

System Admin

Training Assignments

|  |  |
| --- | --- |
| Program Code |  |
| Issue/Revision | x/y |
| Effective date | 04/Aug /2023 |

Assignment Day 19. Cloud Public AWS – Day 2

Mục Lục:

[Hướng dẫn cài đặt và cấu hình website sử dụng Service AWS ECS 2](#_Toc1535194859)

[1. Architecture: 3](#_Toc1994297641)

[2. Prerequisites 3](#_Toc839409568)

[3. Create a New ECS Service 4](#_Toc418288694)

[4. Create a Task Definition 8](#_Toc1475807546)

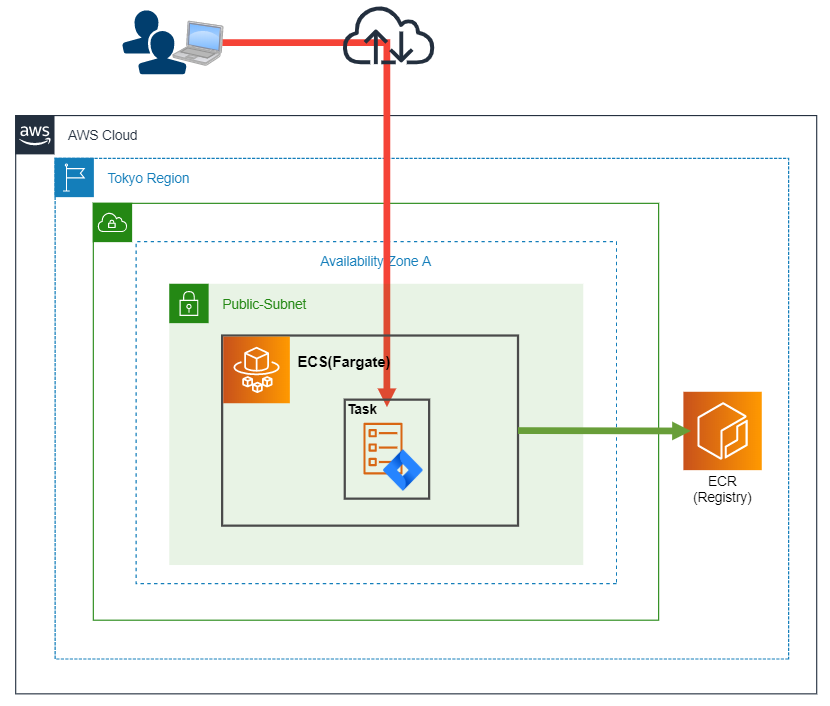
[5. Deploy the Application 11](#_Toc877980400)

[6. Create the service 15](#_Toc357158275)

[7. Access website 16](#_Toc1151915239)

# Hướng dẫn cài đặt và cấu hình website sử dụng Service AWS ECS

## Architecture:



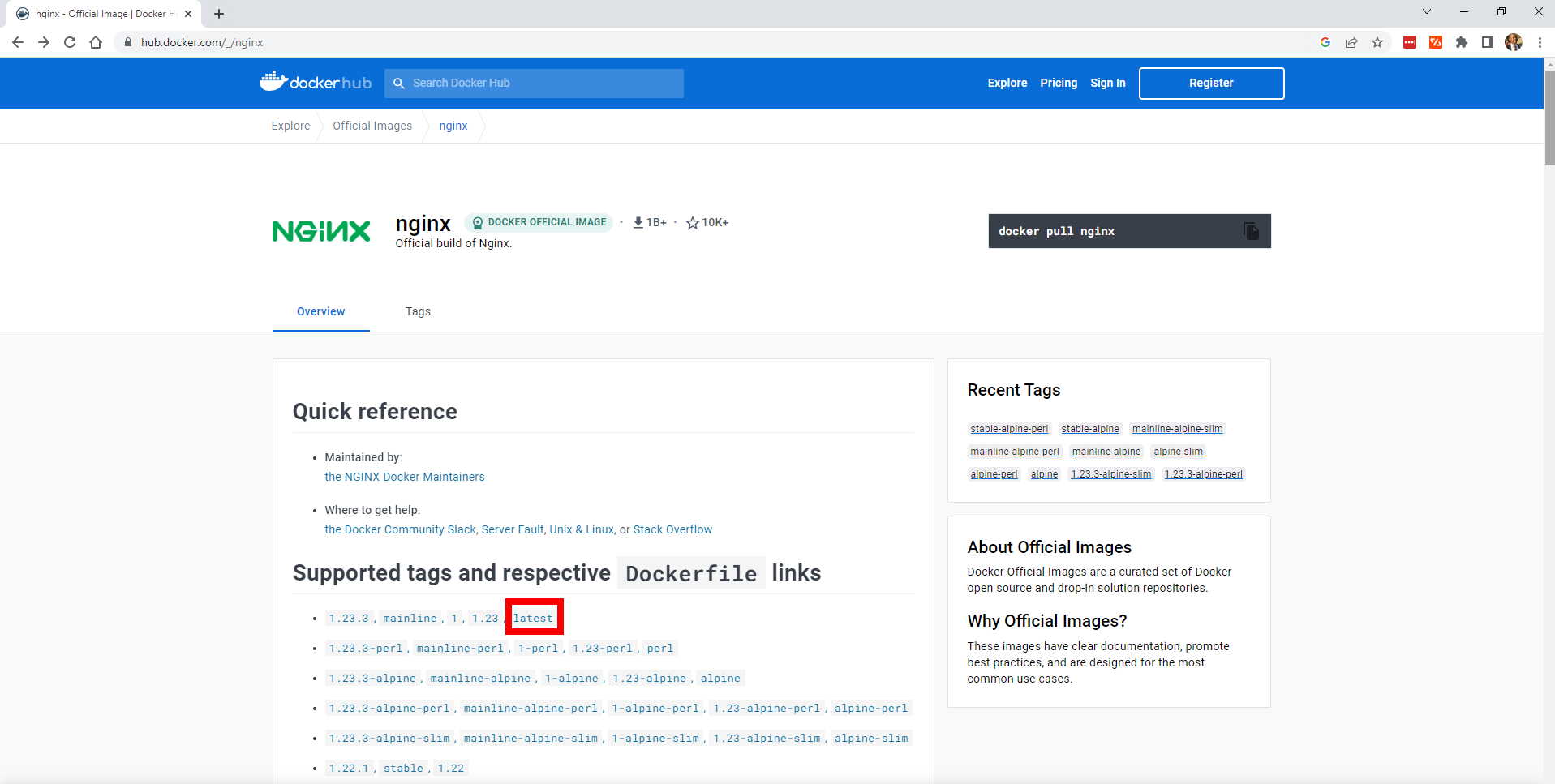
## Prerequisites

Before you begin, you’ll need the following:

* An [Amazon Web Services (AWS) account](https://aws.amazon.com/console/)
* A Docker image (*ie* a readily available Docker Hub image or a Dockerized app of your choosing).

The walkthrough in this article will use a NGINX image which is hosted on Docker Hub. NGINX is a popular open-source web server that is known for its high performance, scalability, and reliability. It’s commonly used to serve static and dynamic content on the web, and can also be used as a reverse proxy, load balancer, and caching server.

To retrieve the latest NGINX image on Docker Hub, use the **nginx:latest** command in the ECS console:

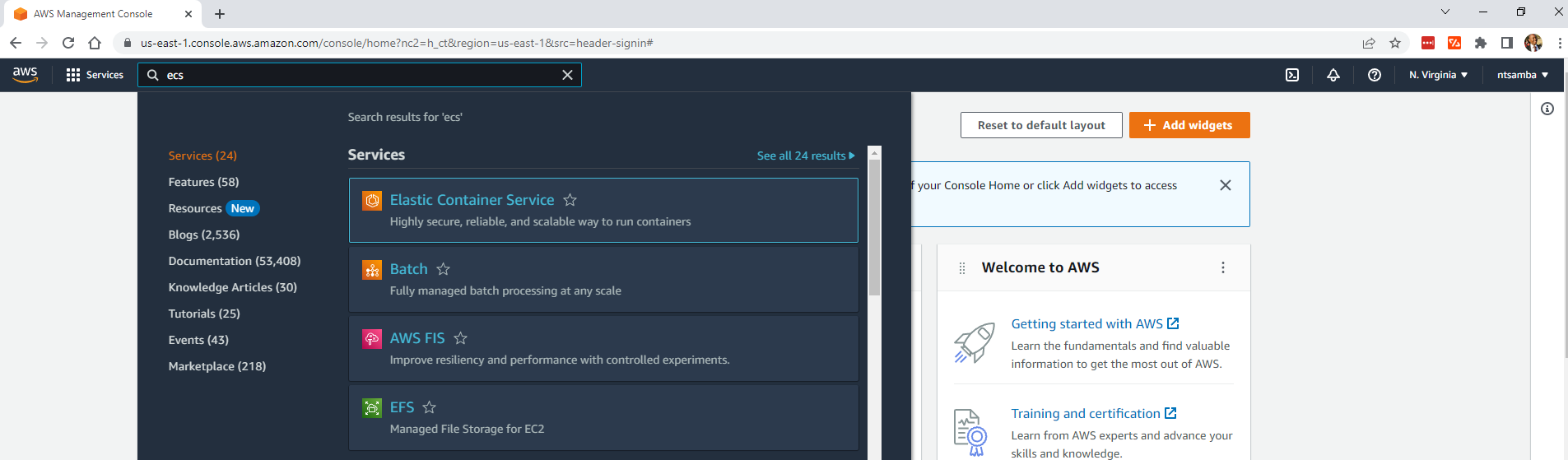


## Create a New ECS Service

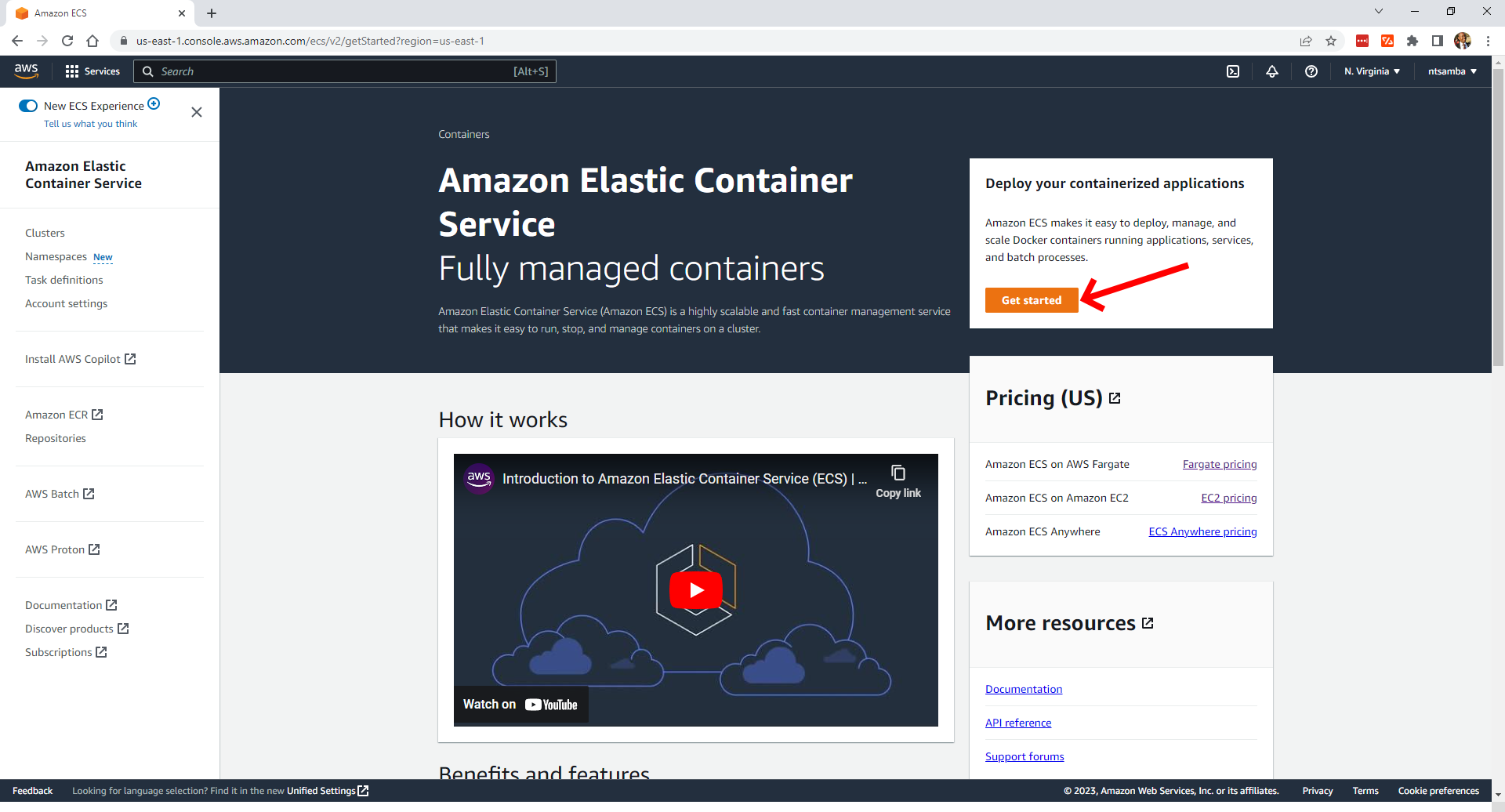
An ECS Service is a way to define and run a group of related Docker containers on an Amazon Elastic Compute Cloud (EC2) or Fargate cluster.

To create an ECS Service, you need to create a cluster, specify a task definition, configure load balancing, and then deploy the service. You’ll walk through each of these steps in the following sections.

Let’s start by creating a new service. Log into the AWS Management Console and search for “ECS”:

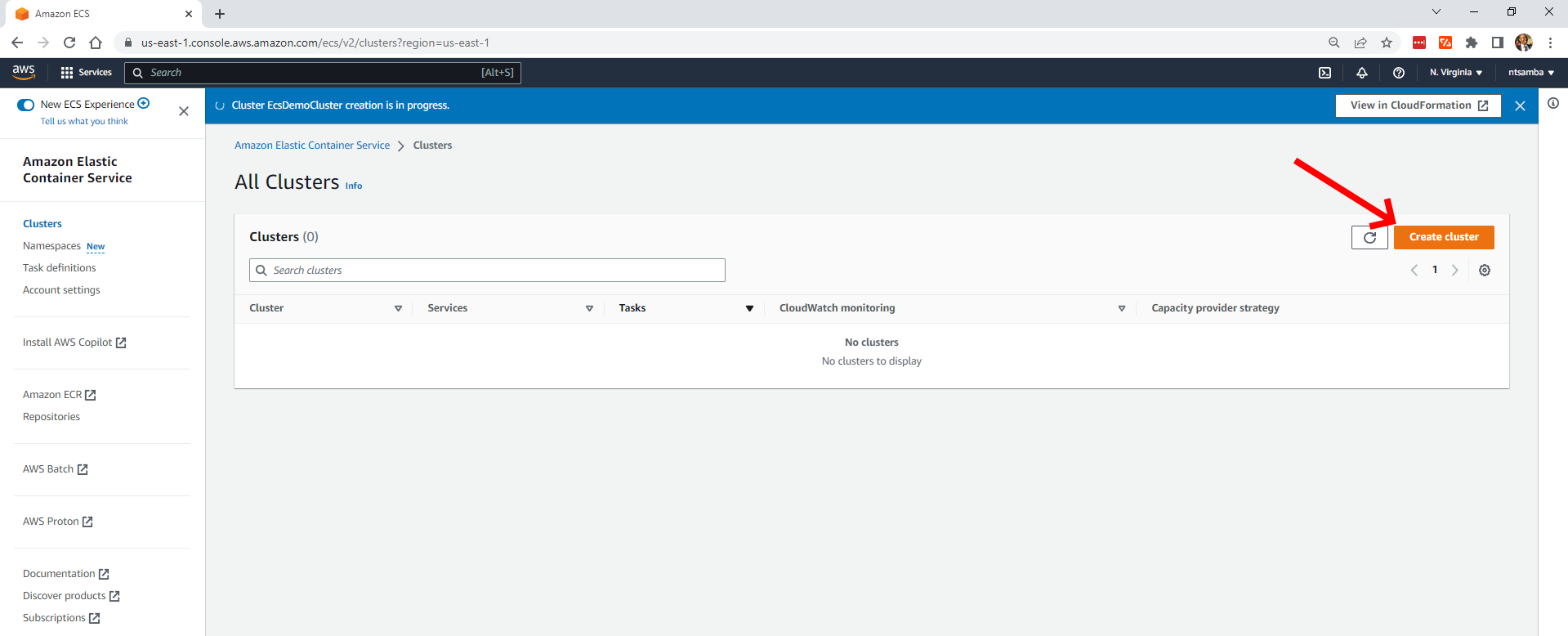


Select **Elastic Container Service** and then **Get started** to deploy your containerized application:

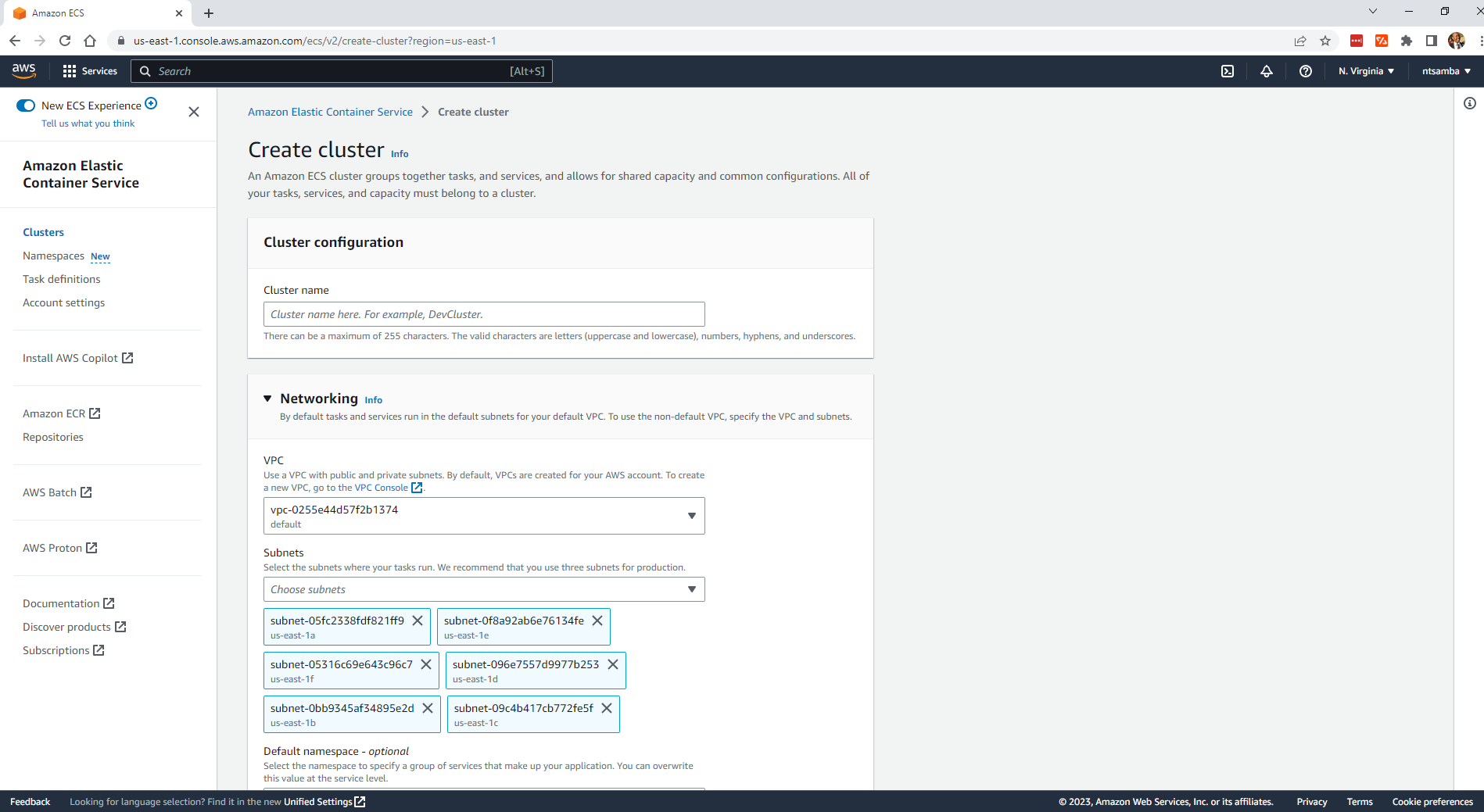


Select **Get started**

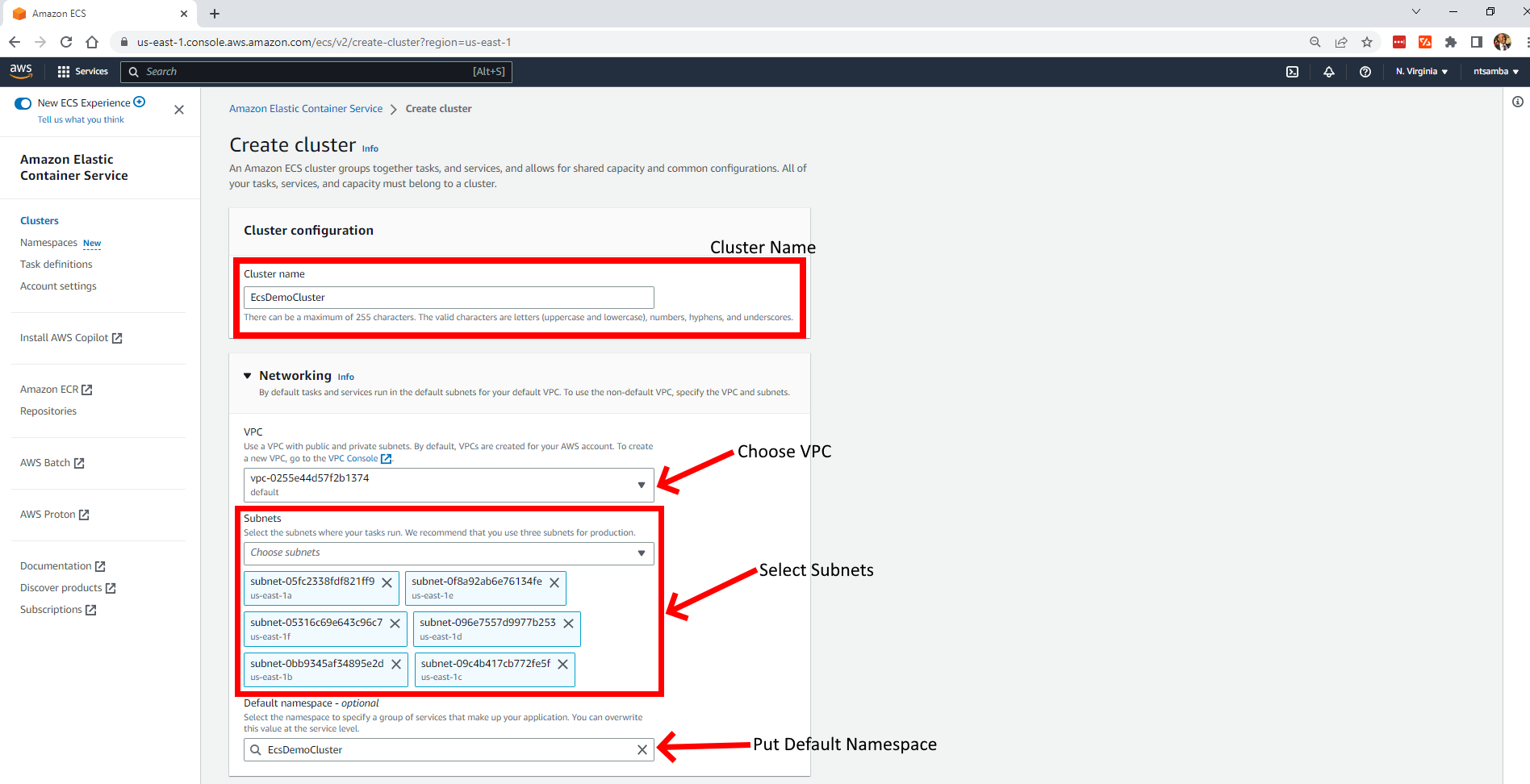
This will open the **Clusters** page, where you can create your cluster:



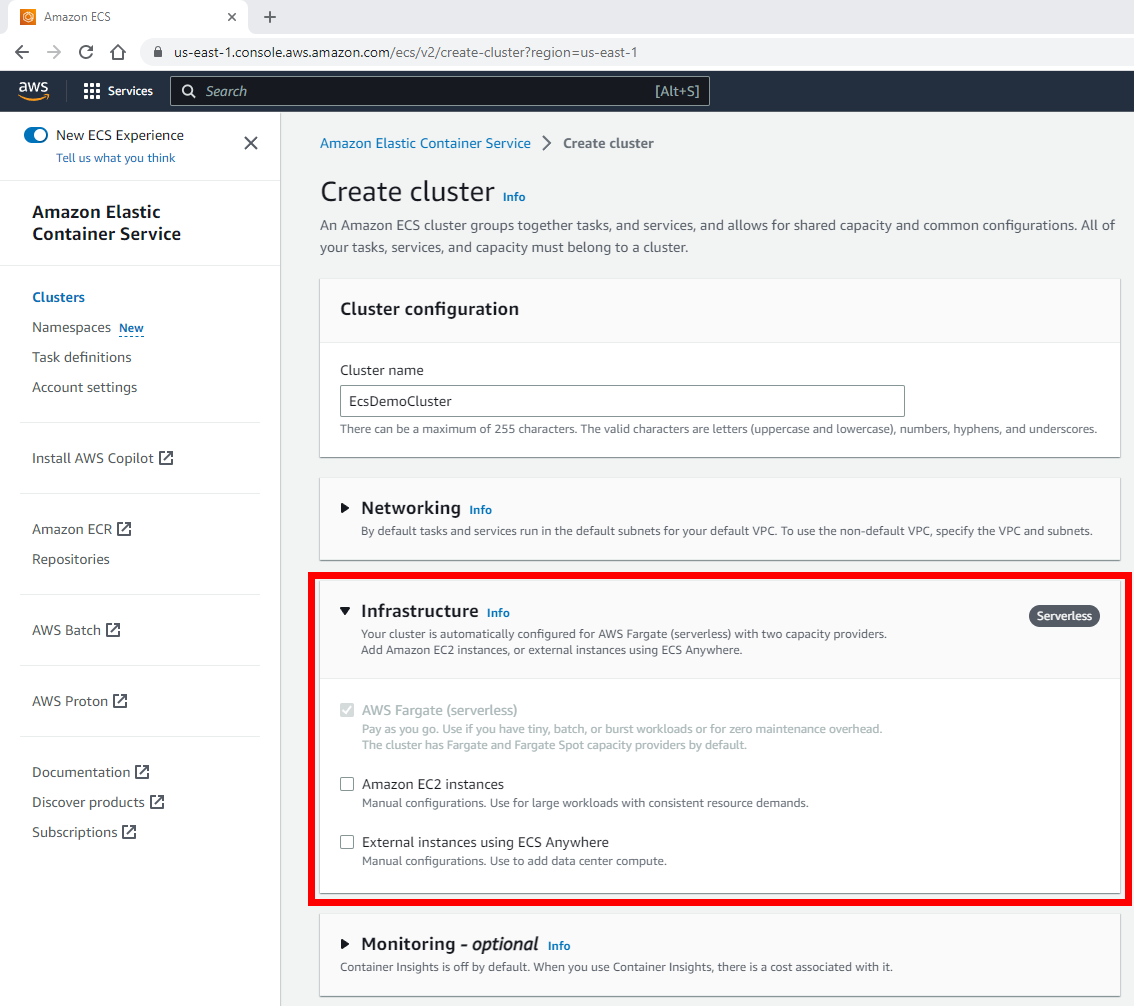
A cluster is simply a logical grouping of services or stand-alone tasks. To create the cluster, click on **Create cluster**, which will take you to the **Cluster configuration** page:



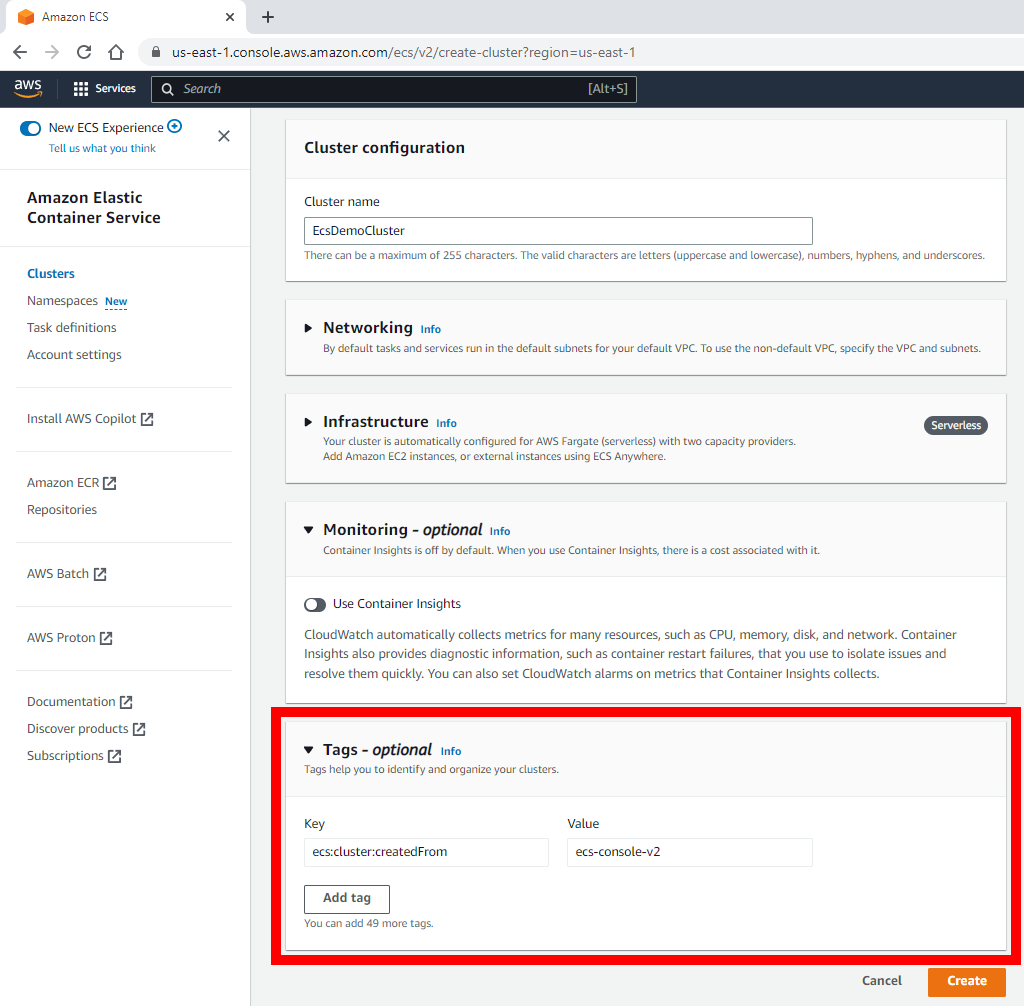
Give the cluster a name (here, it’s “EcsDemoCluster”), and under the **Networking** section, select the default Virtual Private Cloud (VPC).



Now, expand the **Infrastructure** section, which can be found after the **Networking** section. The infrastructure is the actual hardware that will be used to deploy the containers:

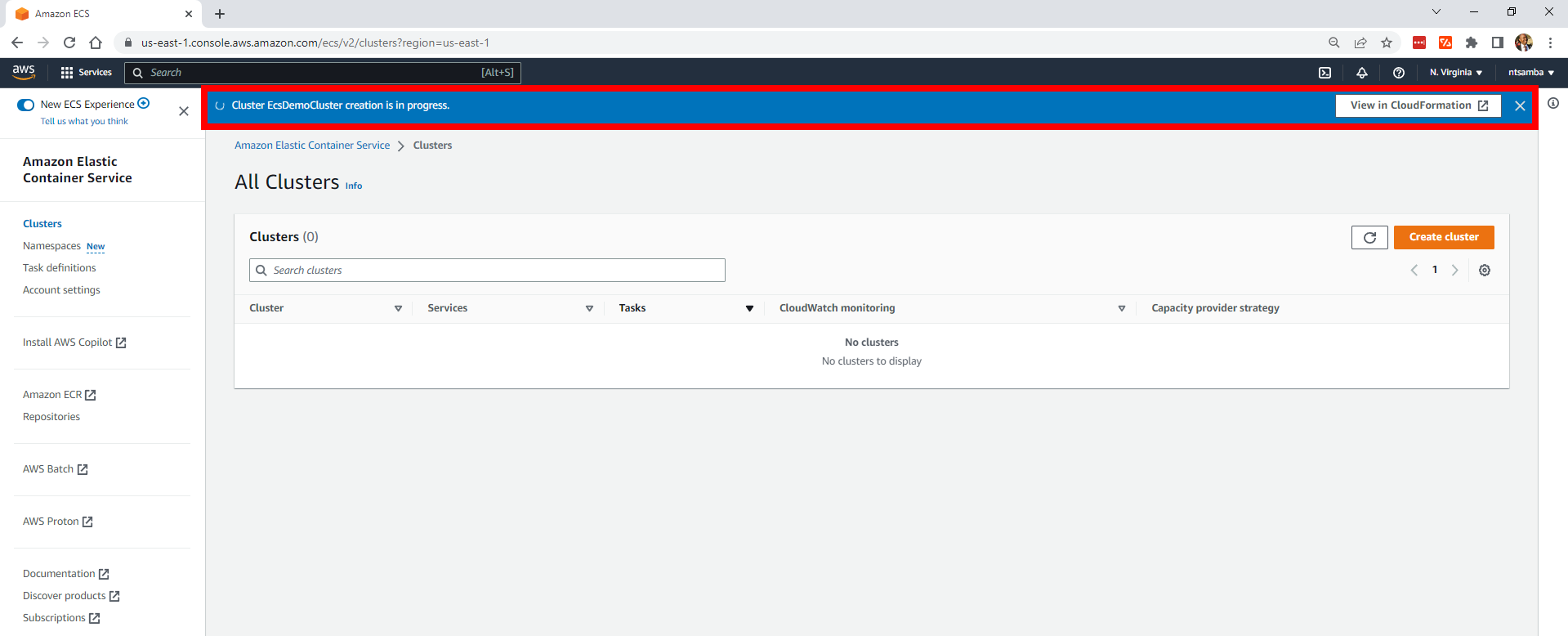


The cluster is automatically configured for AWS Fargate, which is serverless with two capacity providers. You can enable Amazon EC2 instances and external instances using [Amazon ECS Anywhere](https://aws.amazon.com/ecs/anywhere/), but both require manual configurations, which is outside the scope of this article. Leaving the default option is sufficient for this tutorial.

