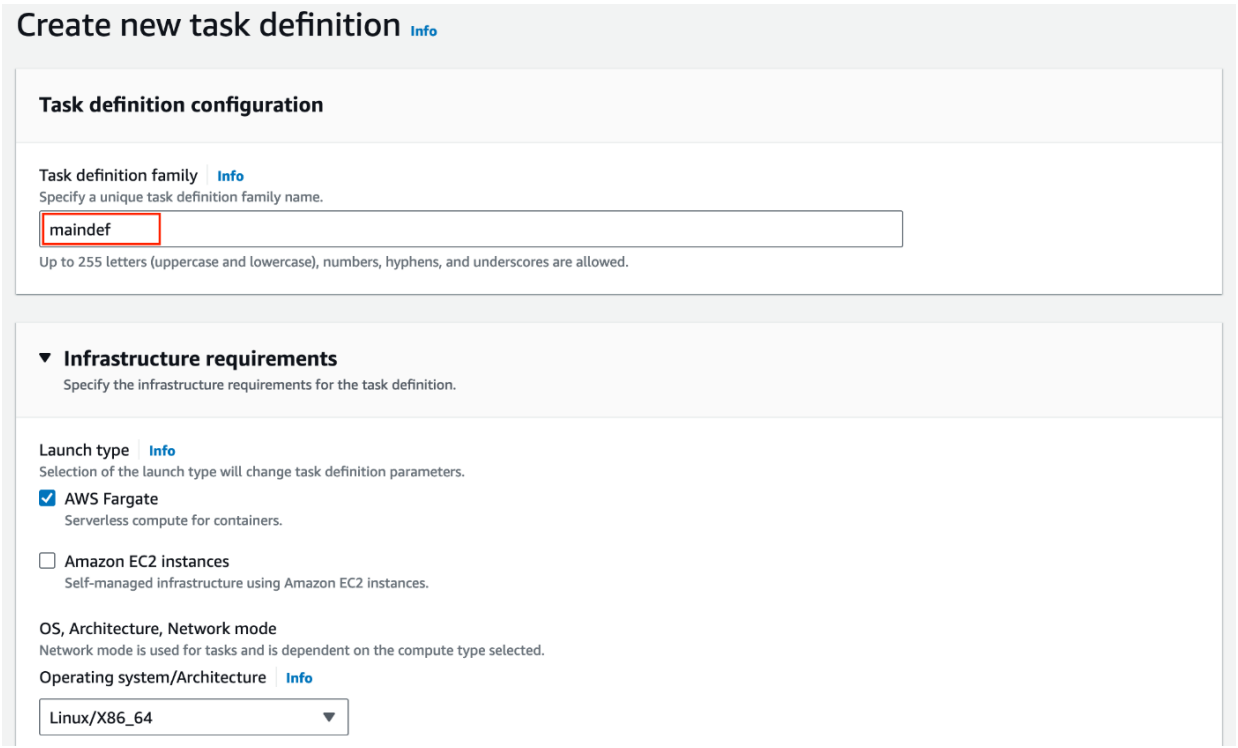
Note: If the command above does not work, copy the command from the output of the setup.sh script.

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.

```
-------------------------------------------------------------
|                     DescribeRepositories                  |
+-----------------------------------------------------------+
|  565692740138.dkr.ecr.us-east-2.amazonaws.com/main        |
|  565692740138.dkr.ecr.us-east-2.amazonaws.com/blue        |
|  565692740138.dkr.ecr.us-east-2.amazonaws.com/red         |
+-----------------------------------------------------------+
```

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

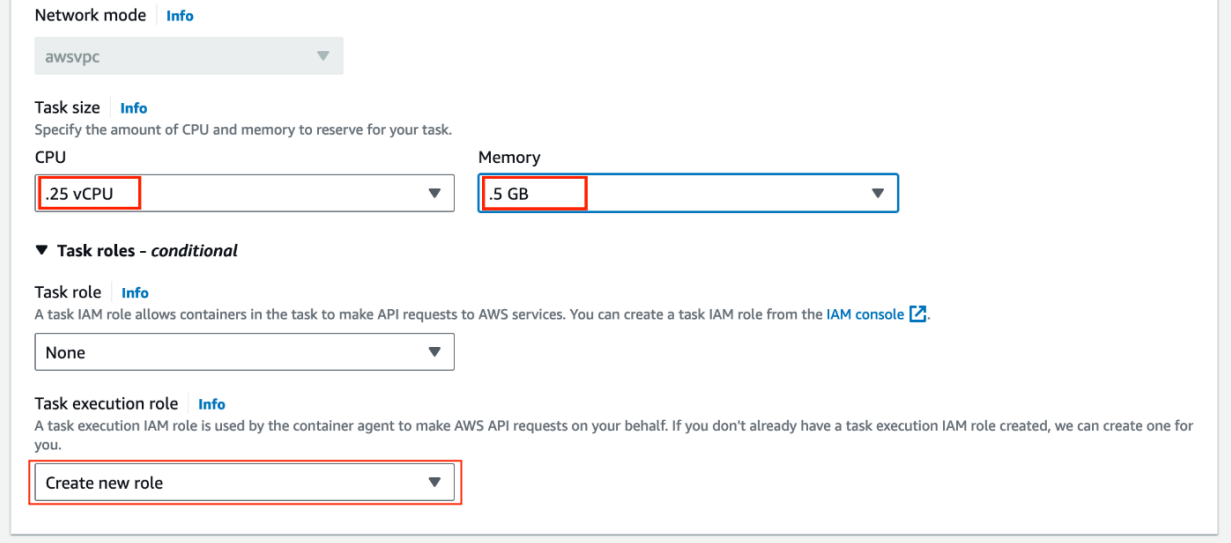
For **Task definition family**, use **maindef**, click launch type **AWS Fargate**



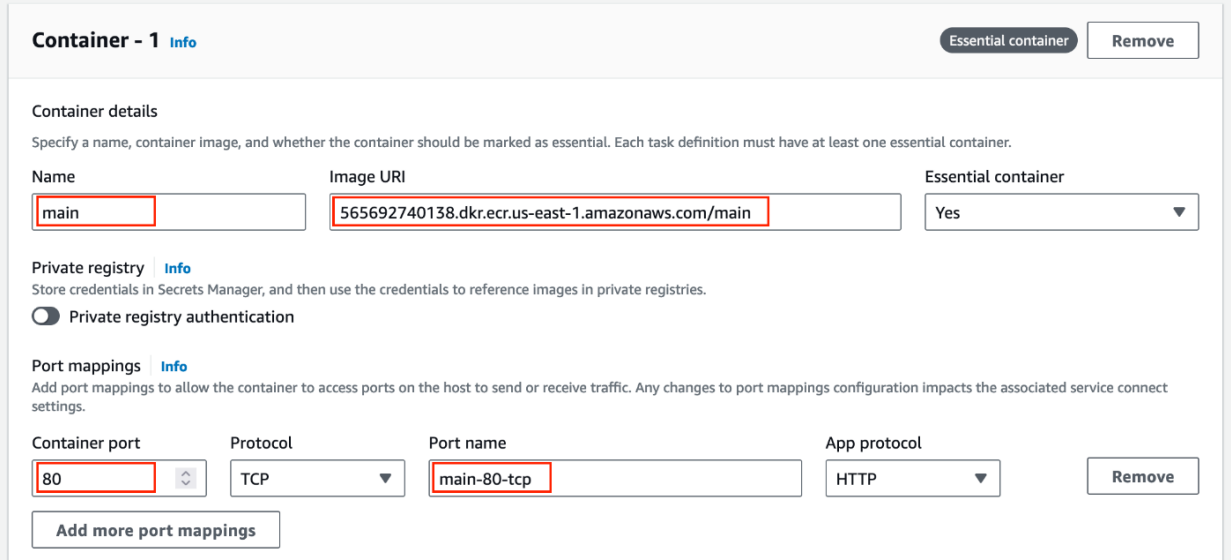
2) Configure the resources as follows:

CPU: 0.25
Memory: 0.5

For Task execution role, select **Create new role**

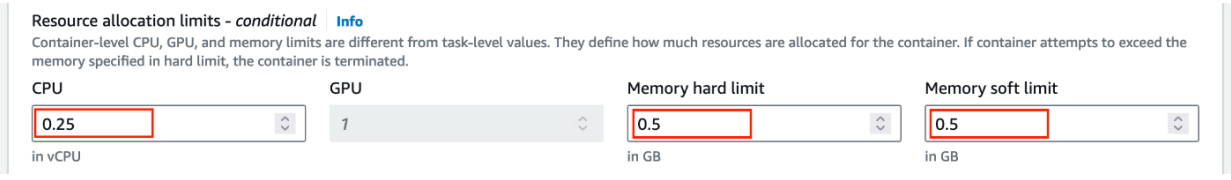


3) Configure the container values as follows:

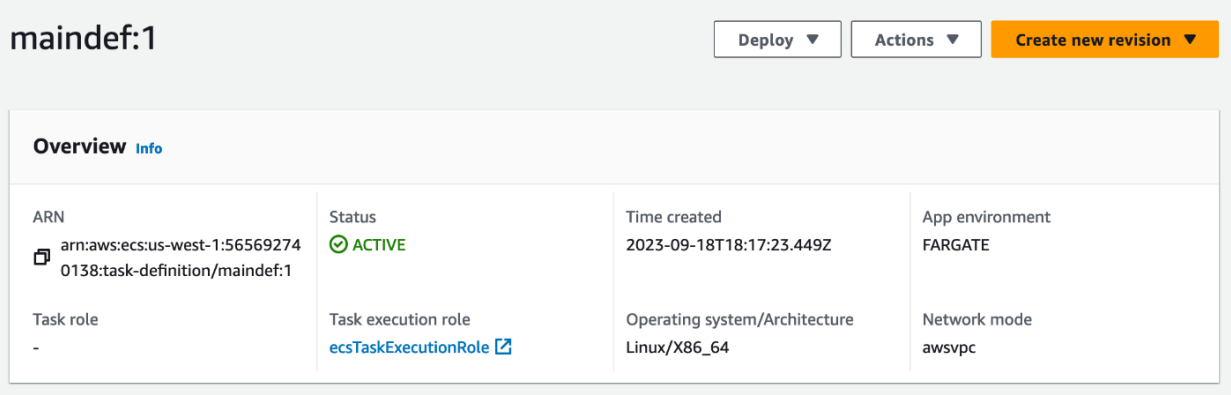


4) For Resource allocation limits - conditional, enter the following values

CPU: 0.25
Memory hard limit: 0.5
Memory soft limit: 0.5



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

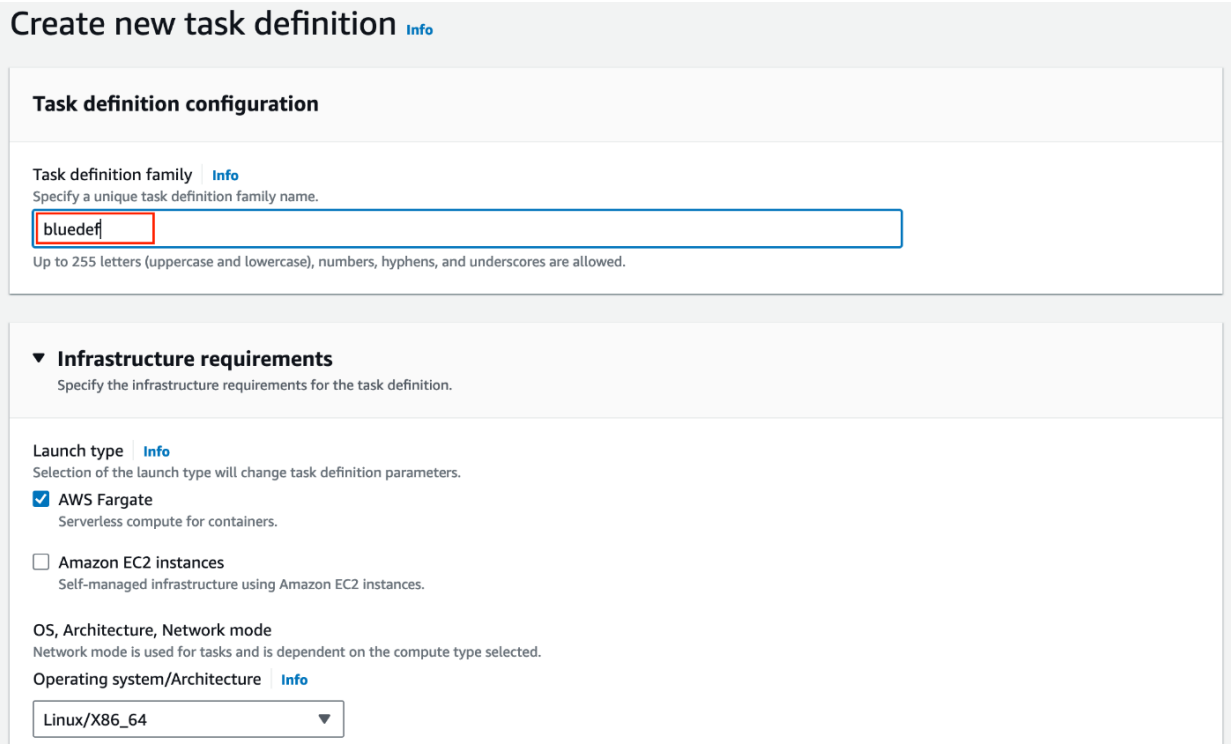


Create the blue and red task definitions

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.**

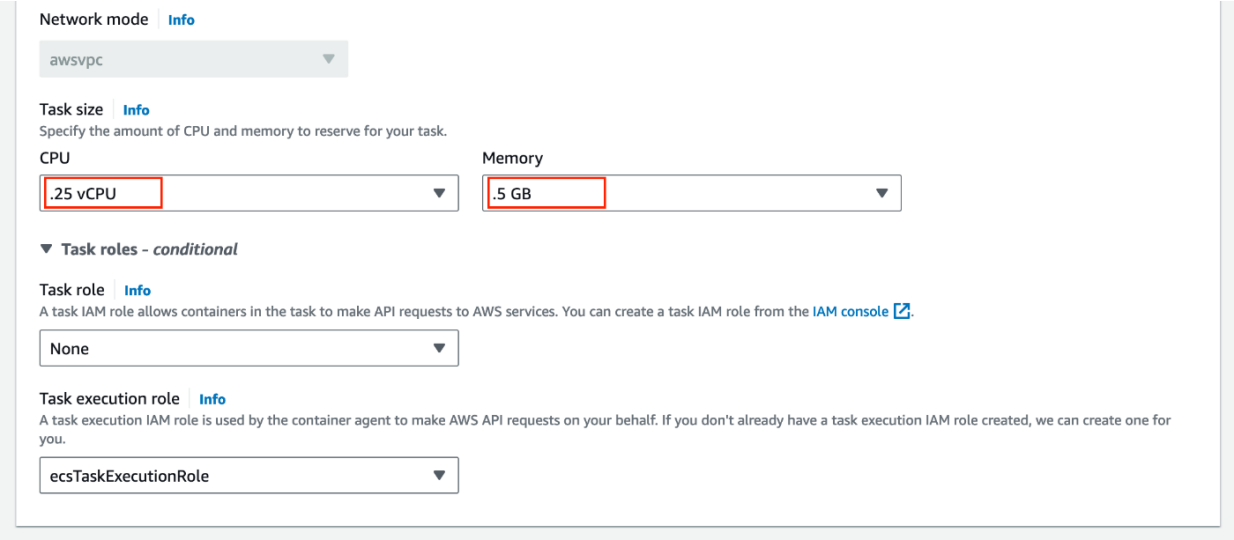
For **Task definition family**, use **bluedef**, click launch type **AWS Fargate**



2) Configure the resources as follows:

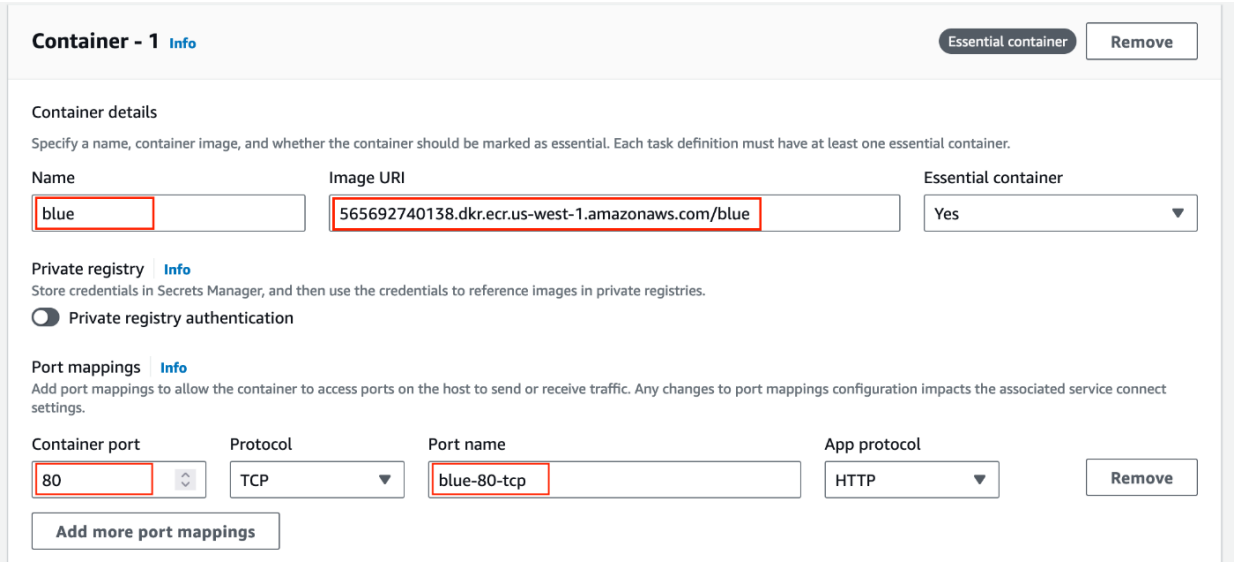
CPU: 0.25
Memory: 0.5

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



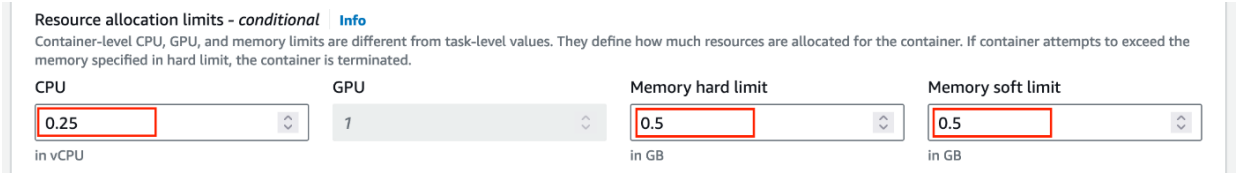
3) Configure the bluedef task definition container values as follows:

NOTE: Replace the Image URI with the image URI from Cloud9. The account number will be different

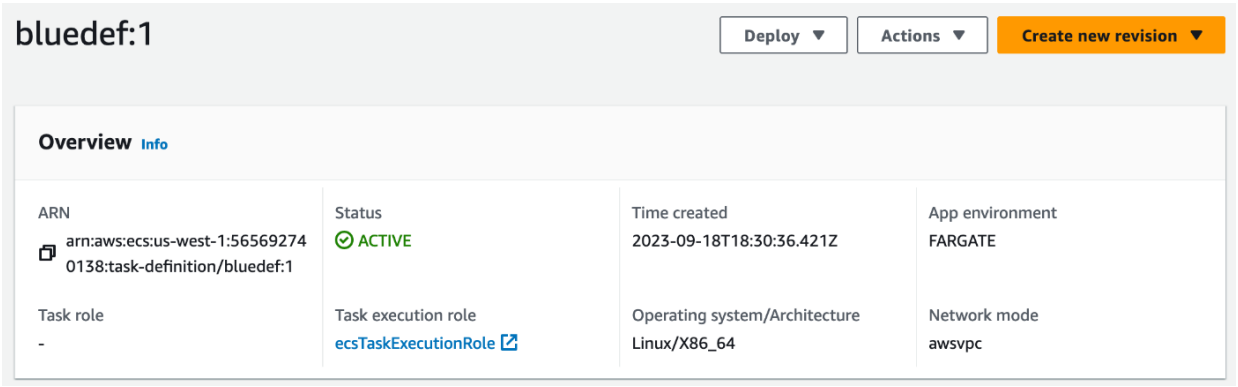


4) For Resource allocation limits - conditional, enter the following values

CPU: 0.25
Memory hard limit: 0.5
Memory soft limit: 0.5

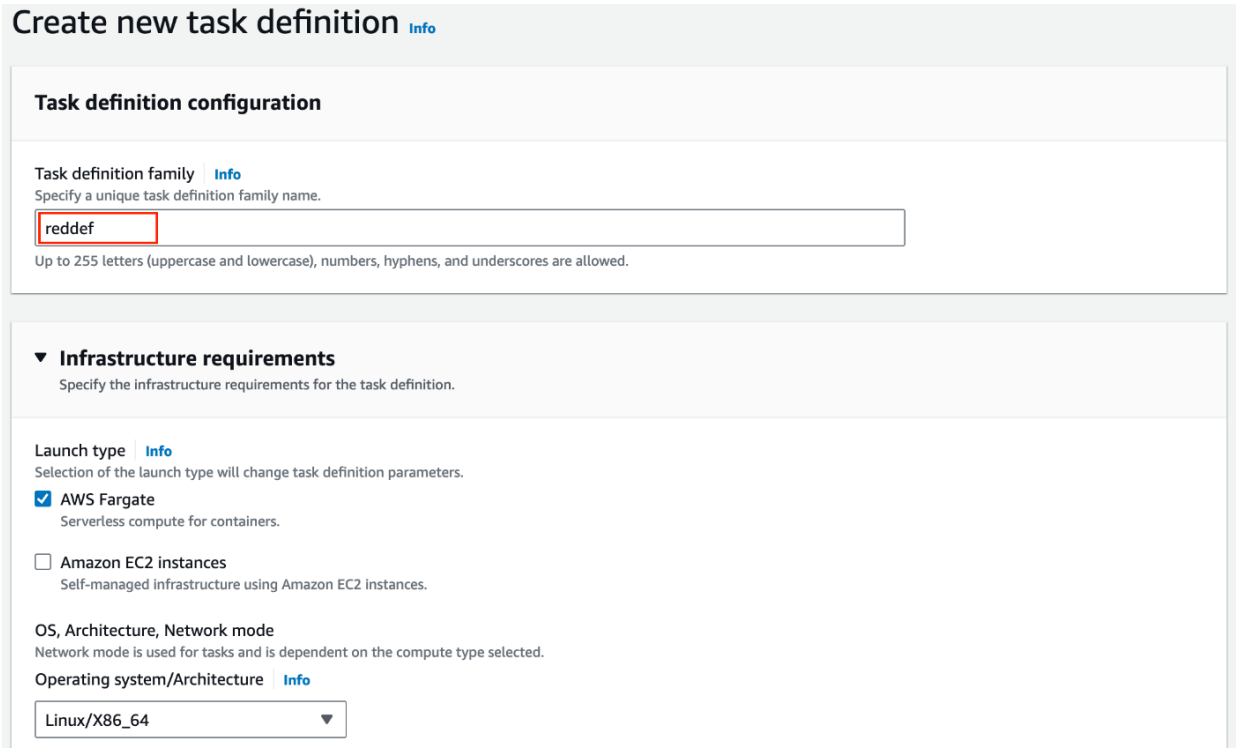


Once done, scroll down and click **Create** to create the task definition. You will see a bluedef:1 task has been created.



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

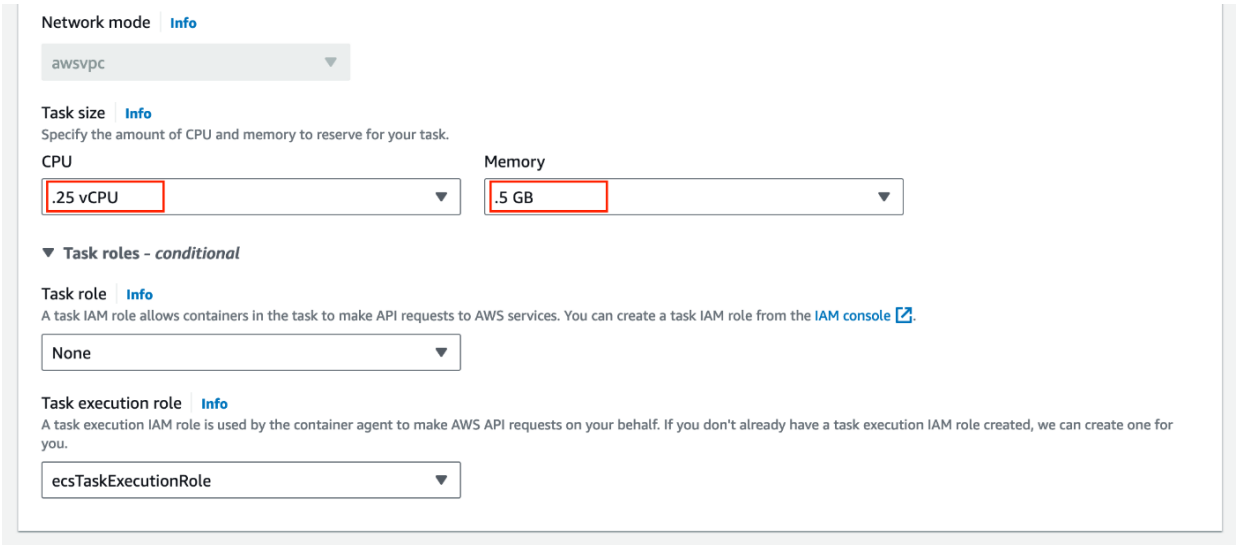
For **Task definition family**, use **reddef**, click launch type **AWS Fargate**



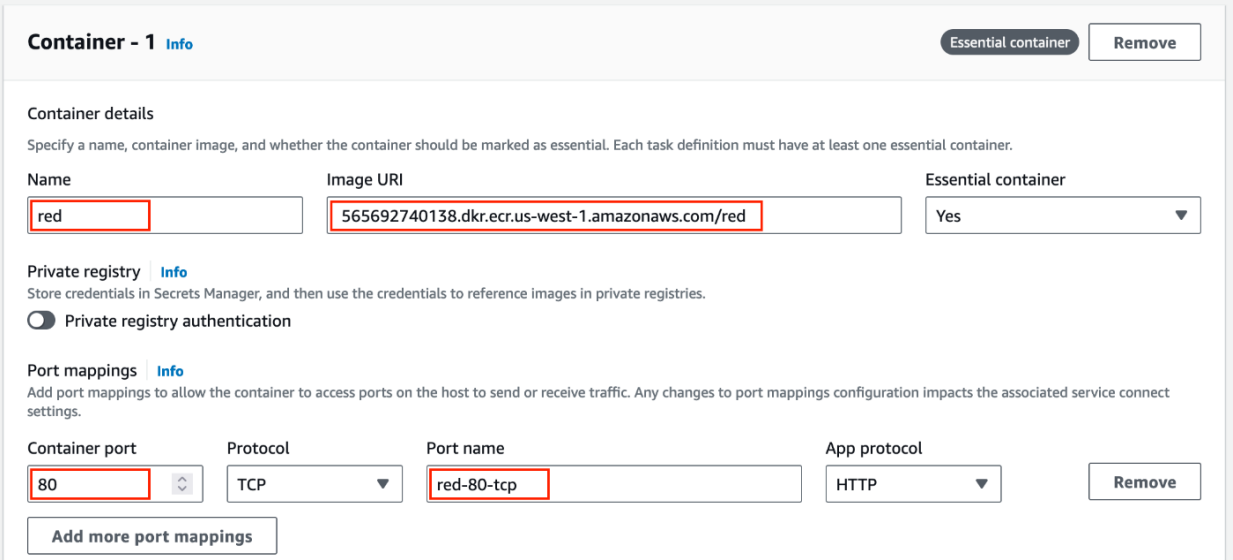
6) Configure the resources as follows:

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

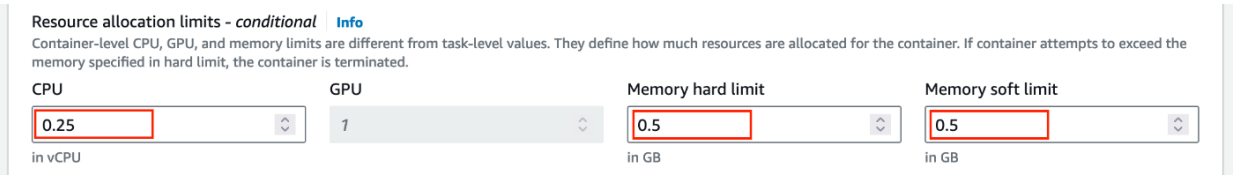


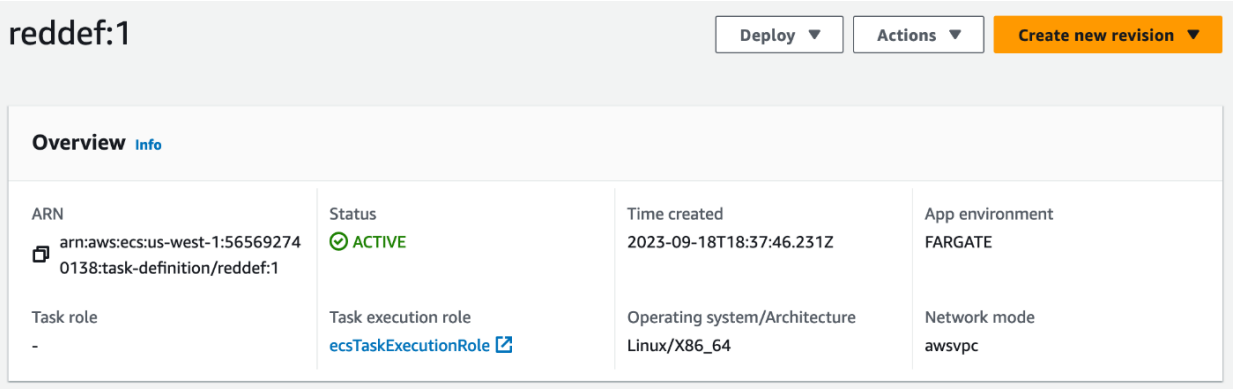
7) Configure the reddef task definition container values as follows



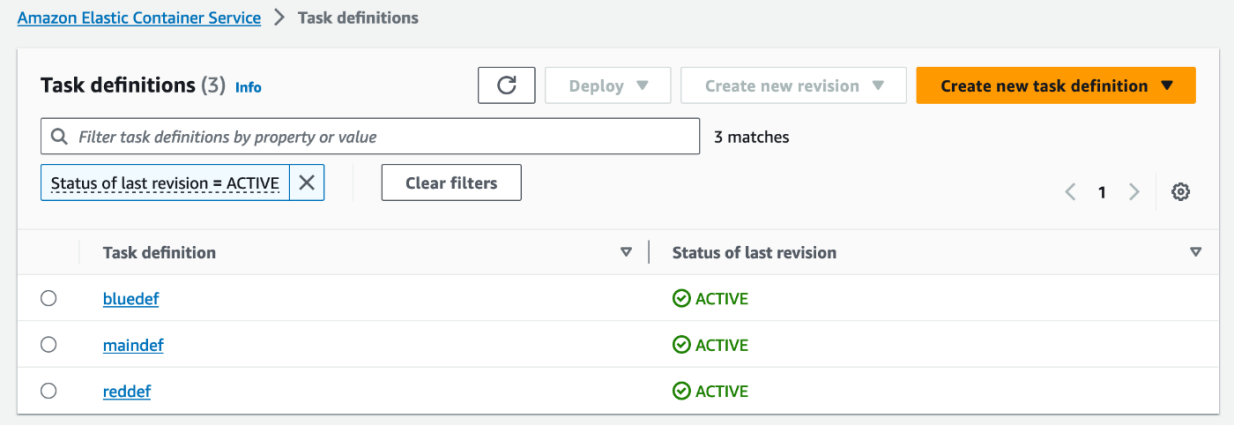
8) For Resource allocation limits - conditional, enter the following values

CPU: 0.25
Memory hard limit: 0.5
Memory soft limit: 0.5



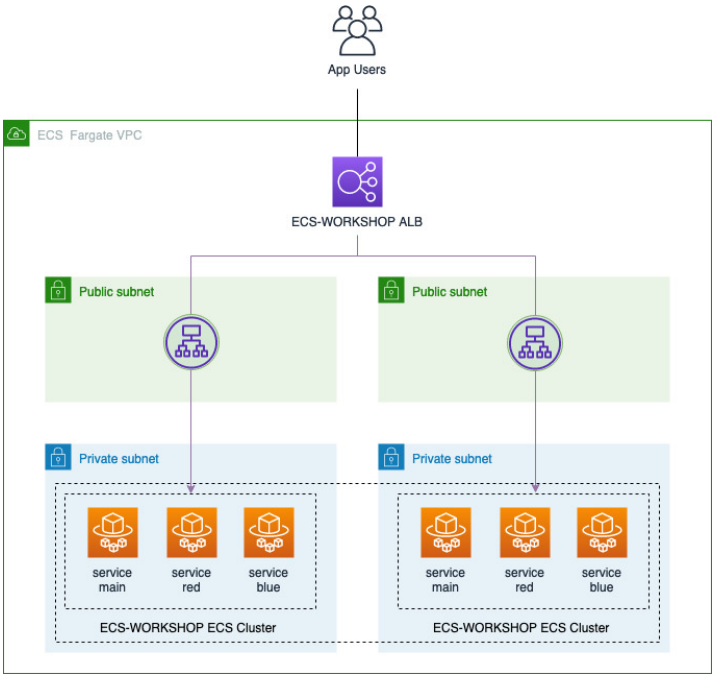


Once complete, you should see 3 task definitions for main, blue and red.



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.

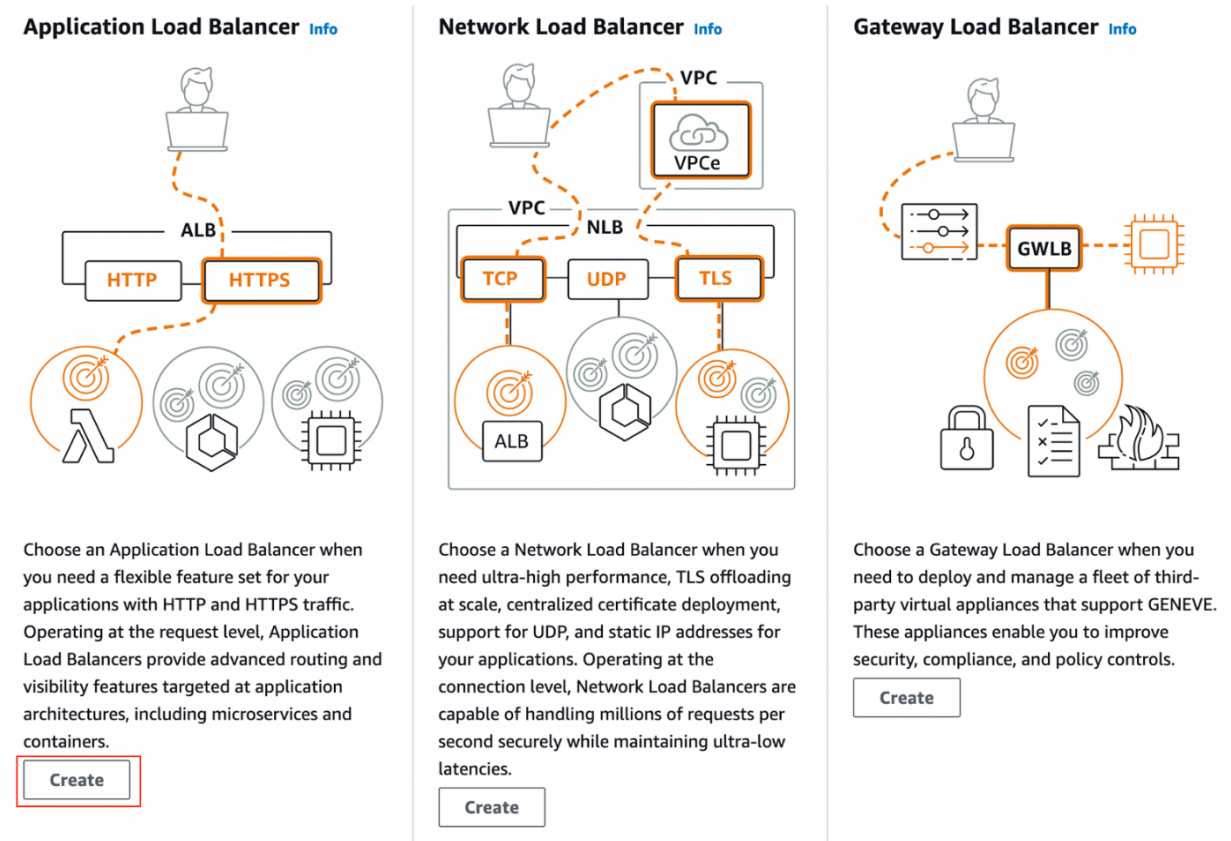
1) Navigate to the EC2 Load Balancers console and scroll down until you see the **Load Balancing** section

▼ **Load Balancing**

Load Balancers

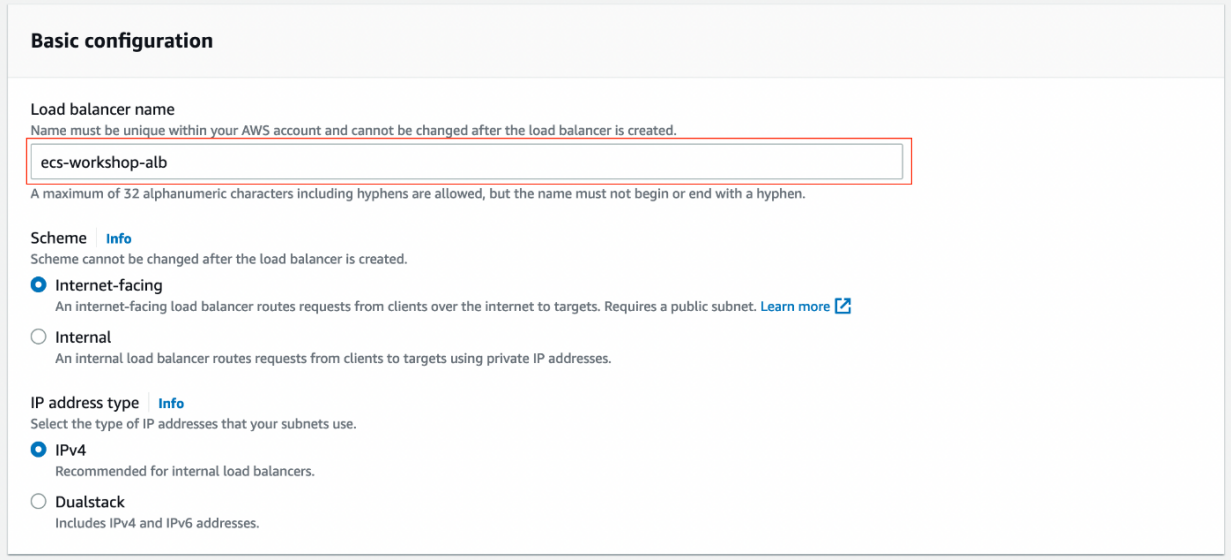
Target Groups

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

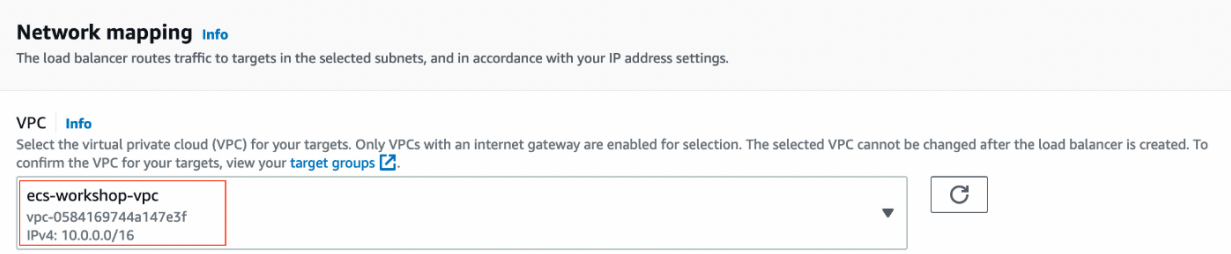


3) On the **Create Application Load Balancer** screen, configure the following

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

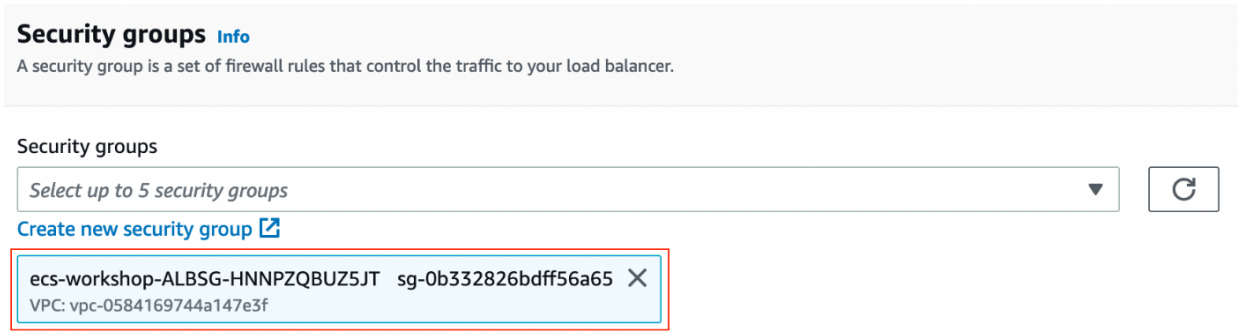


4) For **Network mapping**, select **ecs-workshop-vpc**

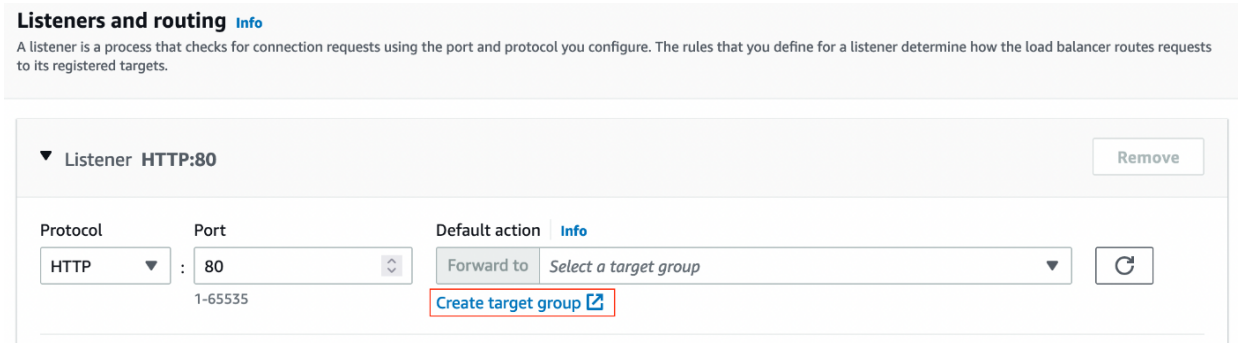


For **Mappings** select two Availability Zones and use the two **PublicSubnet1** and **PublicSubnet2** subnets.

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

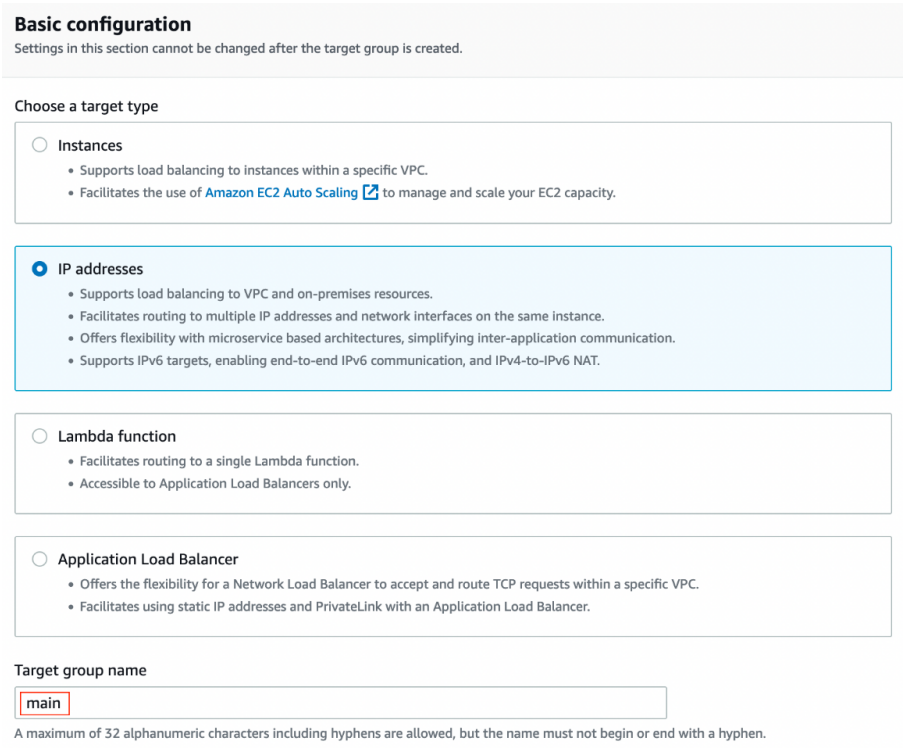


6) For **Listeners and routing** click on **Create target group**



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**

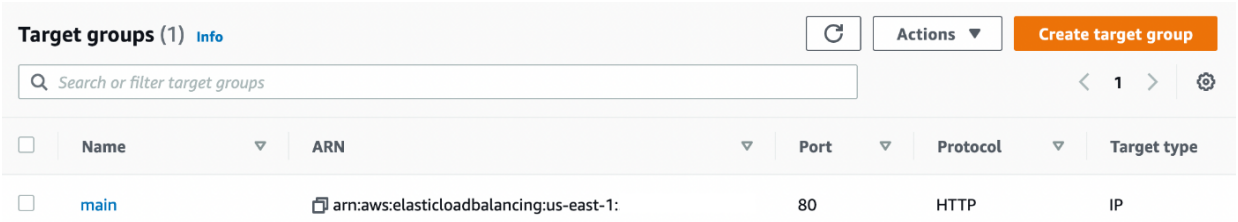


Scroll down and click **Next** to navigate to the next screen

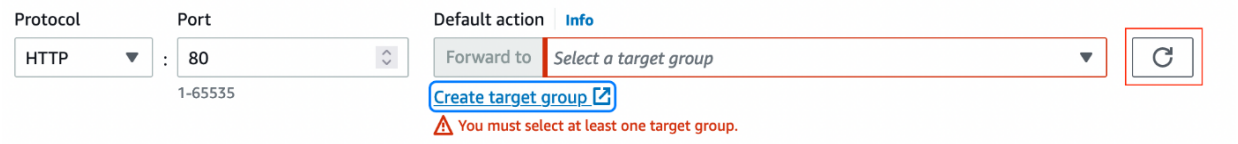
8) Click **Remove** to remove the IPv4 address. These addresses are automatically added by ECS so we don't need anything configured here.

Scroll down and click on Create Target Group

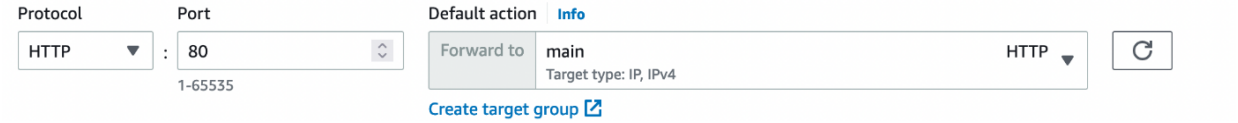
You should see the Target Group available.



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.

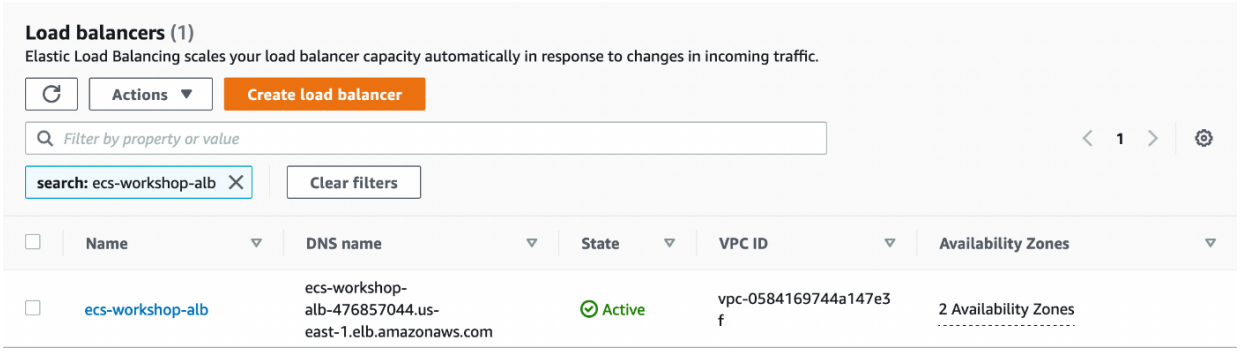


Once refreshed, you can select the **main target group** you created.



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

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



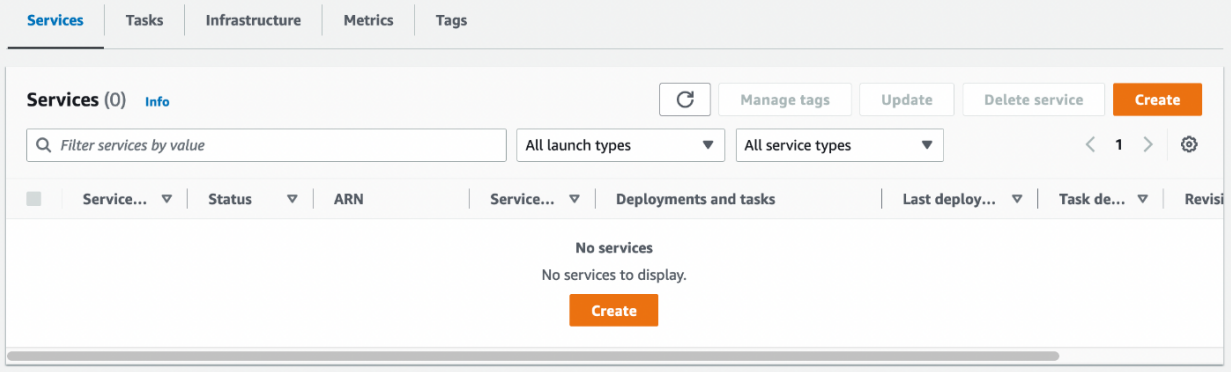
Creating the ECS Services for the Blue Red App

A quick recap before we start the next section

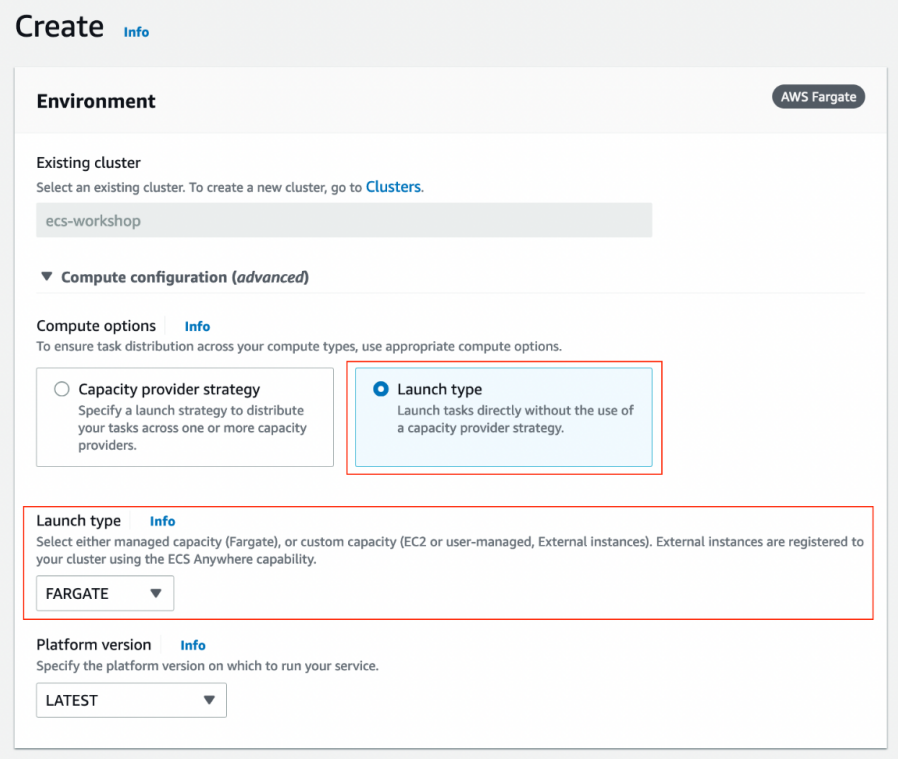
- An **ECS service** is a configuration you can use to run and maintain a specified number of tasks simultaneously in a cluster
- An **ECS task definition** is a blueprint for your application (which are then run as tasks on a given service). Task definitions specify various parameters of your application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

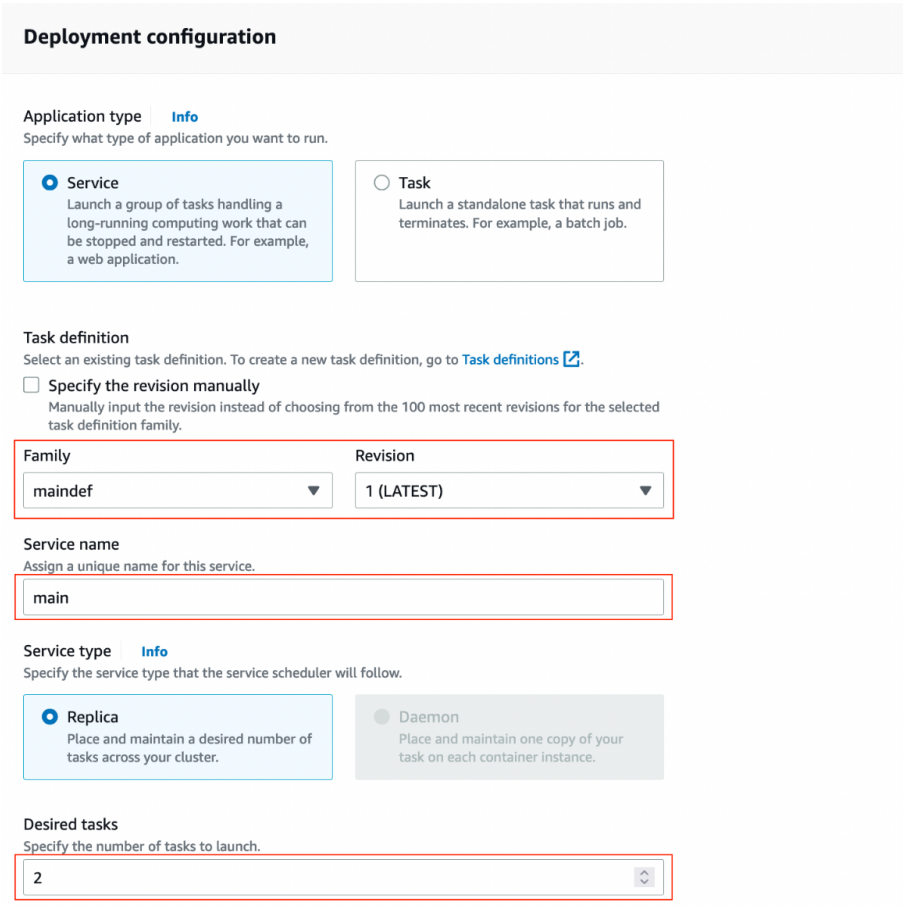
1) Navigate to the ECS console and click on the **ecs-workshop** ECS cluster, navigating your way to the Service section as shown below.



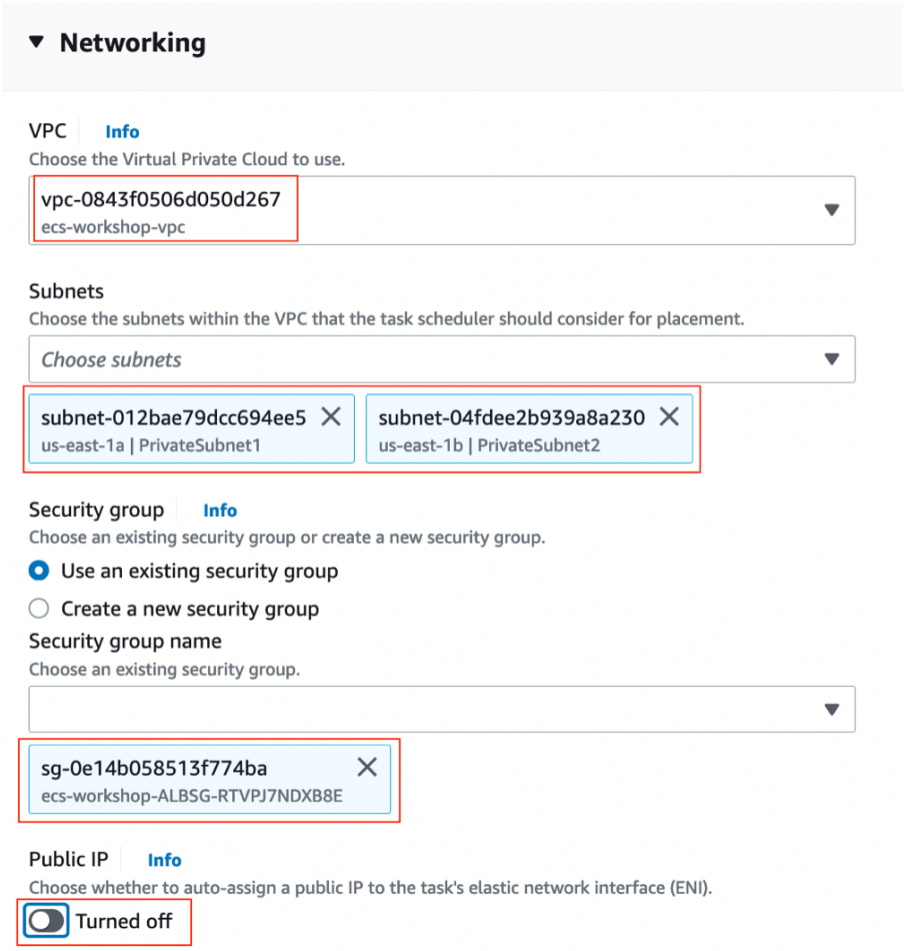
2) Click on Create and select **Launch type** and make sure **FARGATE** is selected as shown below.



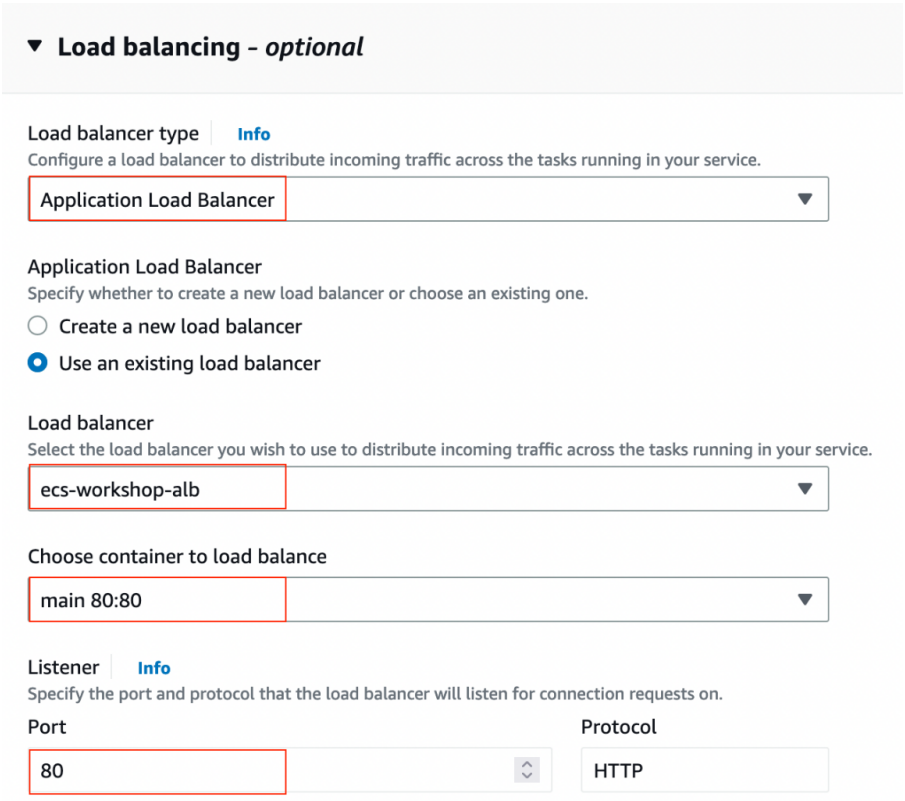
3) Scroll down to the **Deployment configuration** and add select **Service**, **Family** as **maindef** and the **Desired tasks** as **2**.



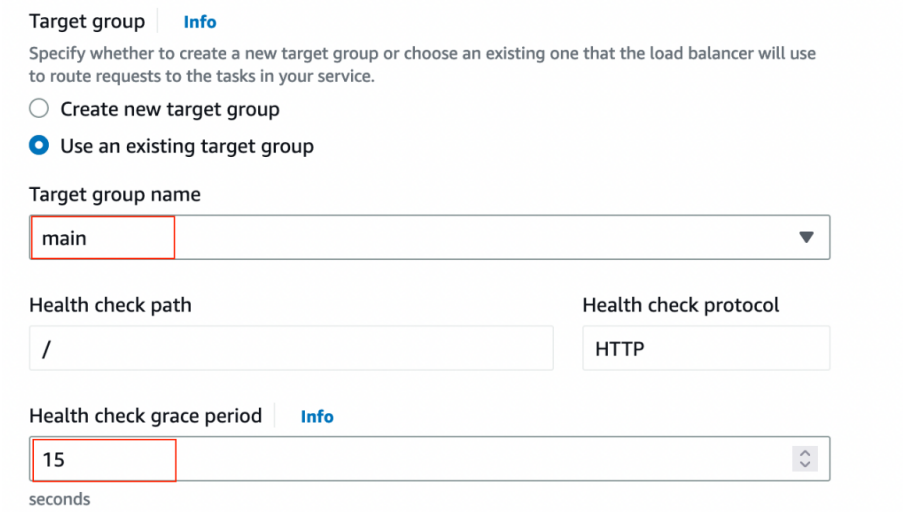
4) Configure the network as shown below. For the VPC, select the **ecs-workshop-vpc** and the subnets, select **PrivateSubnet1** and **PrivateSubnet2**. Use the existing security group starting with **ecs-workshop-ALB** and make sure **Public IP** is **Turned off.**



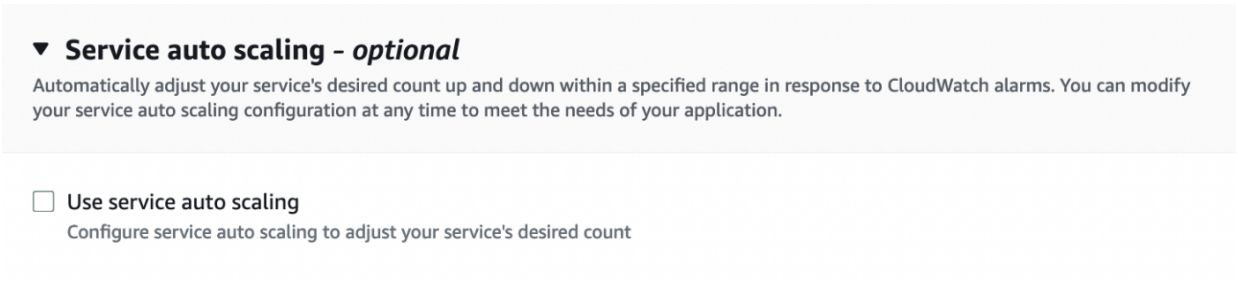
5) Under **Load balancing**, for the **Load balancer** type choose **Application Load Balancer** and configure **ecs-workshop-alb** as the **Load balancer name.** Make sure the **main 80:80** is selected as the container to load balance. Check the **Listener** is configured to use port **80**. Use an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



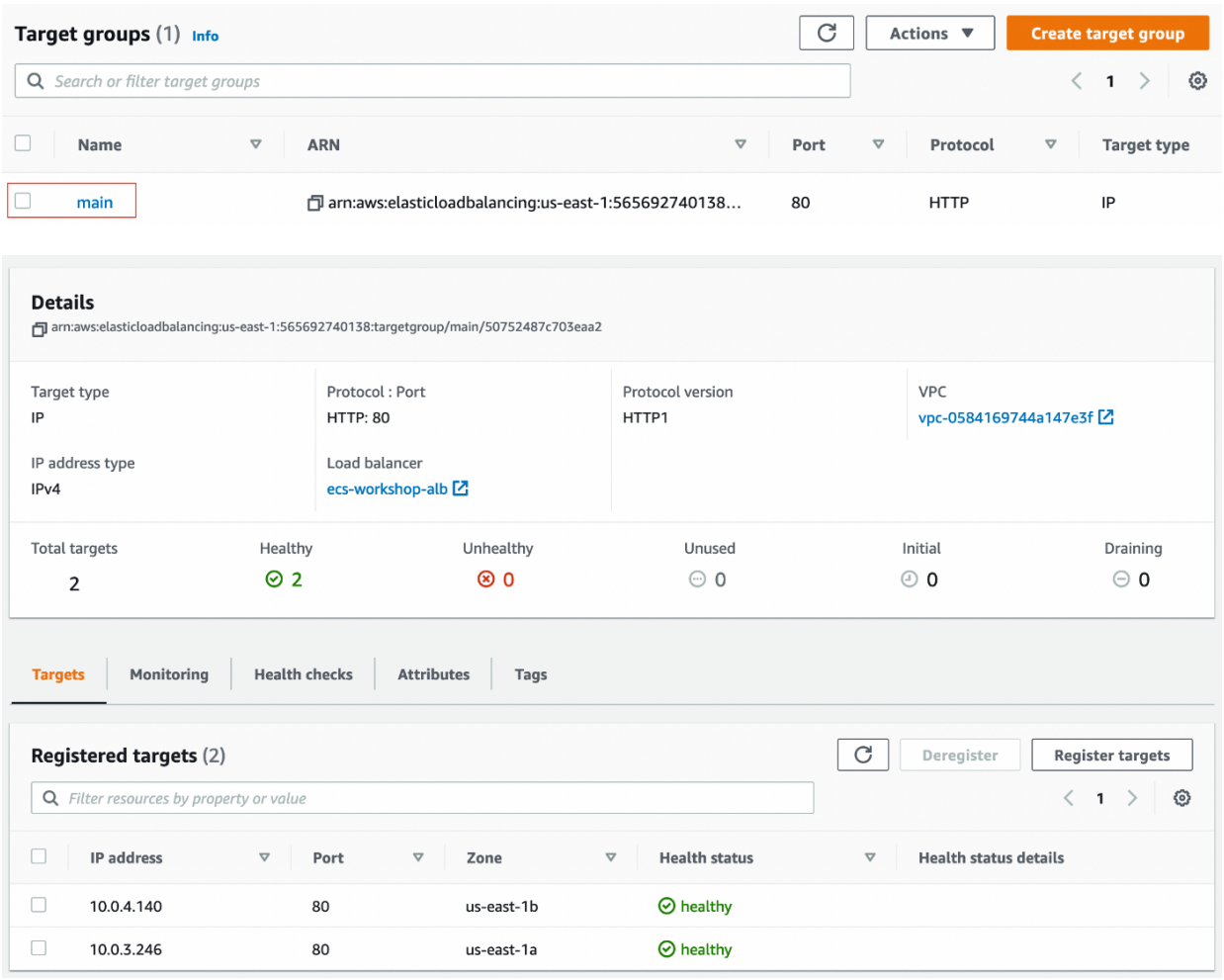
6) For **Target group name** select **Use an existing target group**, name the **Target group name** as **main.**



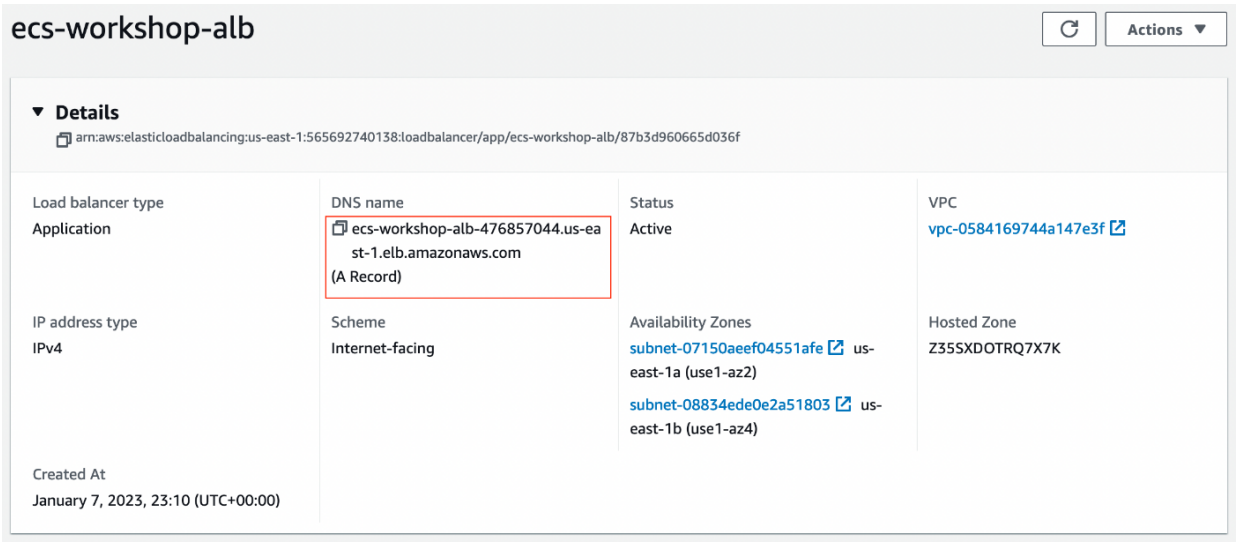
7) Leave the **Service auto scaling - *optional*** checkbox as empty and click **Create**.



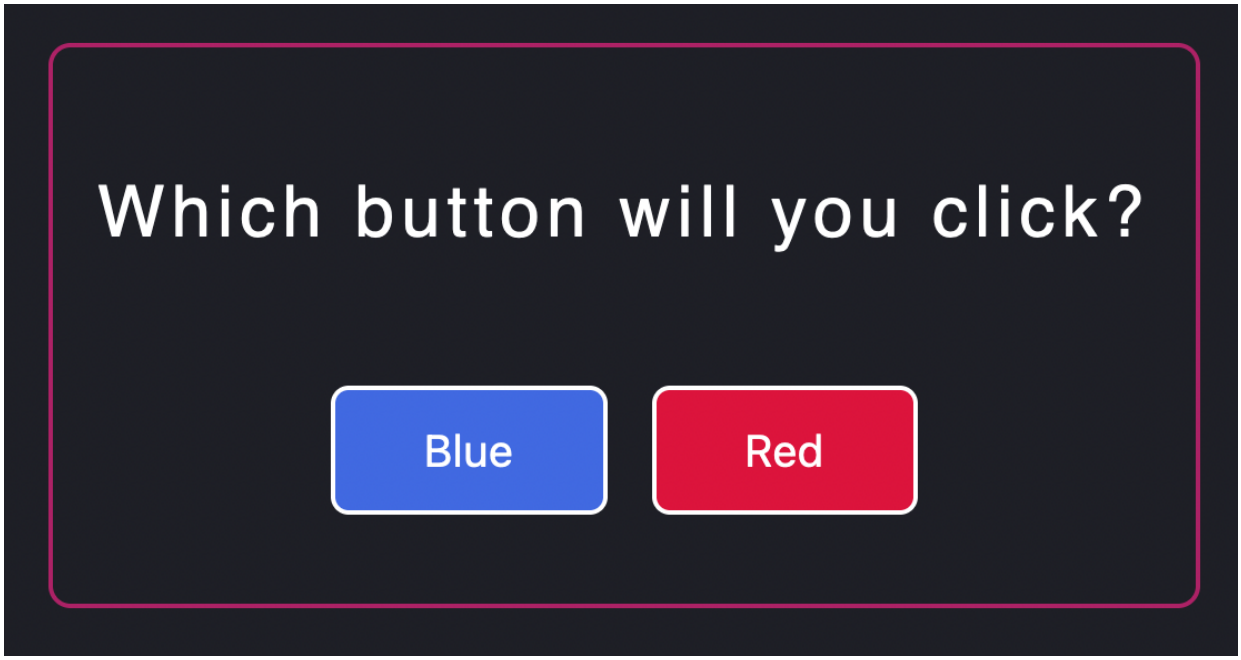
You can also check the Registered targets **in the Application Load Balancer** to make sure the targets have registered successfully. Navigate to the EC2 console and click on Target Groups. Click on the **main** Target Group and check the Targets are **Healthy.**



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.



The following page should load in the browser.

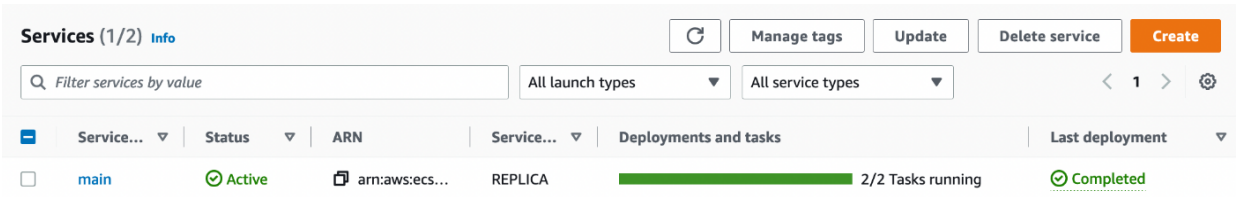


Congratulations! This completes setting up the main ECS service.

Create the Blue Service

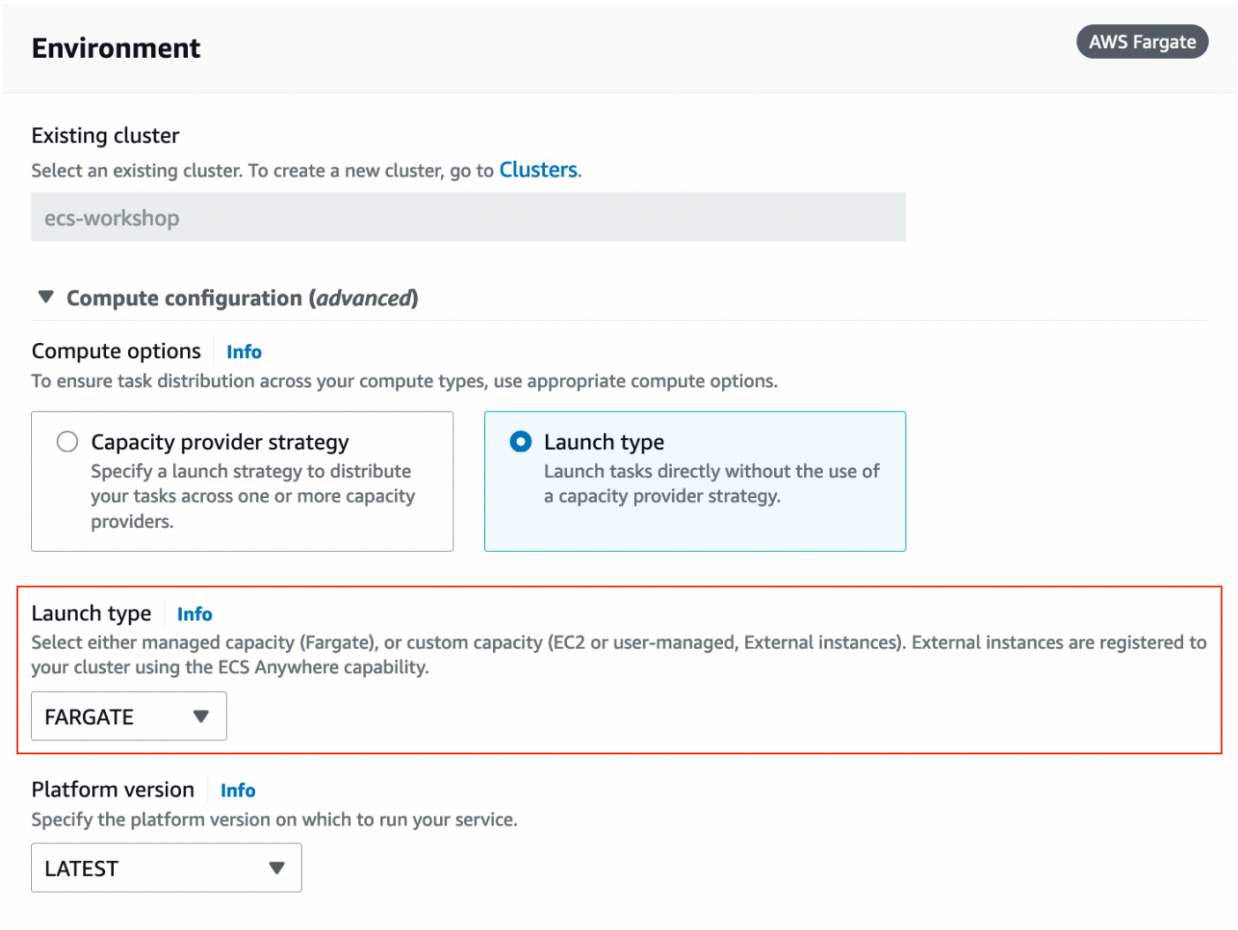
Now we need to set up the **blue** service. We'll reuse the same steps to create the **red** service once we've completed the **blue** service setup.

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

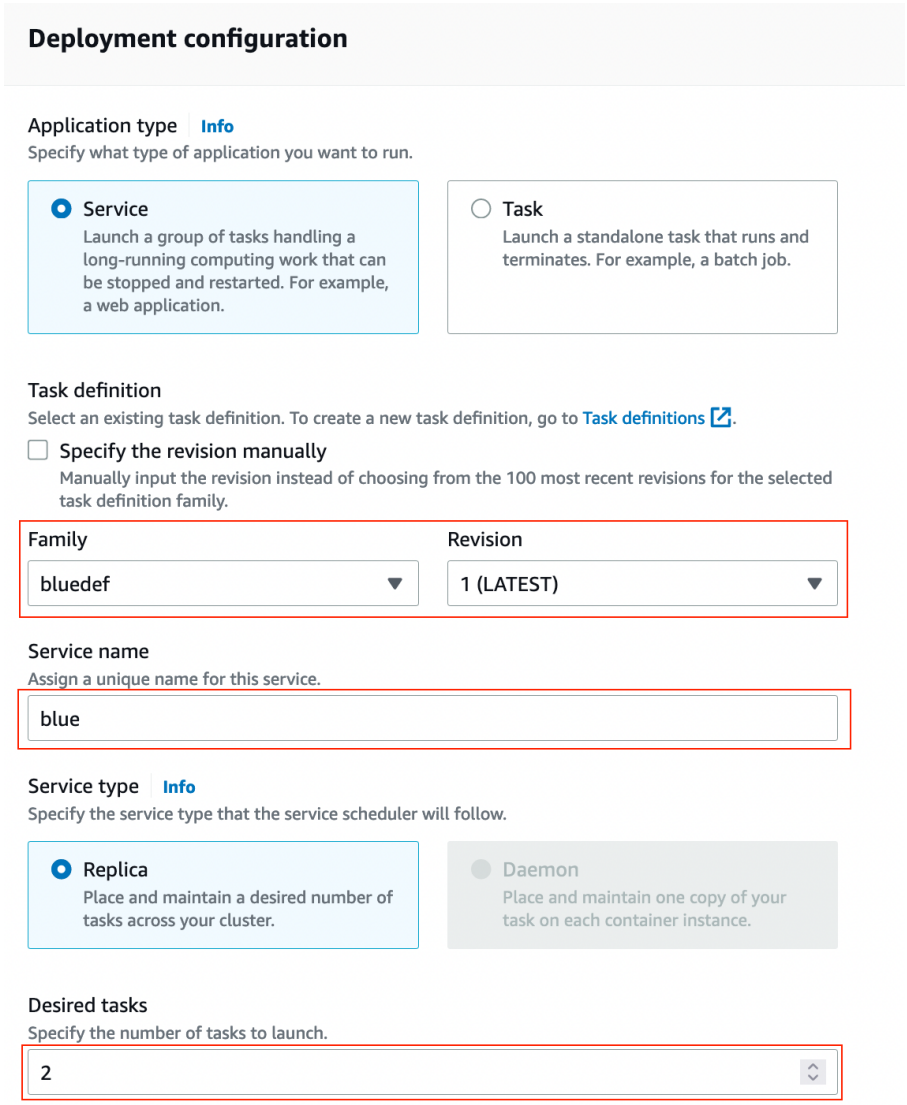


2) Click on Create and select **Launch type** and make sure **FARGATE** is selected as shown below.

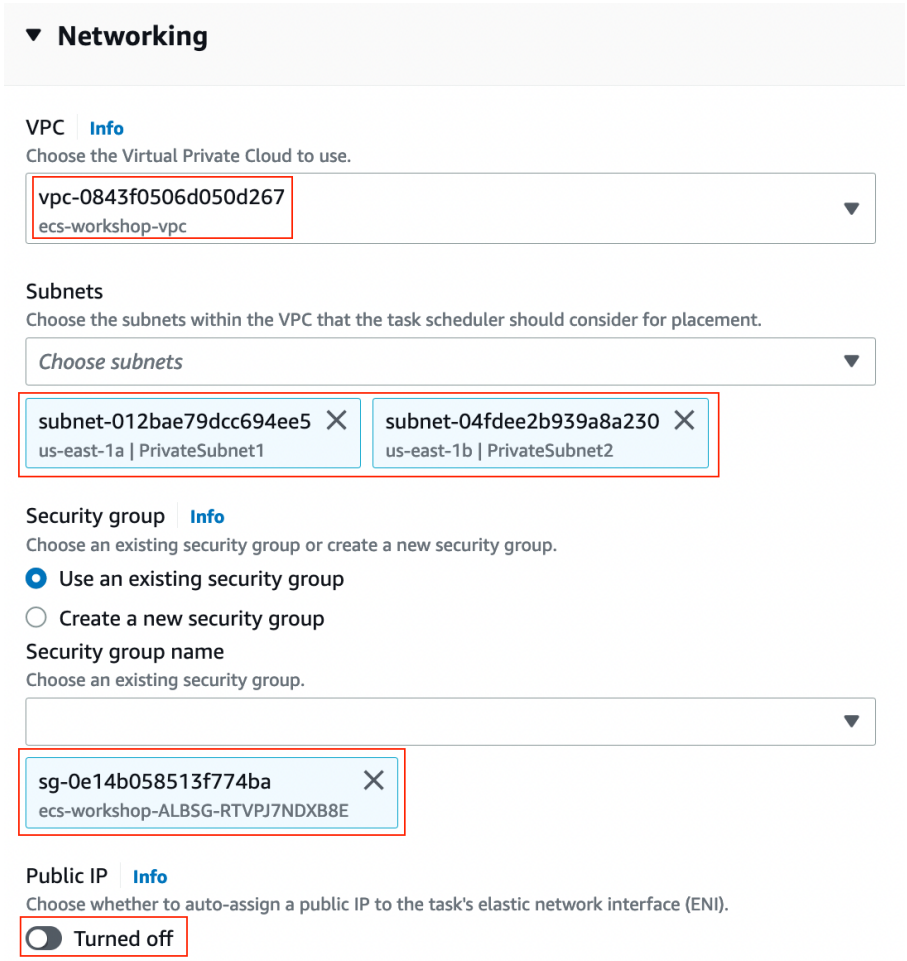
For the **Launch Type**, click on **Fargate**



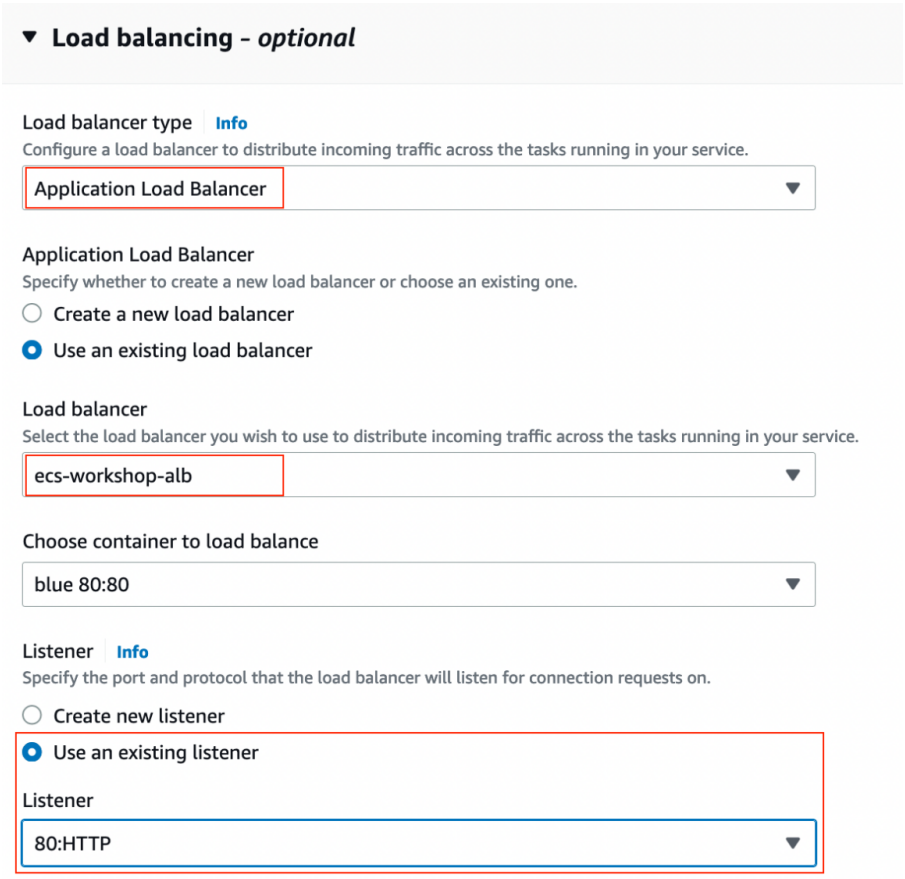
3) Scroll down to the **Deployment configuration** and add select **Service**, **Family** as **bluedef,** the **Service name** as **blue** and the **Desired tasks** as **2**.



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.**

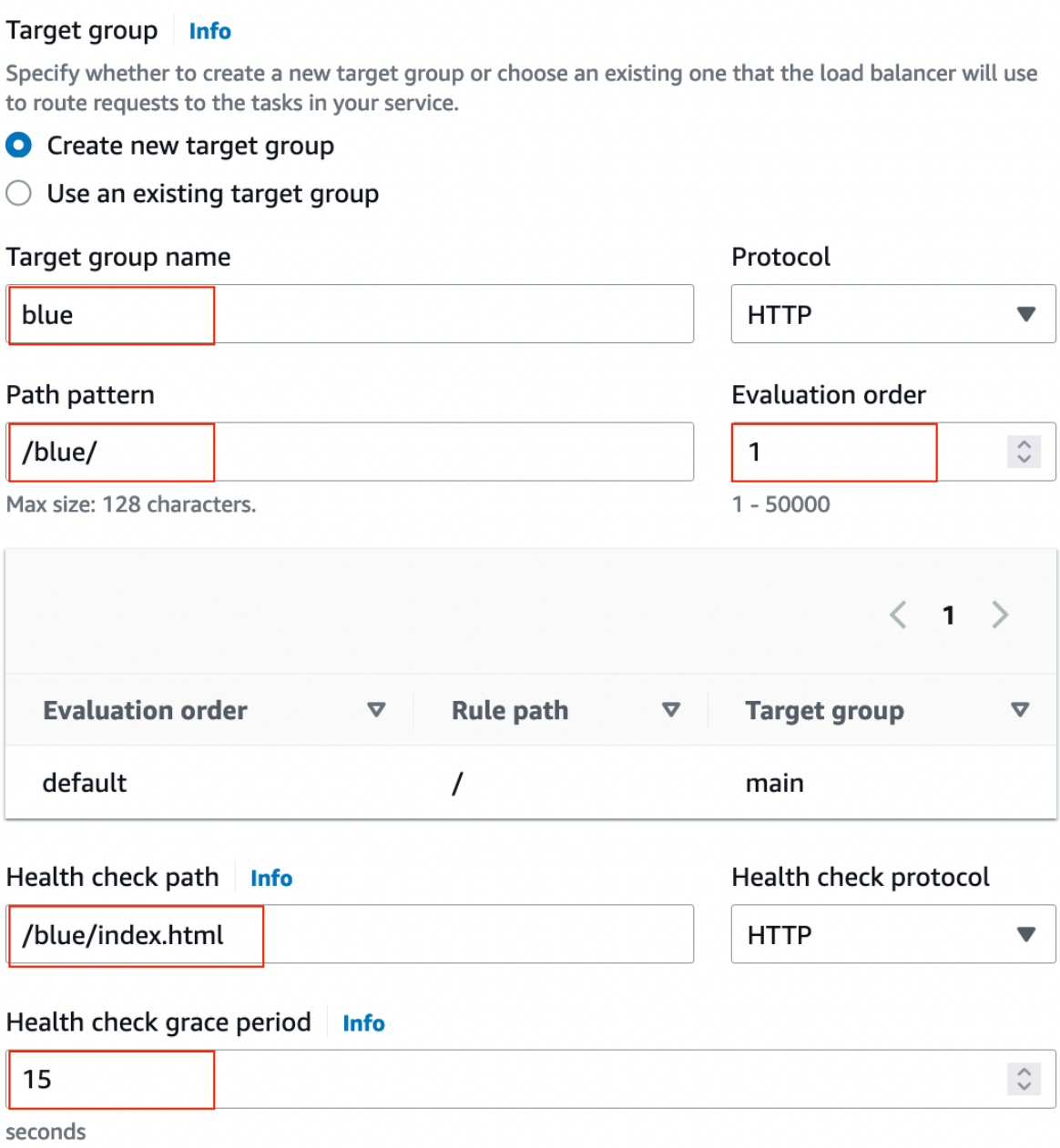


5) Under **Load balancing**, for the **Load balancer** type choose **Application Load Balancer** and configure **ecs-workshop-alb** as the **Load balancer name.** Make sure the **blue 80:80** is selected as the container to load balance. Check the **Listener** is configured to use port **80**. Use an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

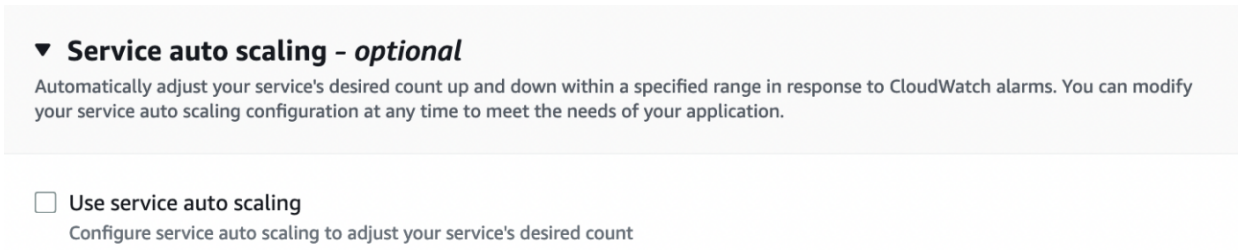


6) For **Target group name** select **create new** name the **Target group** name as **blue**, **Path pattern** as **/blue/** the **Evaluation order** as **1**, the **Health check path** as **/blue/index.html** and the Health check grace period as **15.**

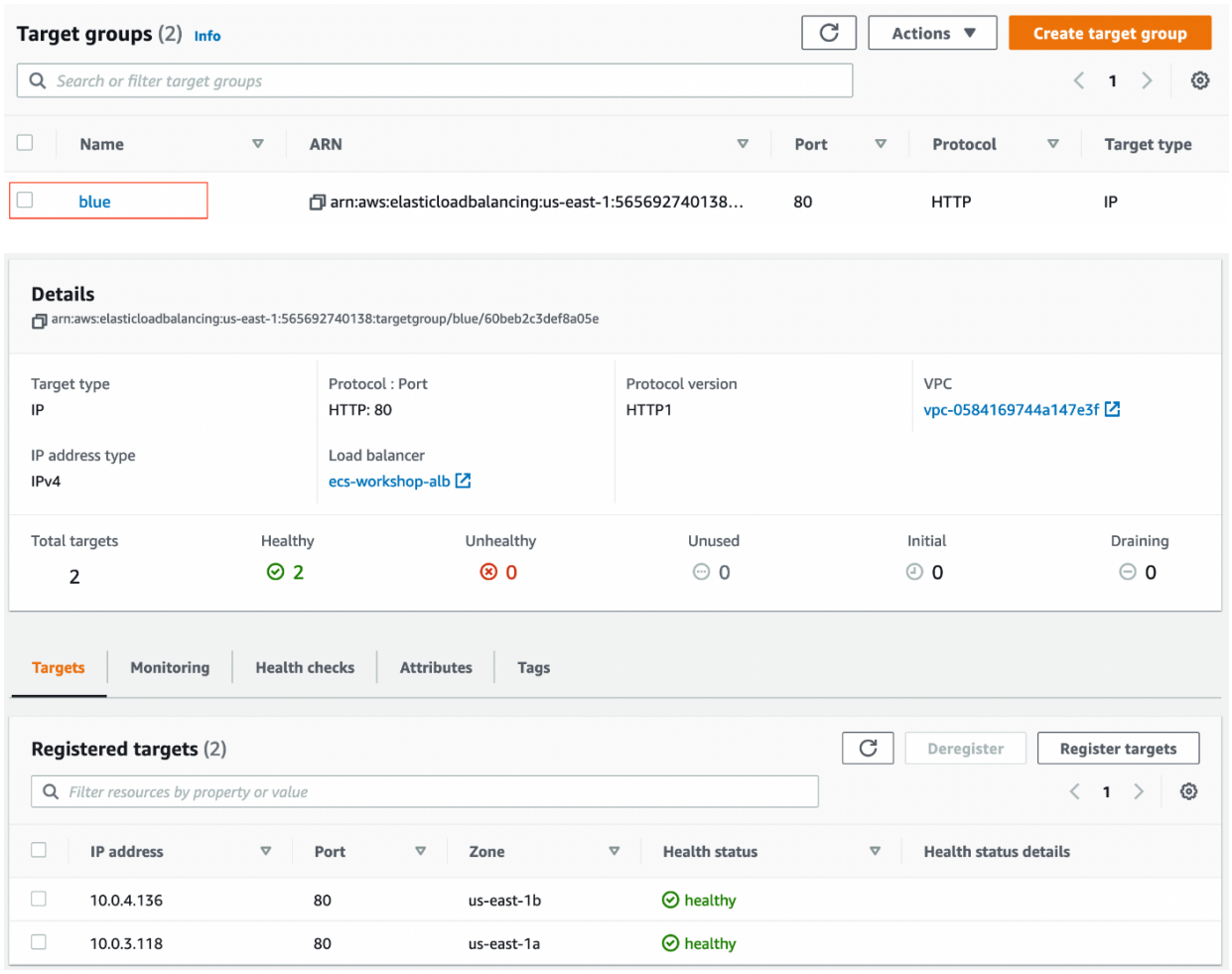
IMPORTANT NOTE: You must configure the **Path pattern** as **/blue/** and **Health check path** as **/blue/index.html** otherwise the checks will fail



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



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.**

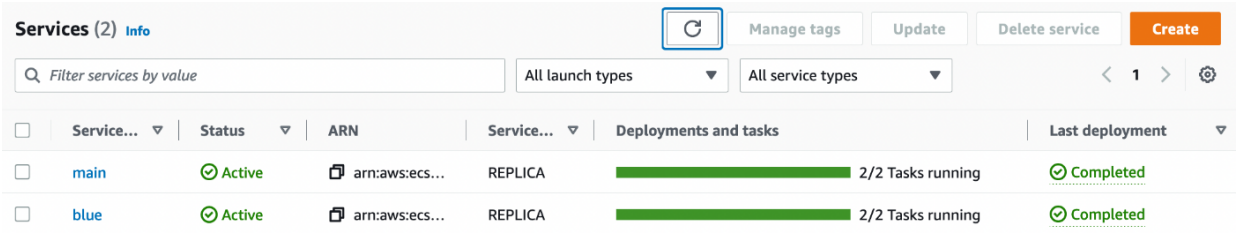


Now the **blue** service is up and running and the Health status is **healthy**, we need to repeat the steps for the **red** service.

Create the Red Service

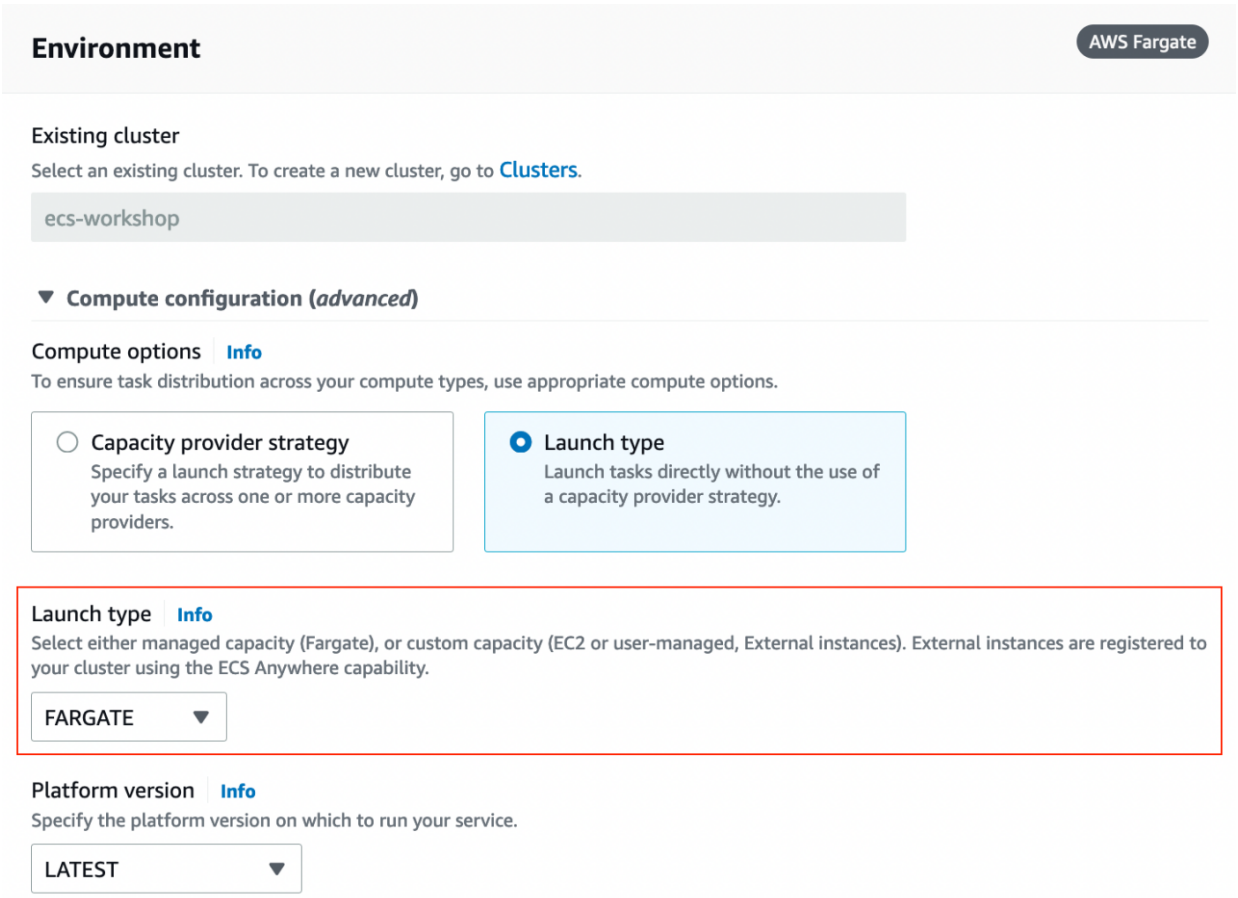
Now we need to set up the **blue** service. We'll reuse the same steps to create the **red** service once we've completed the **blue** service setup.

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

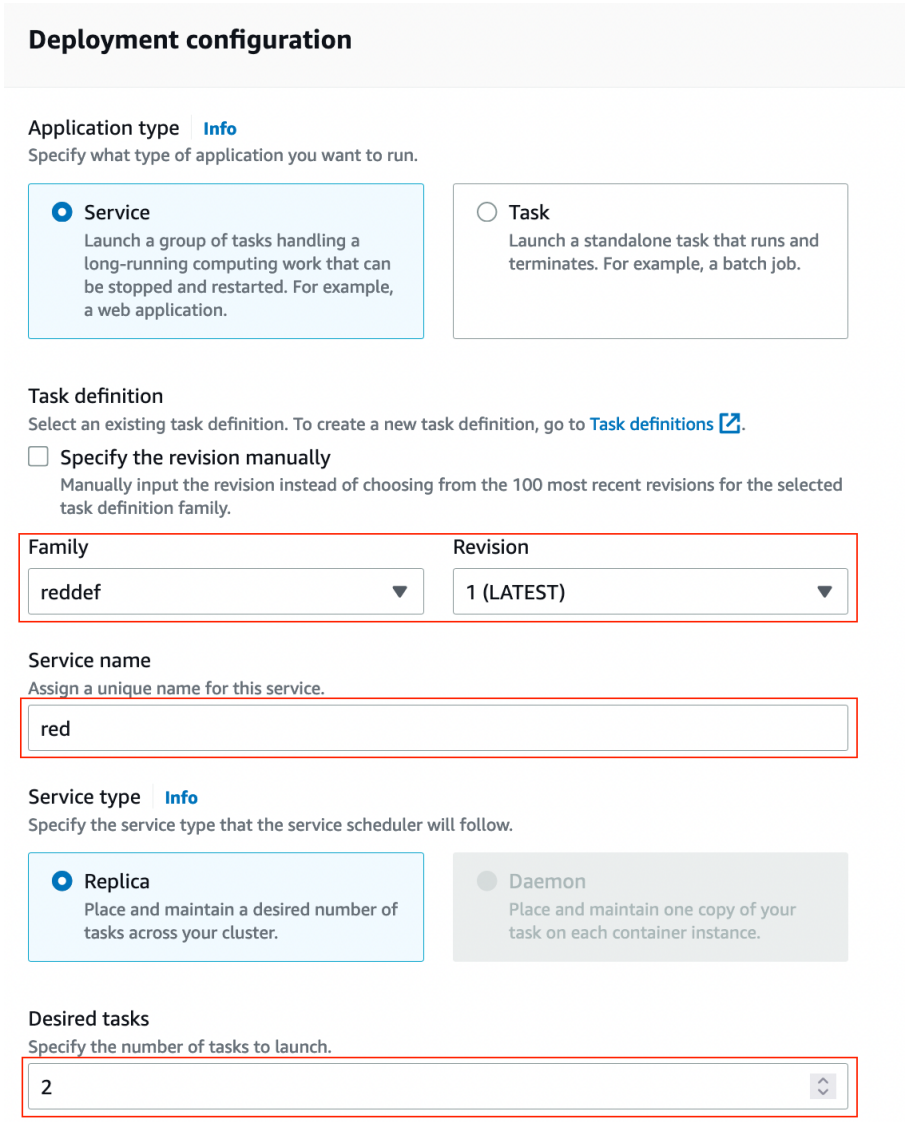


2) Click on Create and select **Launch type** and make sure **FARGATE** is selected as shown below.

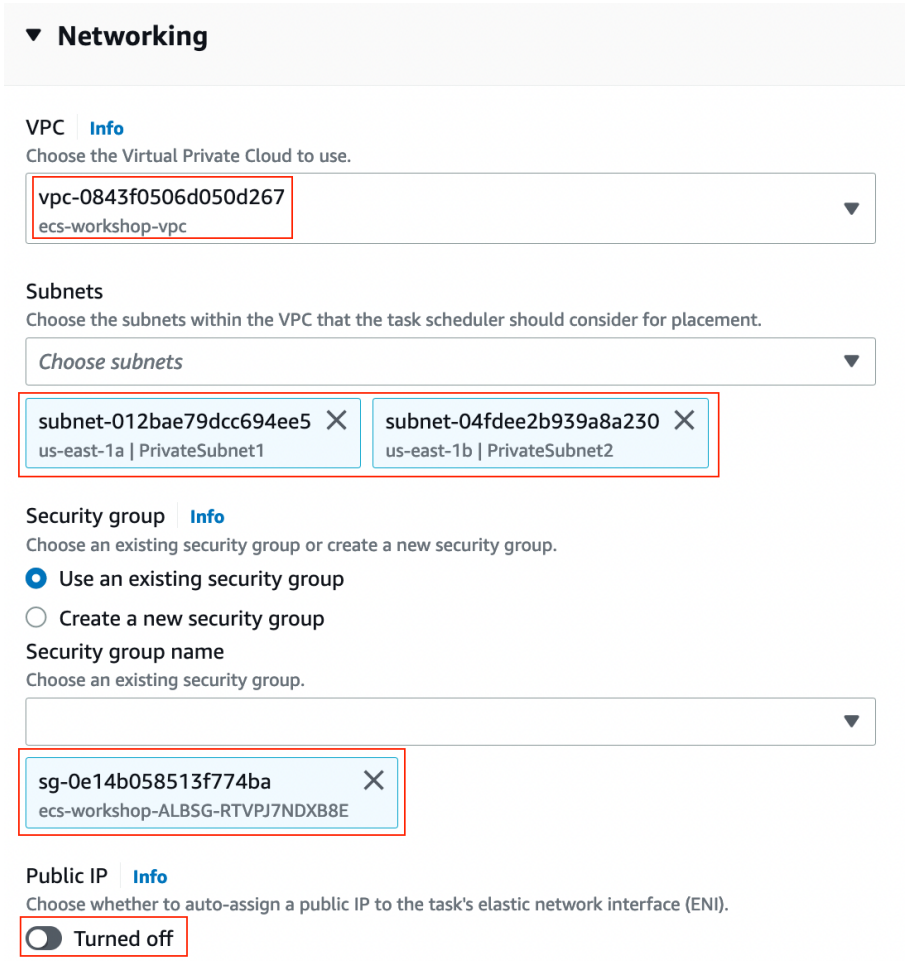
For the **Launch Type**, click on **Fargate**



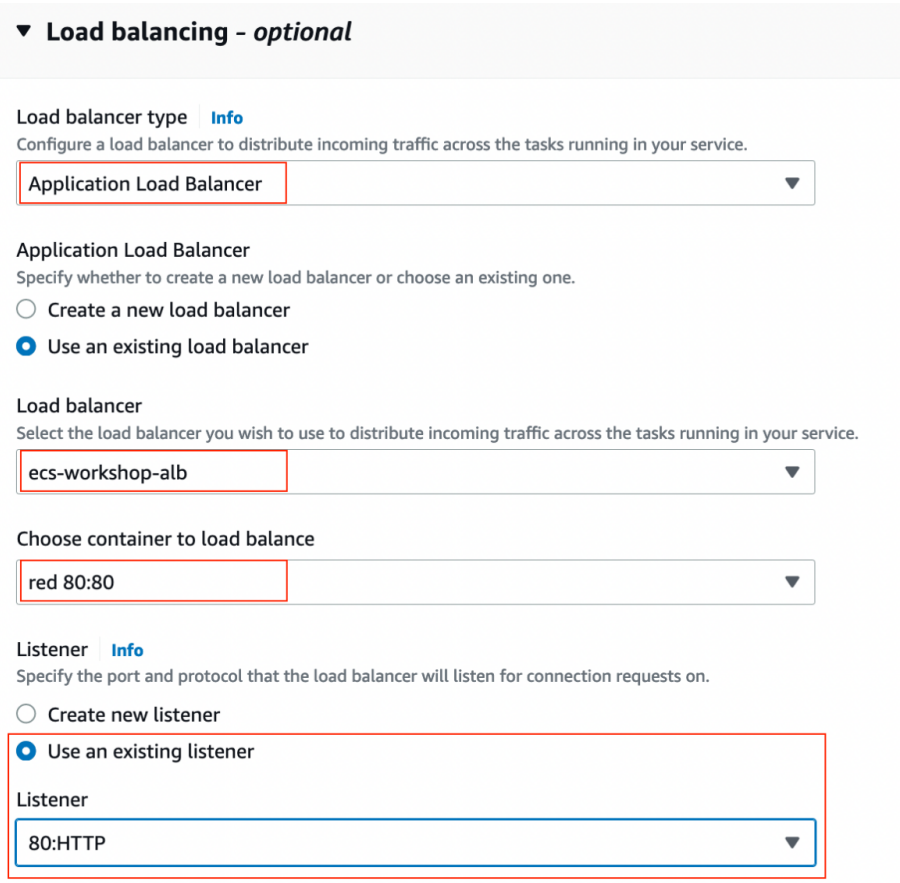
3) Scroll down to the **Deployment configuration** and add select **Service**, **Family** as **reddef,** the **Service name** as **red** and the **Desired tasks** as **2**.



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.**

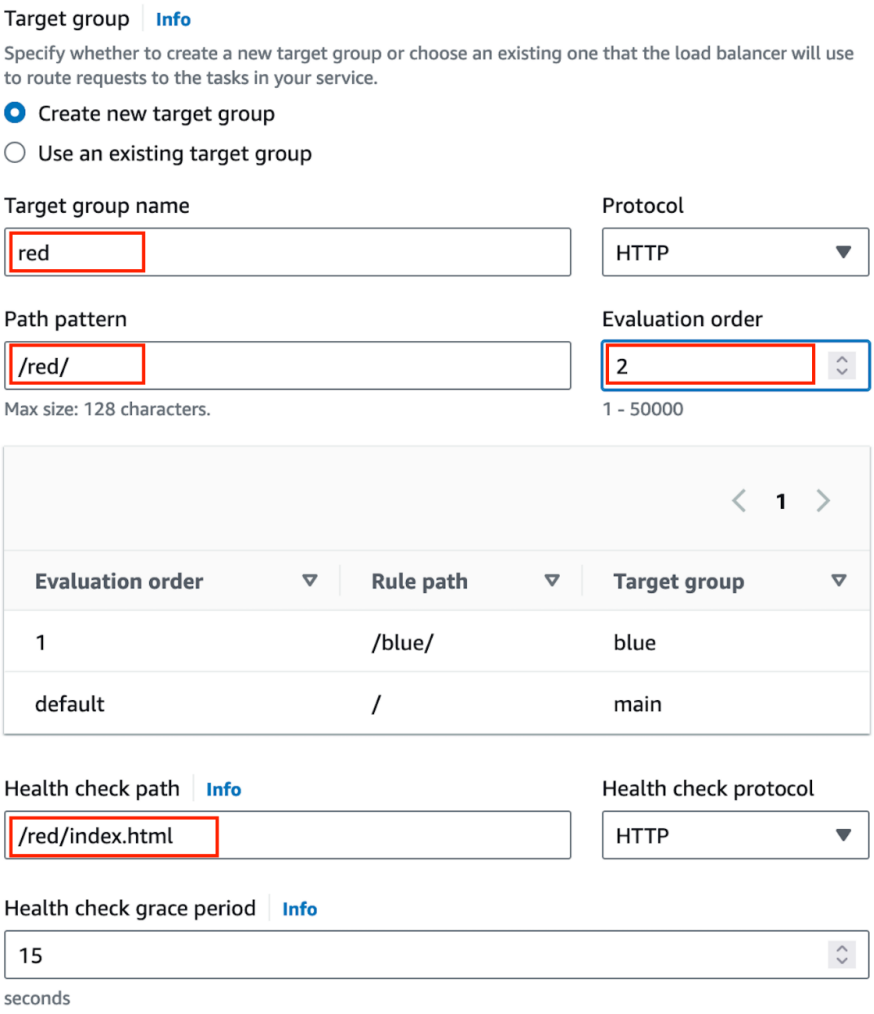


5) Under **Load balancing**, for the **Load balancer** type choose **Application Load Balancer** and configure **ecs-workshop-alb** as the **Load balancer name.** Make sure the **red 80:80** is selected as the container to load balance. Check the **Listener** is configured to use port **80**. Use an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

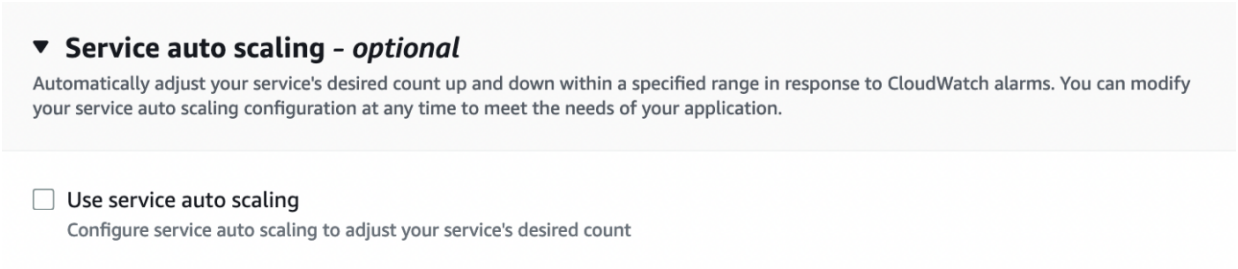


6) For **Target group name** select **create new** name the **Target group** name as **red**, **Path pattern** as **/red/**, the **Evaluation order** as **2**, the **Health check path** as **/red/index.html** and the Health check grace period as **15.**

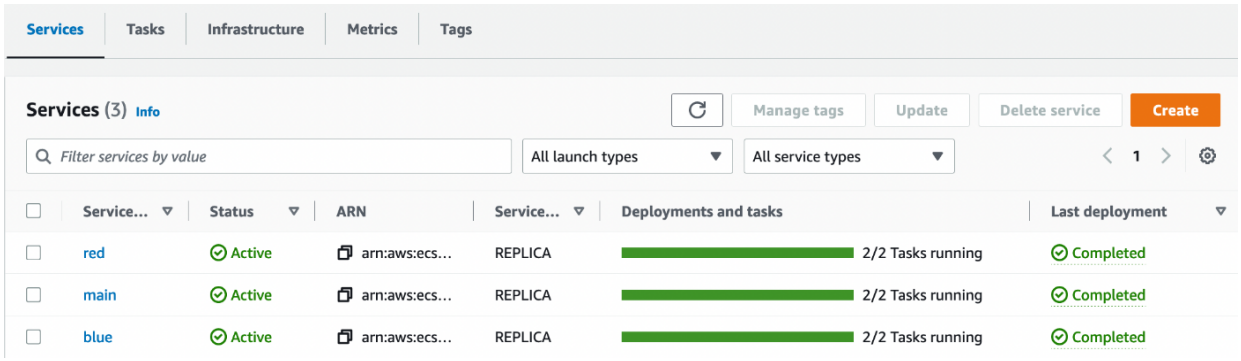
IMPORTANT NOTE: You must configure the **Path pattern** as **/red/** and health check **/red/index.html** otherwise the checks will fail.



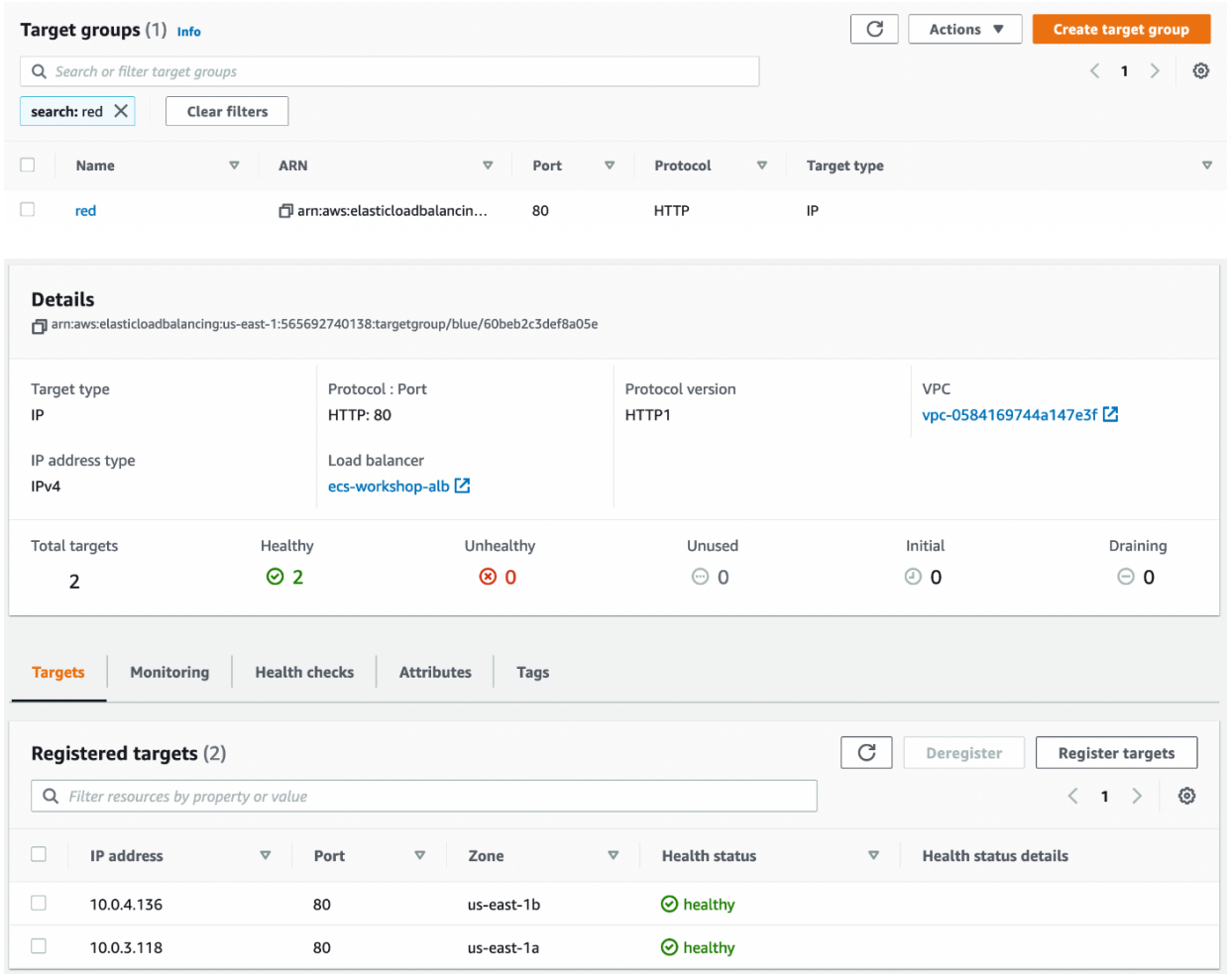
7) Leave the **Service auto scaling - *optional*** checkbox as empty and click **Create**.



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



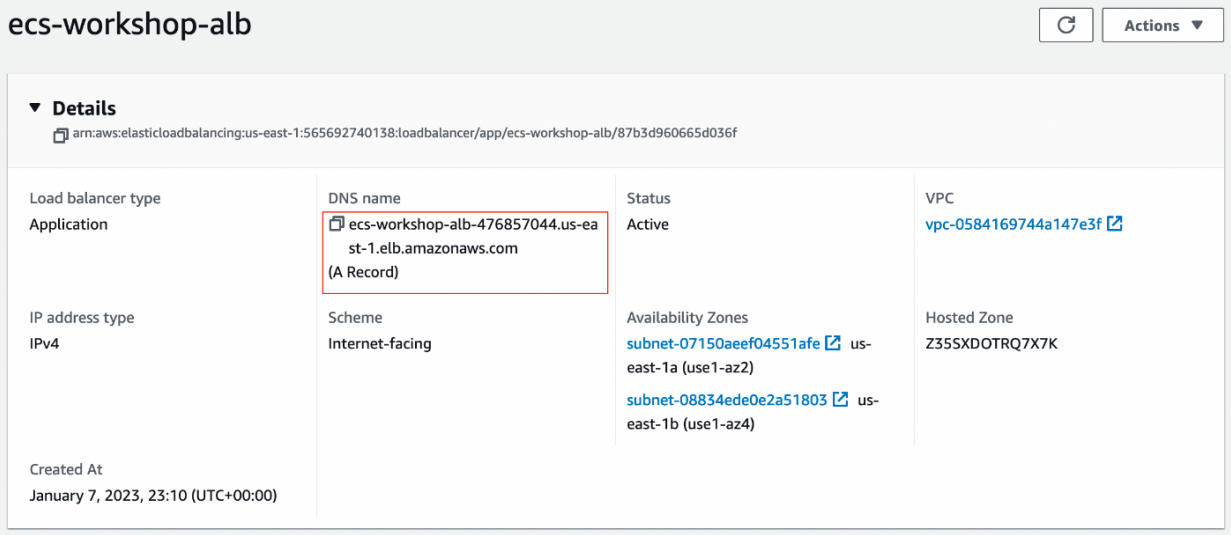
As you did previously for the **blue** and **main** service, you can also check the Registered targets **in the Application Load Balancer** to make sure the **red** targets have registered successfully. Navigate to the **Load Balancers** in the EC2 console and click on **Target Groups**. Click on the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



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.



Container Insights

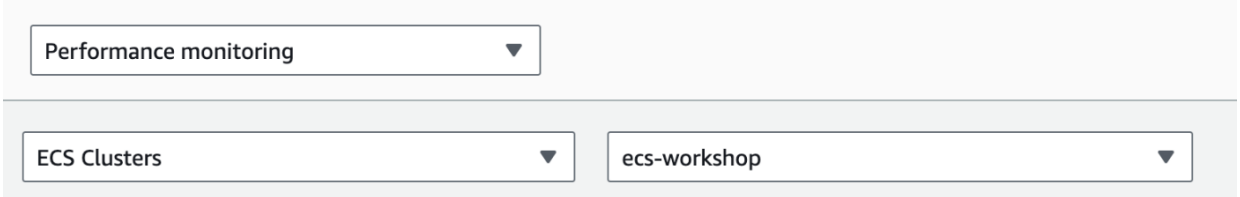
CloudWatch Container Insights to collect, aggregate, and summarize metrics and logs from your containerized applications and microservices. CloudWatch automatically collects metrics for many resources, such as CPU, memory, disk, and network. Container Insights also provides diagnostic information, such 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

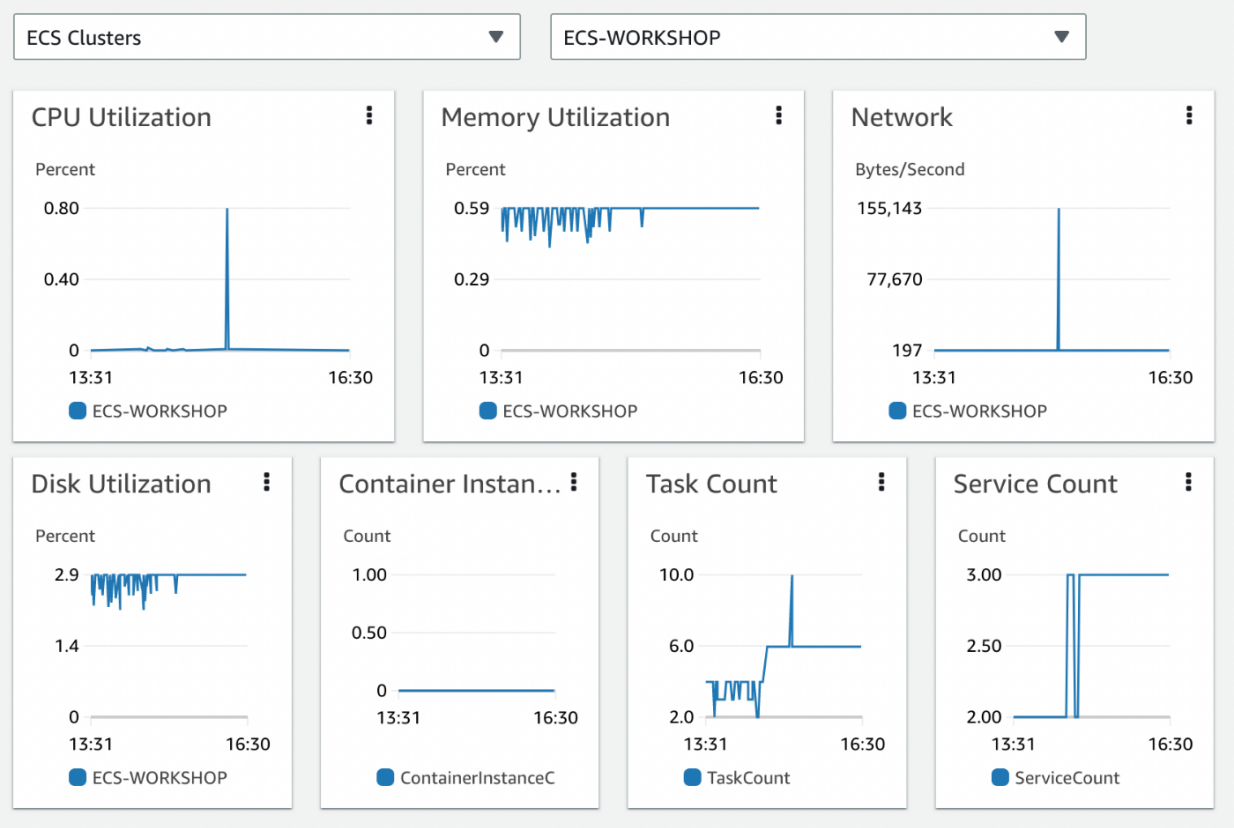
Container Insights was enabled when we created the ECS cluster. Let's take a look at what is configured.

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**

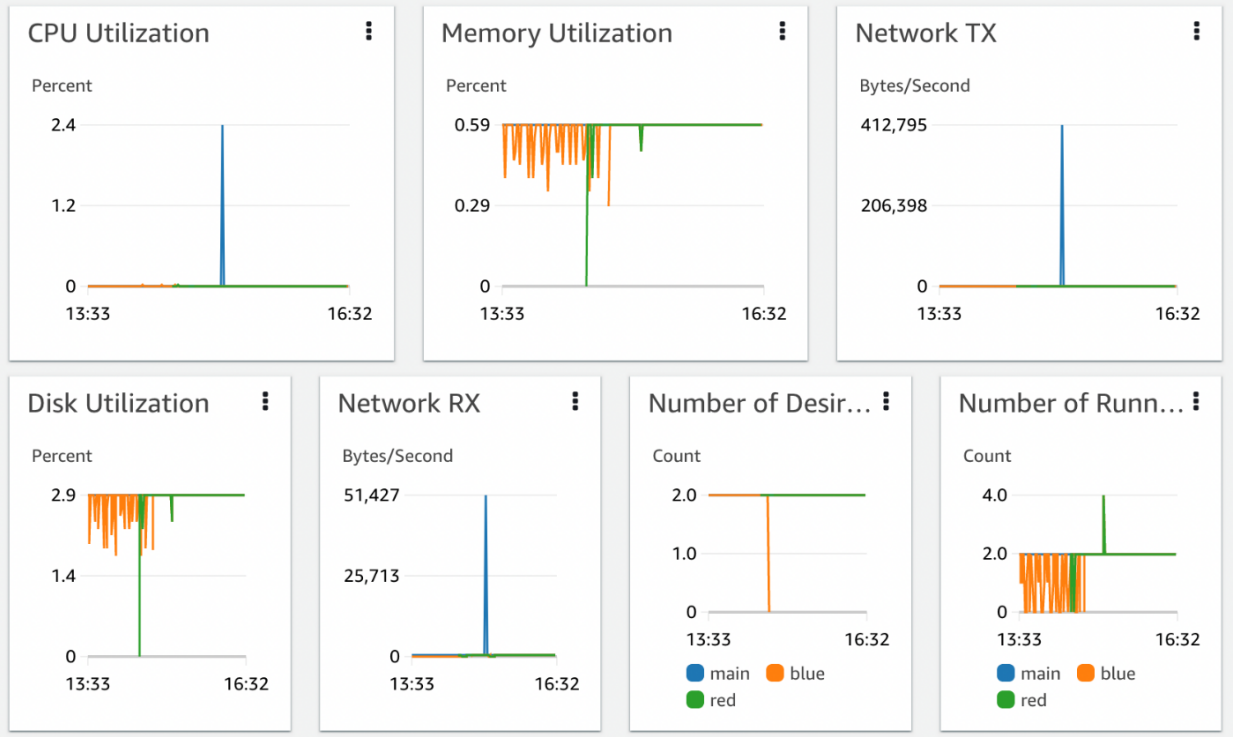


The following metrics for your **ECS Cluster** will be displayed

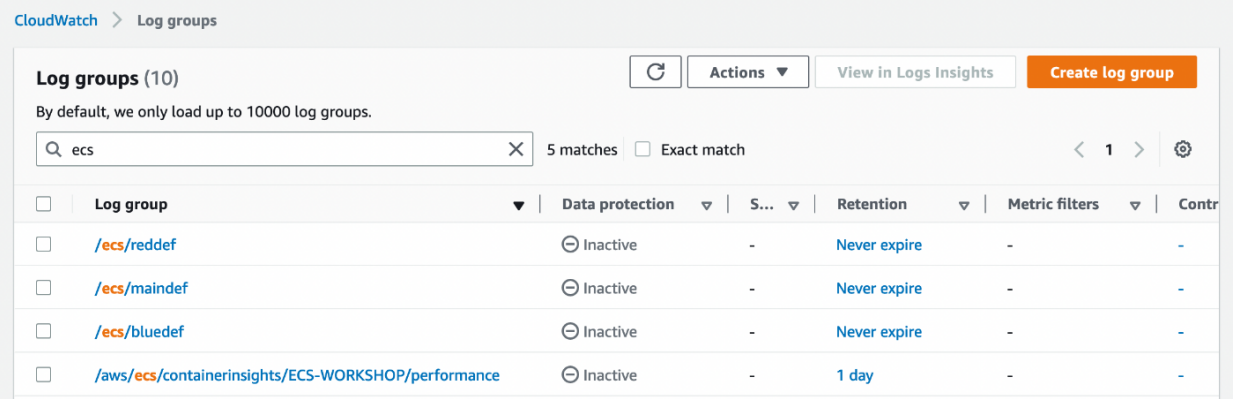


Click on the ECS Cluster drop down and select ECS Services for more information on ECS resources.

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



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



CloudWatch Alarms

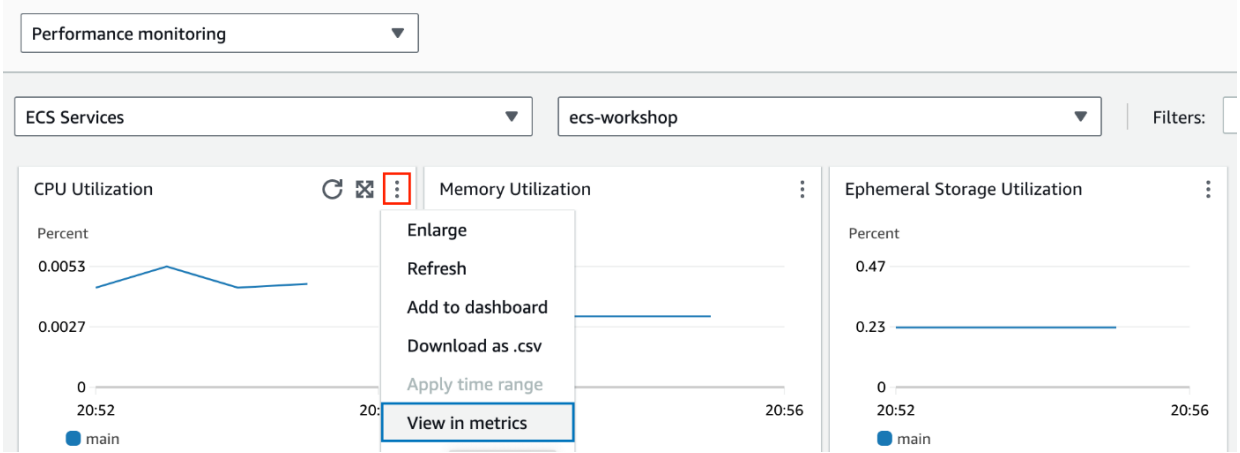
The metrics captured through Container Insights can be used to setup Alarms to get notified of any anomalies in the environment behavior.

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.

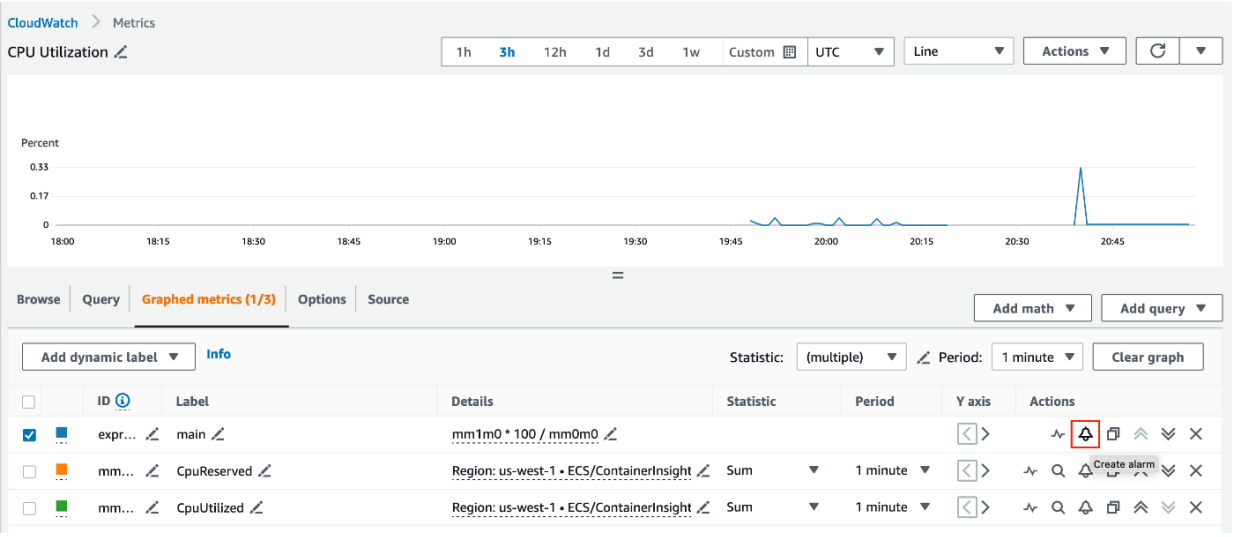
CloudWatch alarm setup

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**.

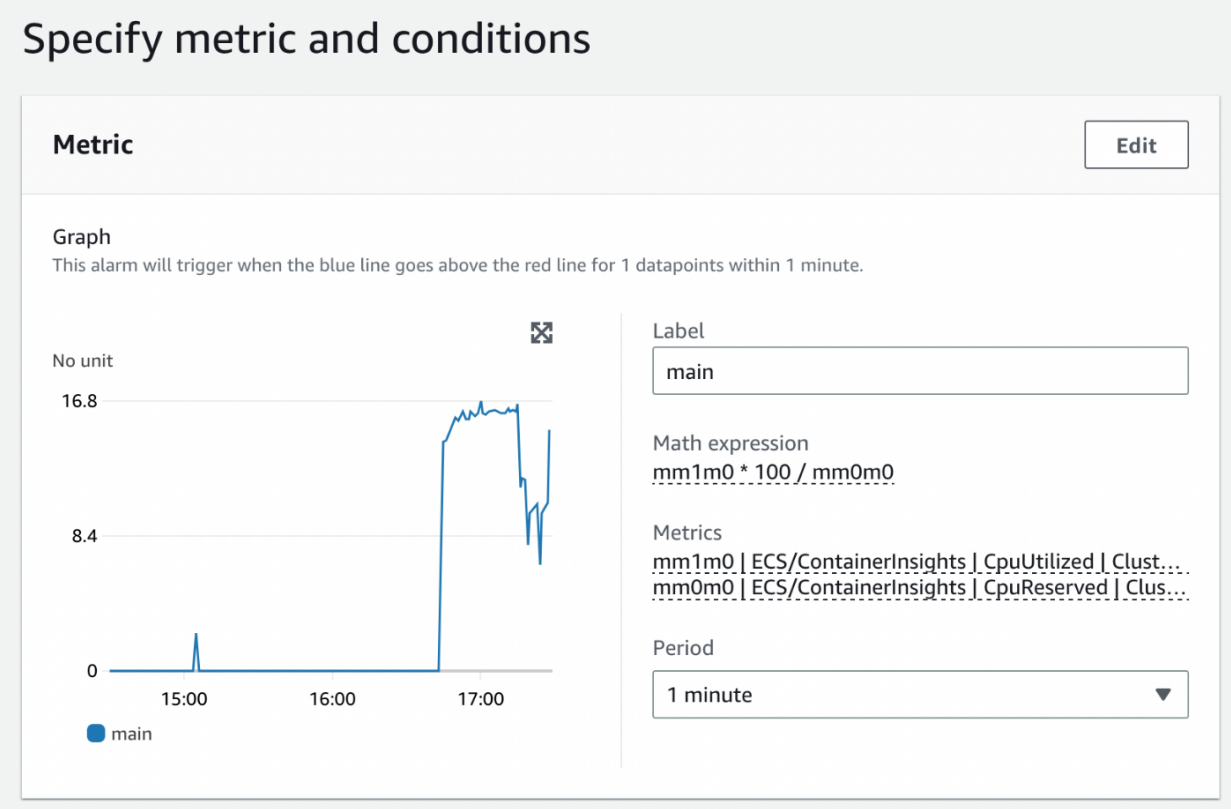


This will take you to a screen like the one below where you can configure an alarm

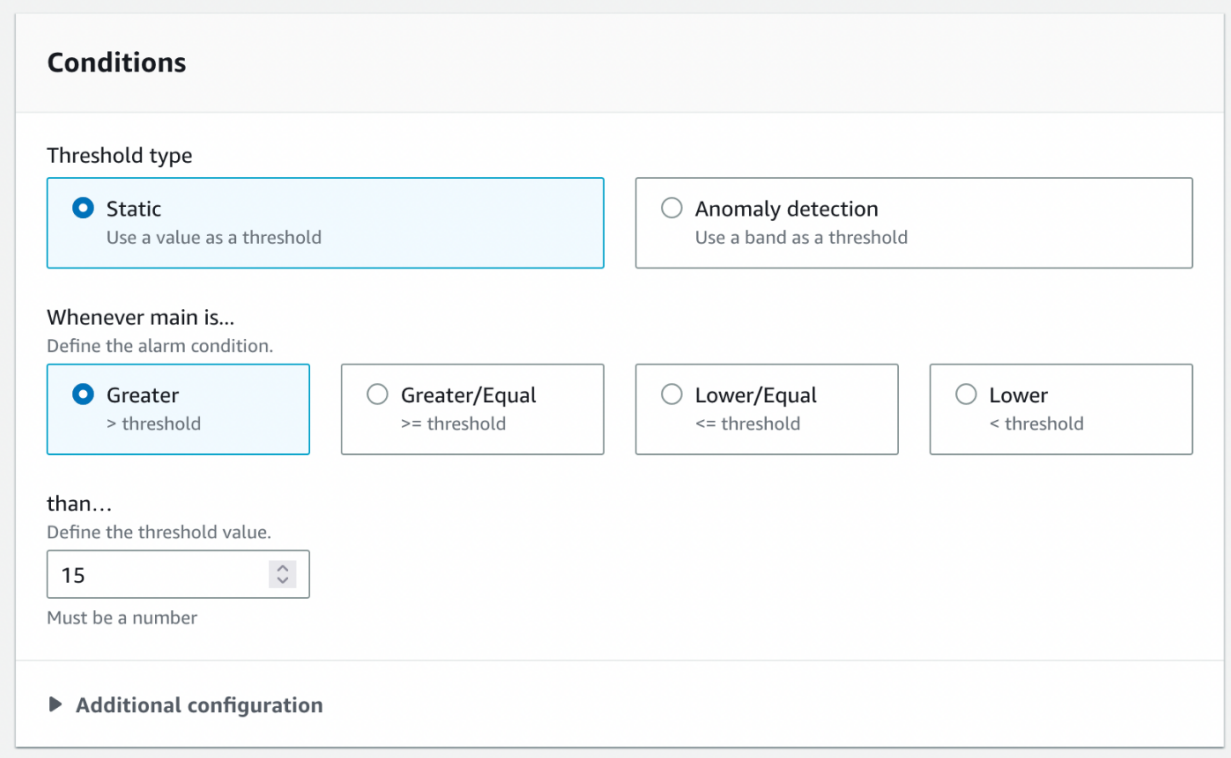


In the Graph metrics (x/y) section below the graph, click on the alarm bell 🔔 under actions.

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

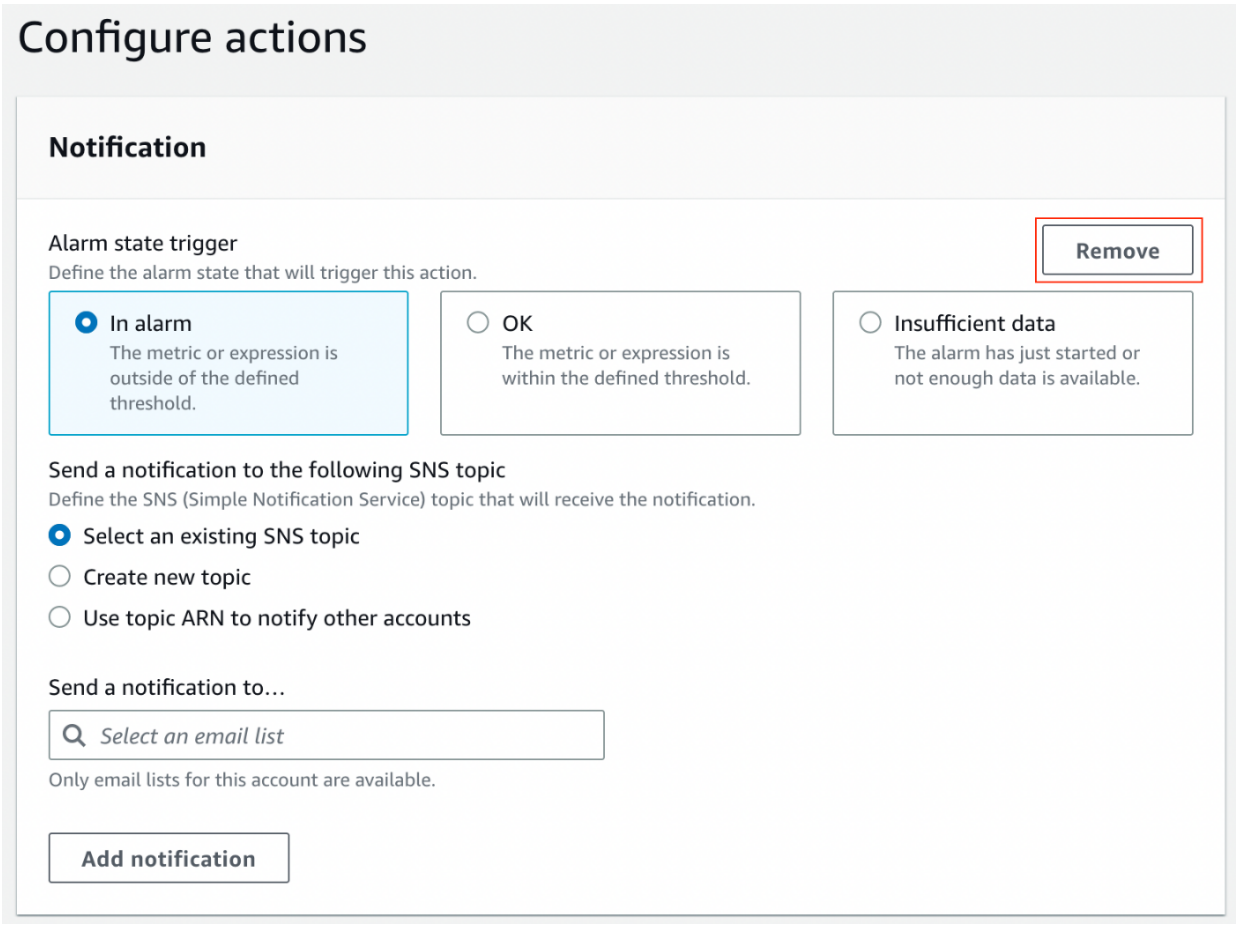


Leave everything as default and enter **15** in the **Define the threshold value** screen. By doing this, we are setting the CPU Utilization threshold for the alarm to be 15%. Select **Next**

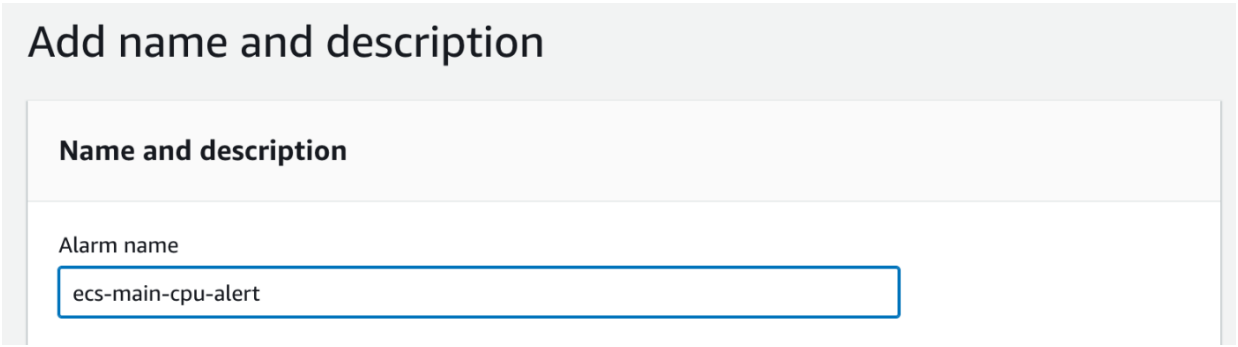


Click **Next**

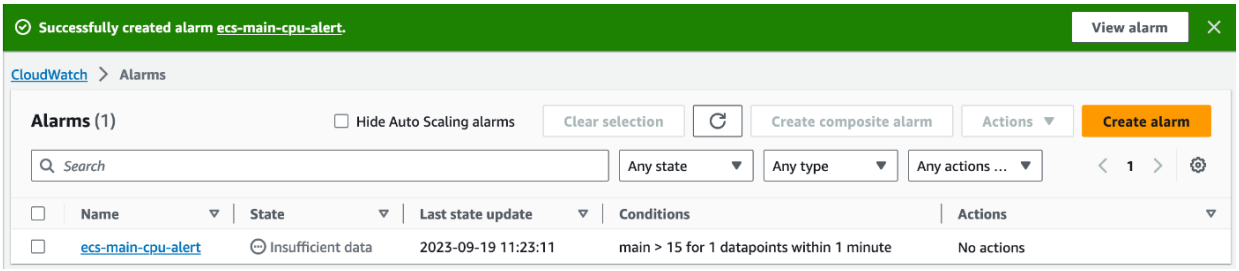
In the **Configure actions** screen, click Remove to remove the default action.



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



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, where you can view the alarm.

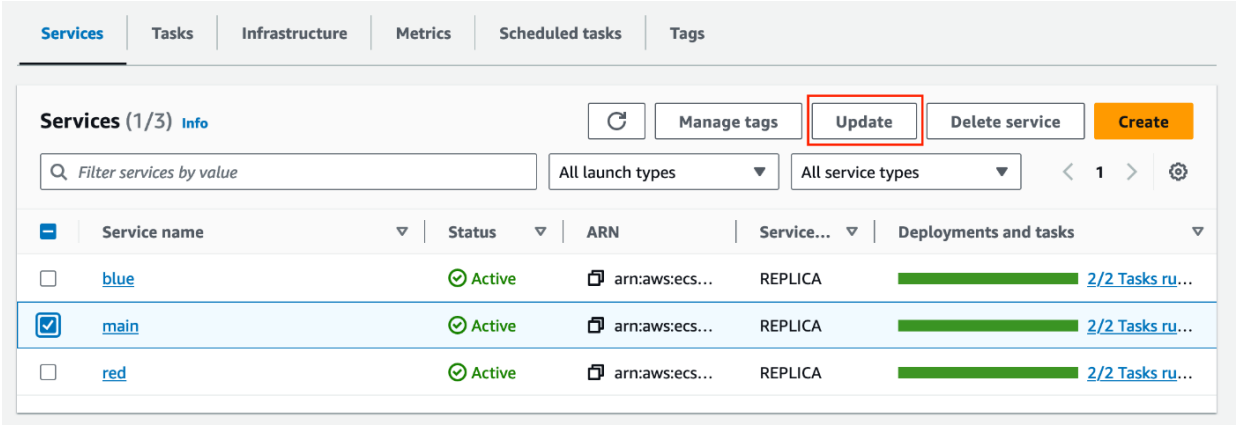


Configure Service Auto Scaling in ECS

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.

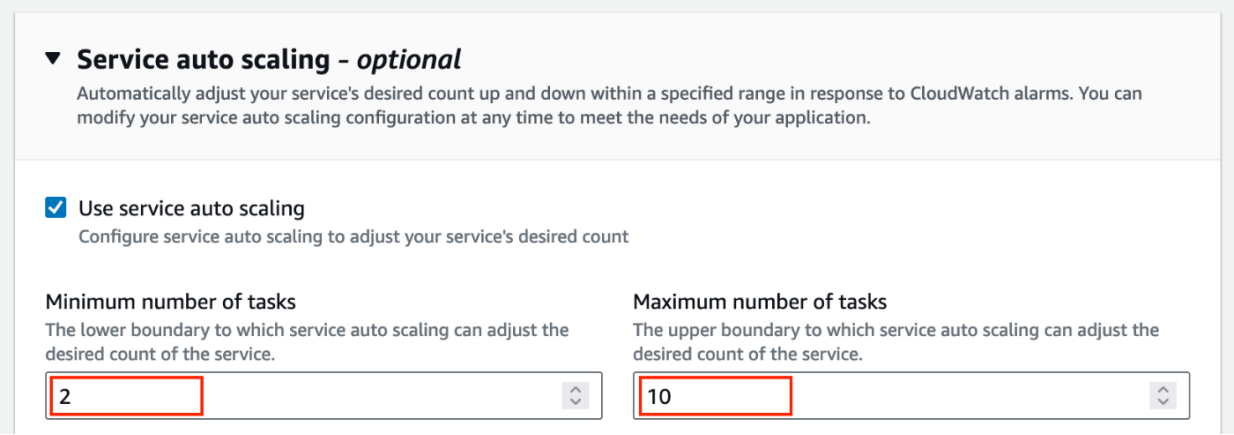
When your CloudWatch alarms trigger an Auto Scaling policy, Application Auto Scaling decides the new desired count based on the configured scaling policy. Then, Application Auto Scaling makes the **UpdateService** API call to Amazon ECS with the new desired count value. The Amazon ECS service scheduler 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**.



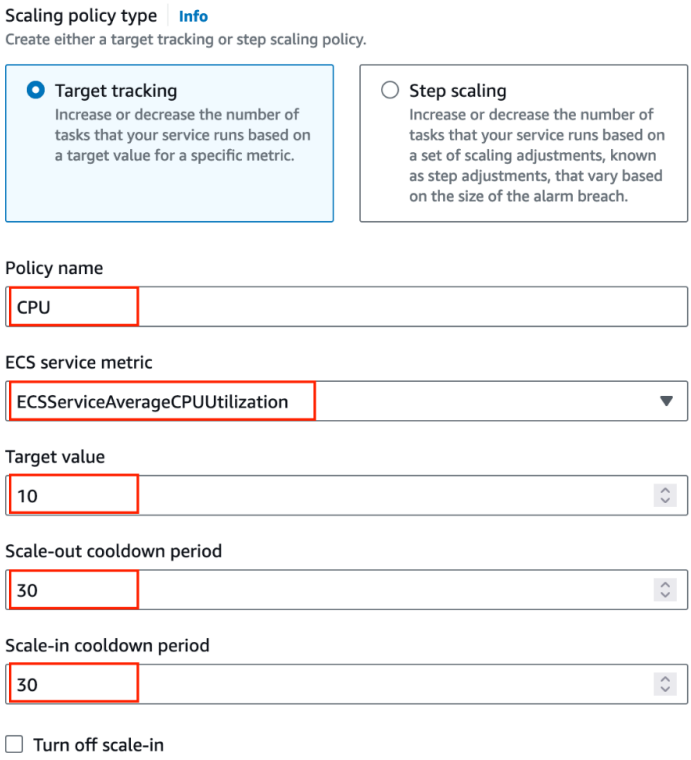
Scroll down to the **Set Auto Scaling (optional)** and click on **Configure Service Auto Scaling to adjust your service's desired count,** as shown below

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



Click on **Add scaling policy**

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



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
```

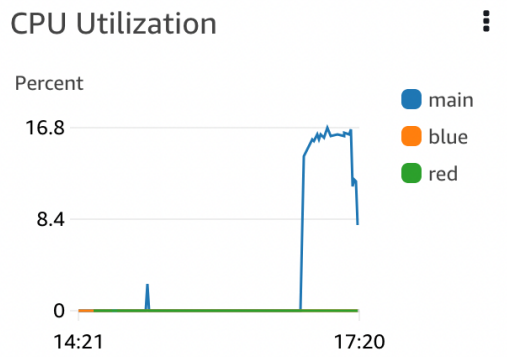
Note: Make sure that '/' is at the end of the ALB DNS name, because the **web** service is bound to the default path (/). Without it, the traffic burst will not hit our web application.

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

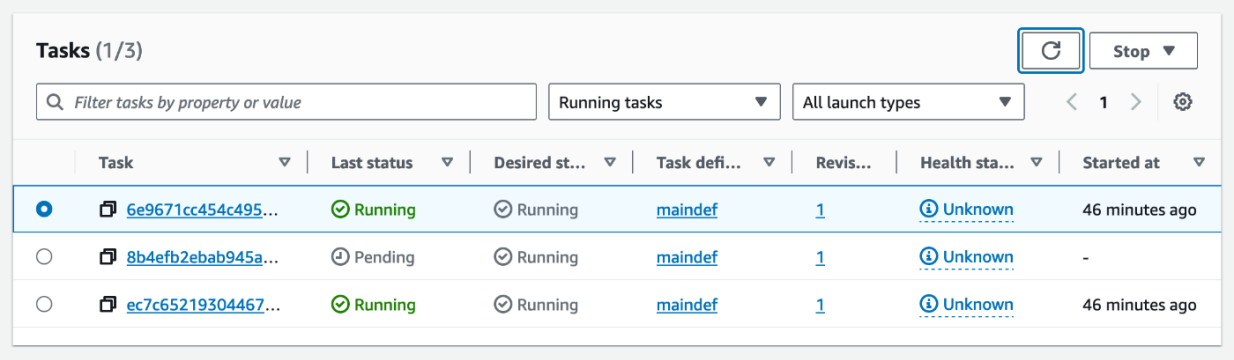
```
\oli:~/environment/ecsworkshop (main) $siege -c 500 -i  http://ecs-workshop-alb-476857044.us-east-1.elb.amazonaws.com/ > /dev/null
** SIEGE 3.0.8
** Preparing 500 concurrent users for battle.
The server is now under siege...
```

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,

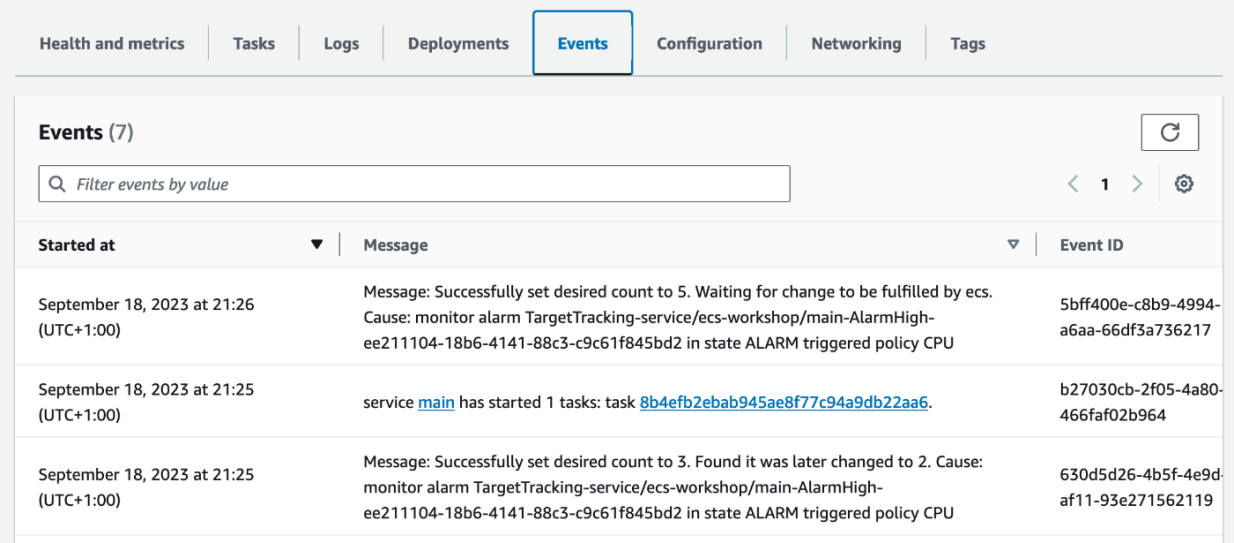


After a while, you will notice additional **maindef** tasks are being started, as shown below one is in **Pending** status.



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.



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