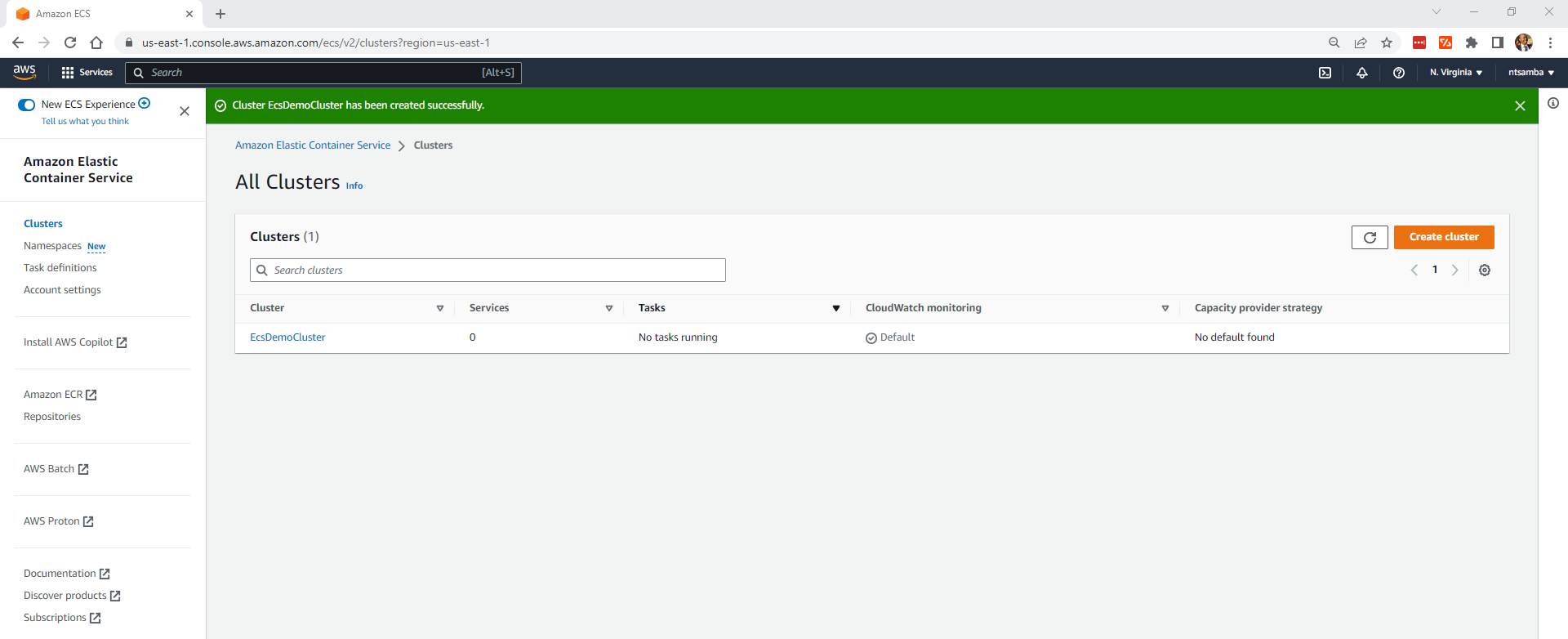


To create a tag, expand the **Tags** section. As you can see in the previous image, you only have one tag that identifies how the cluster was created with the key **ecs:cluster:createdFrom** and value **ecs-console-v2**.

To create the cluster, click **Create cluster** on the right. This will take about twenty minutes, and you should see a notification at the top of the screen letting you know the cluster creation is in progress:

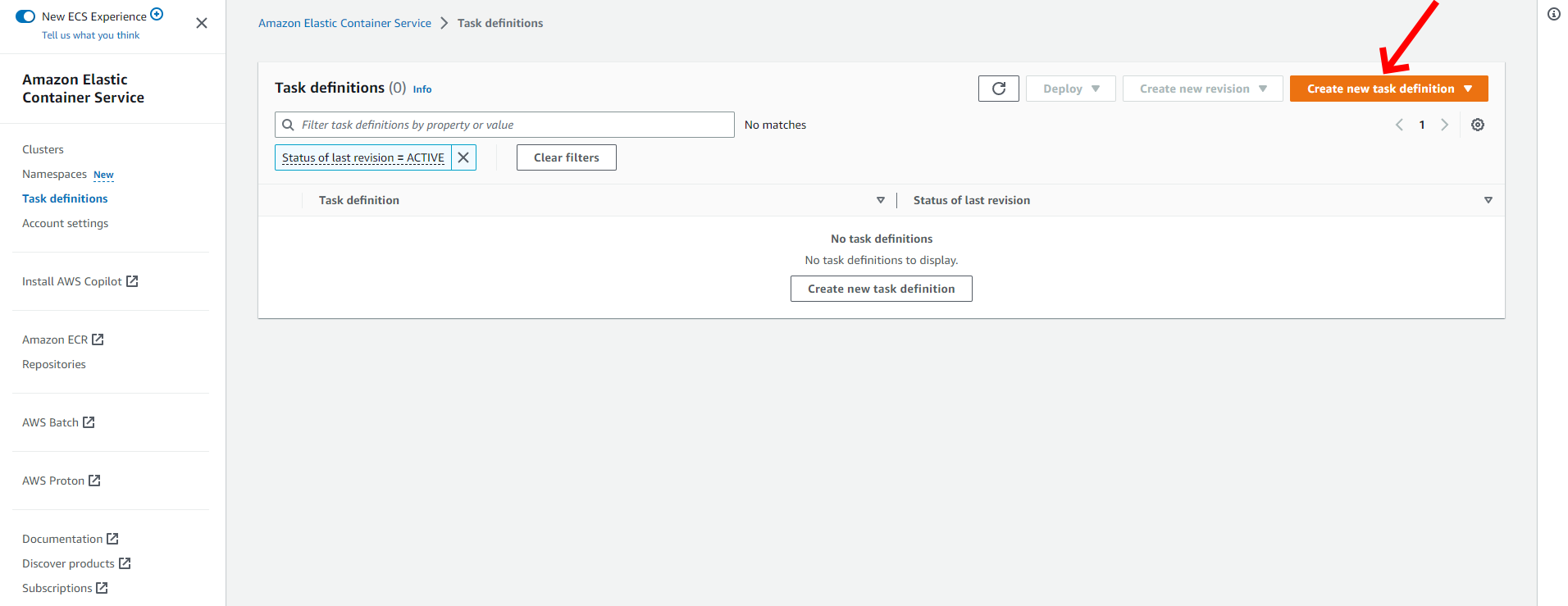


Once the cluster has been created successfully, you’ll receive a confirmation message, and the cluster will now be listed in your **All Clusters** list:

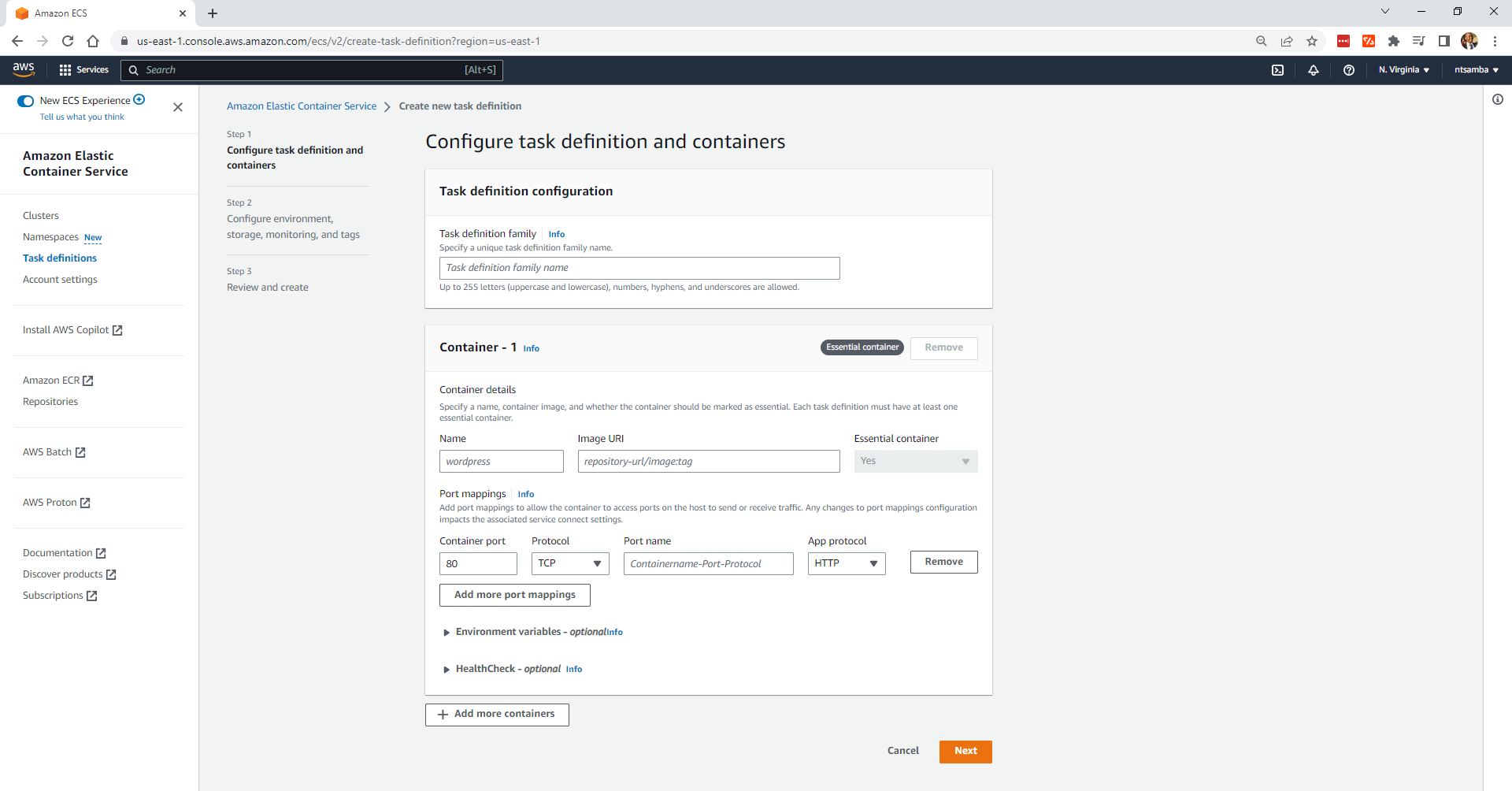


## Create a Task Definition

The next step is to create a task definition, which is essentially a blueprint of how your container should launch. It contains details such as how much CPU and memory your container should have, what image it should use, and what ports need to be opened. To create a task, click **Task definitions** from the menu on the left. This will open the **Task definition configuration** page:

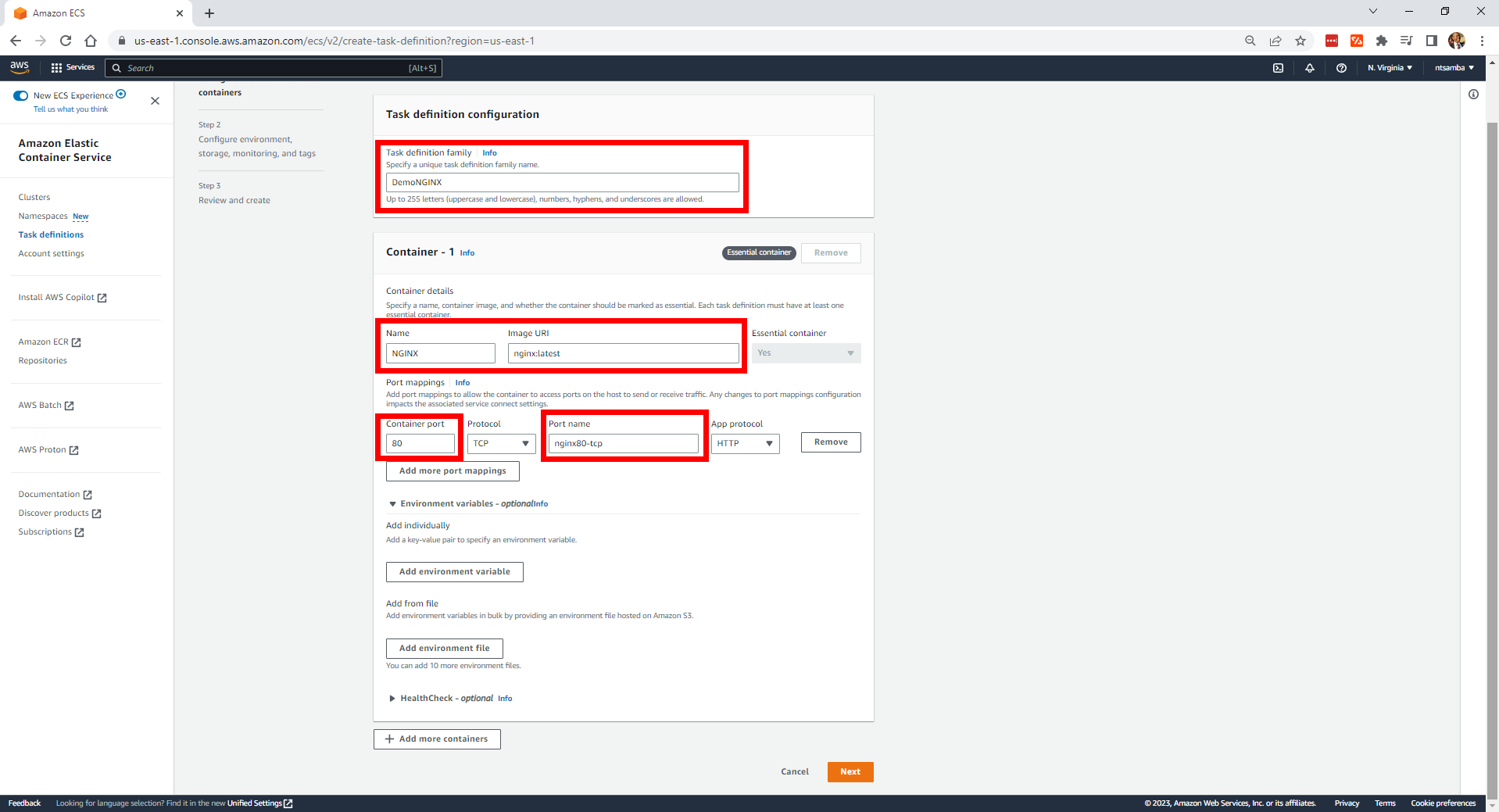


Then click on **Create new task definition** to configure the task definition:



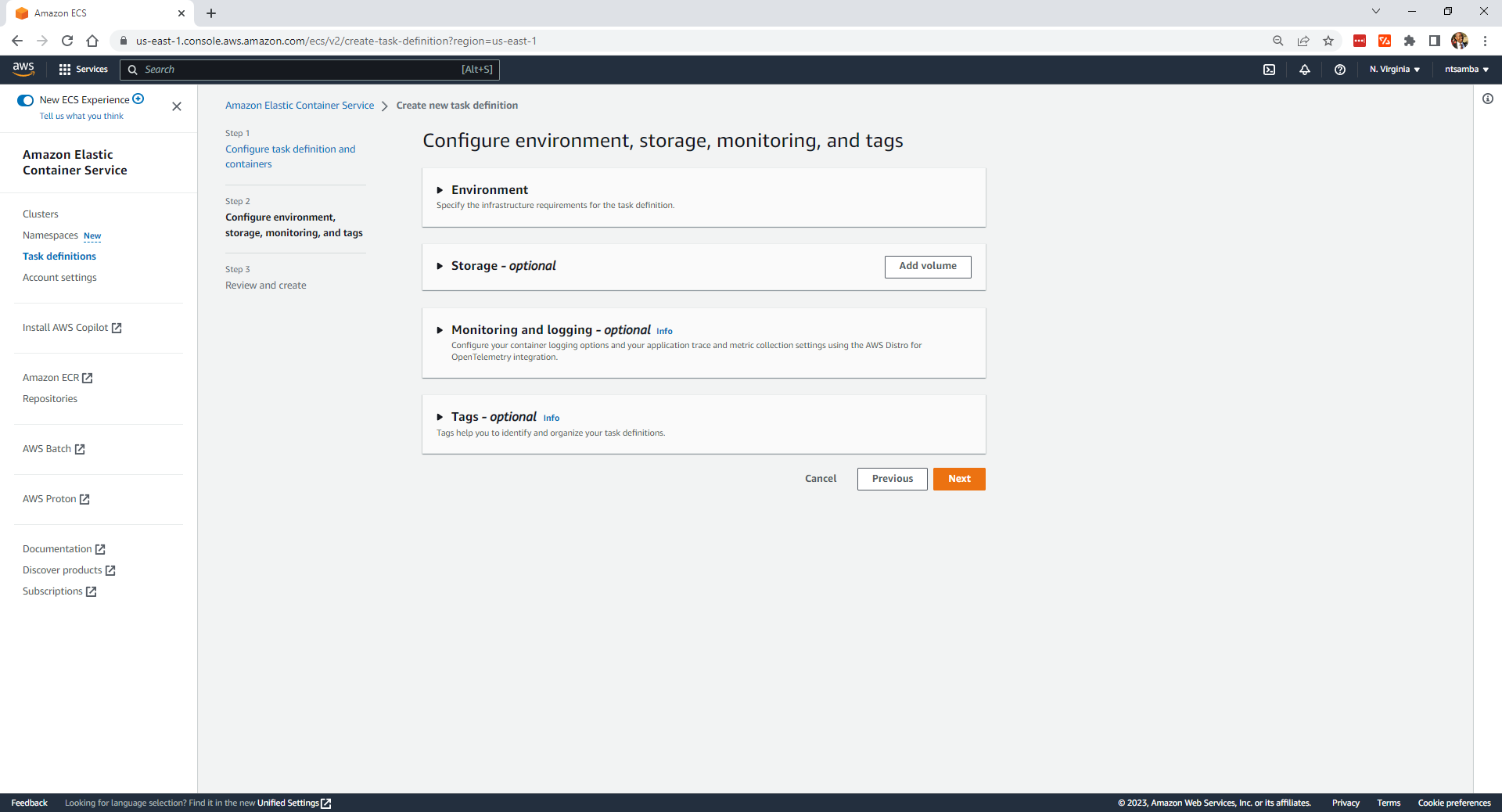
Fill in the **Task definition family name** (here, “DemoNGINX” was used), and then fill in the container details. “NGINX” was used as the name of the container, and “nginx:latest” as the image URI.

Input “80” for the container port; then give the port a name (*ie* “nginx80-tcp”) and select **Next**:

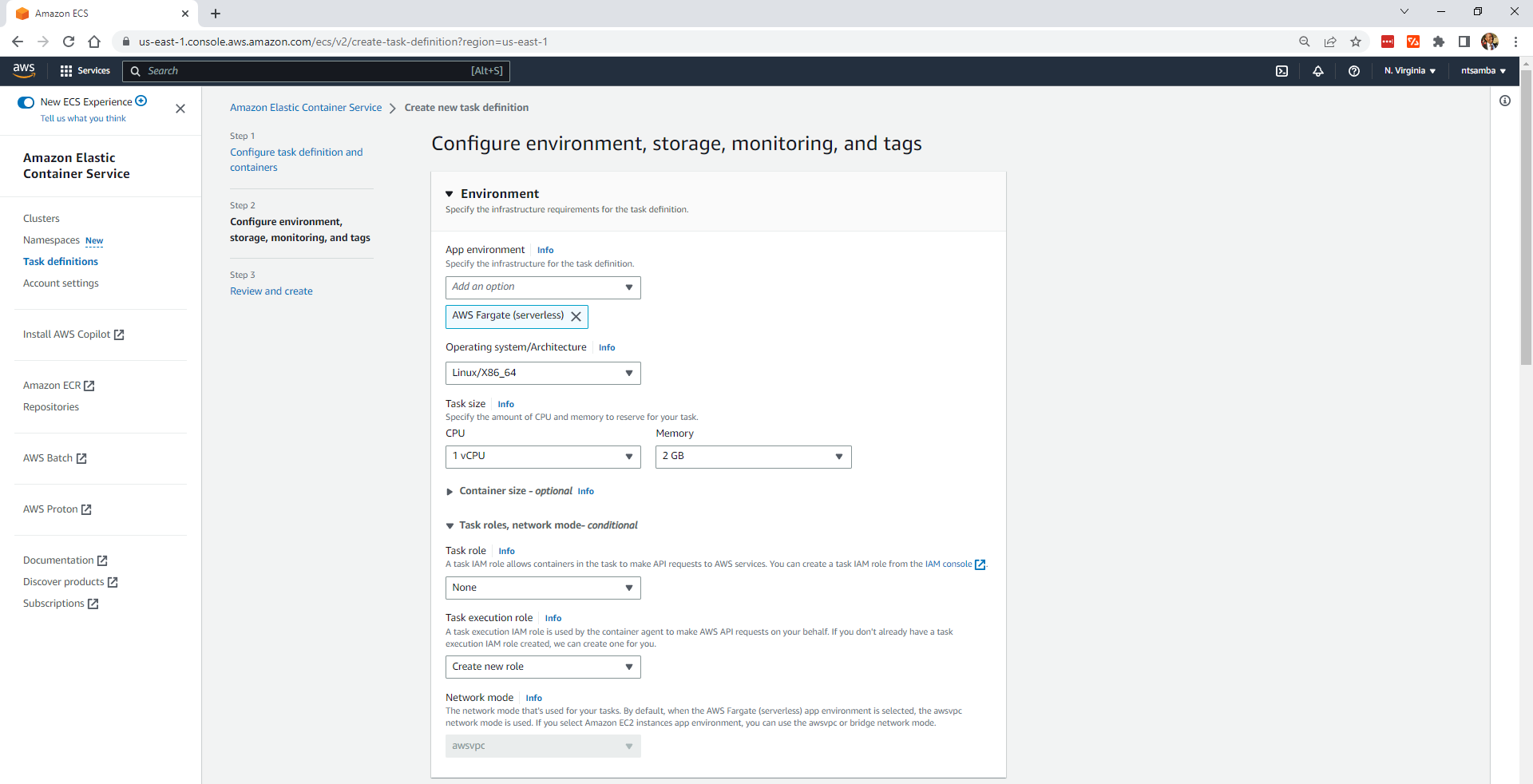


This will take you to a page where you can configure the environment, storage, monitoring, and tags. The environment allows you to specify the infrastructure requirements for the task definition and storage allows you to configure the storage options for your container including setting up data volumes and Elastic File System (EFS) mounts. These storage options can help you manage persistent data and share files across multiple containers.

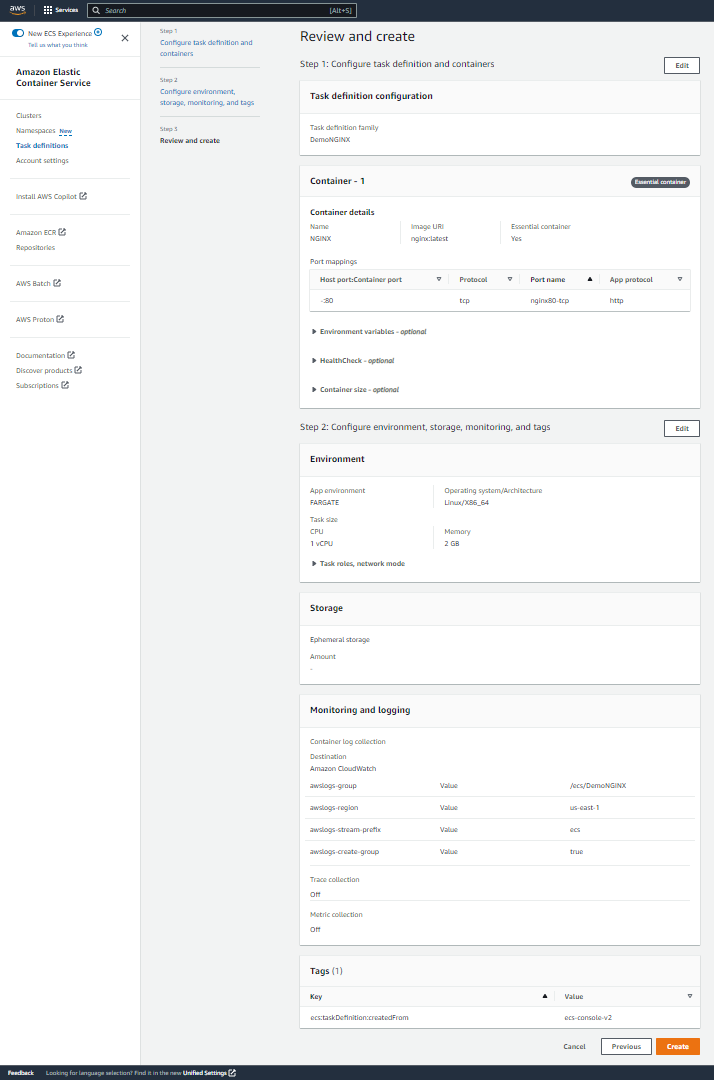
Monitoring allows you to specify the logging options such as the log group to use and tags help you to organize and categorize your task definitions. This makes it easy to manage and transfer resources across your container environment:



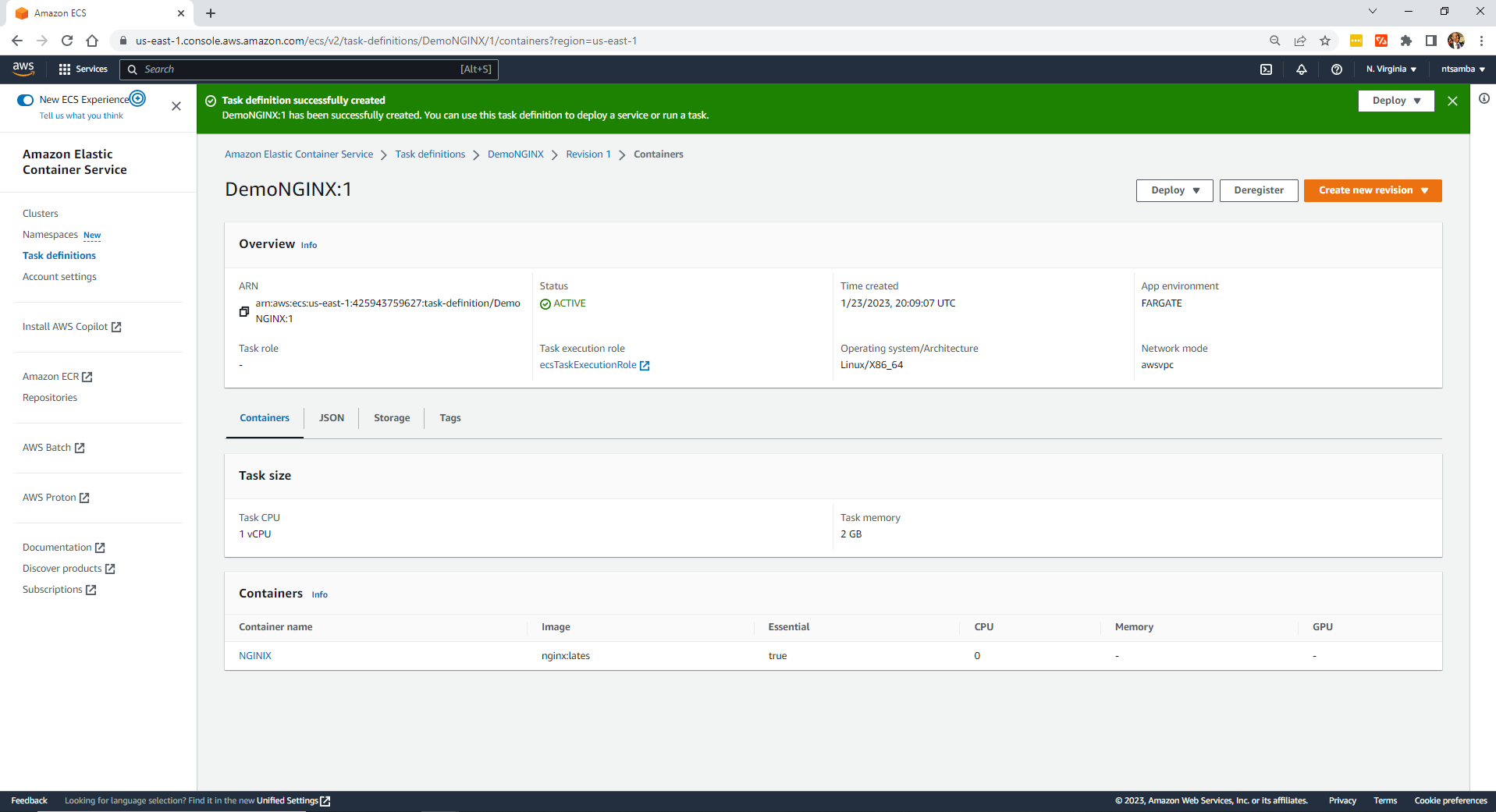
To save money, expand the **Environment** section and change the memory from 3 GB to 2 GB. If you have a task Identity and Access Management (IAM) role, you can select it under **Task role**; otherwise, select **None**. Finally, under the **Task execution role**, select **Create new role**:



Leave everything else as is and select **Next**. This will take you to the **Review and create** page, where you can review your settings:

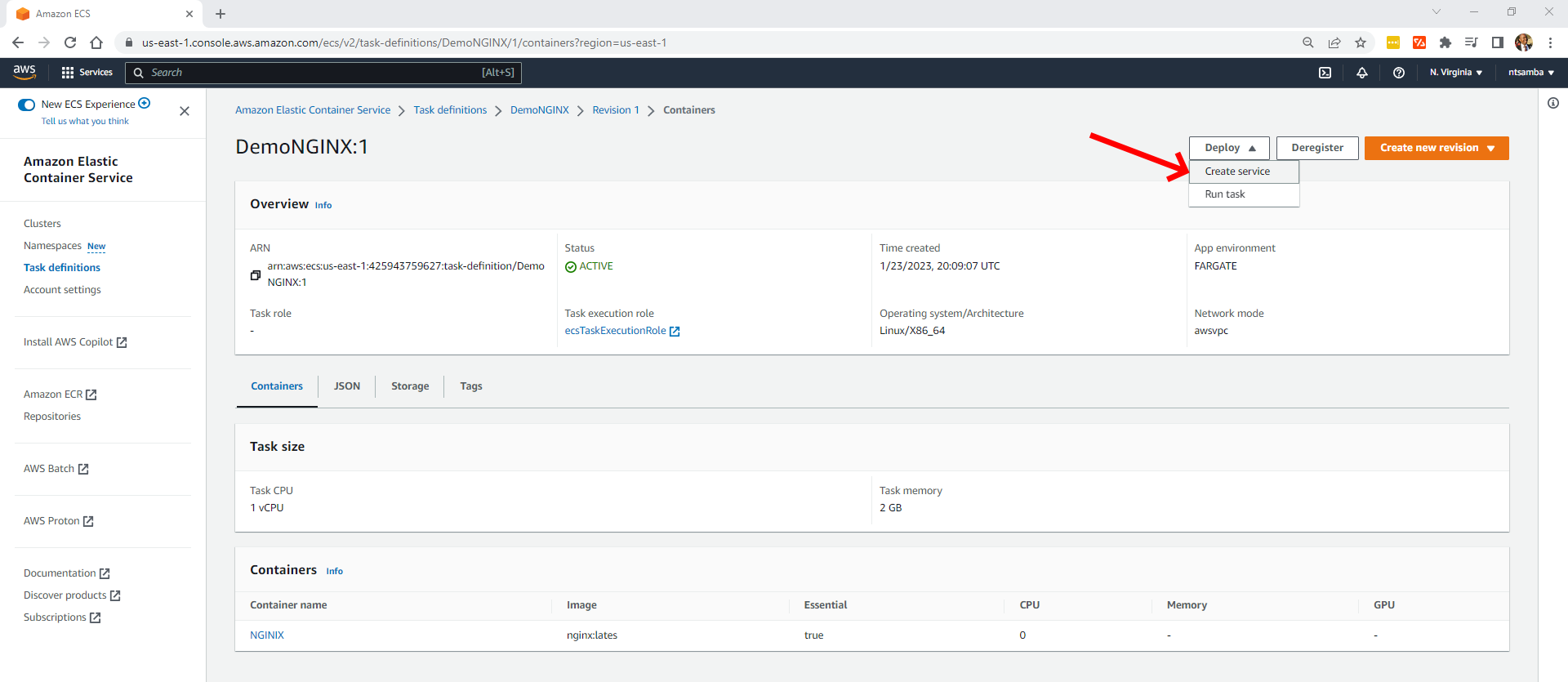


If everything looks good, click on **Create** at the bottom to create the task definition. You would get a confirmation message if the DemoNGINX task definition was created successfully:

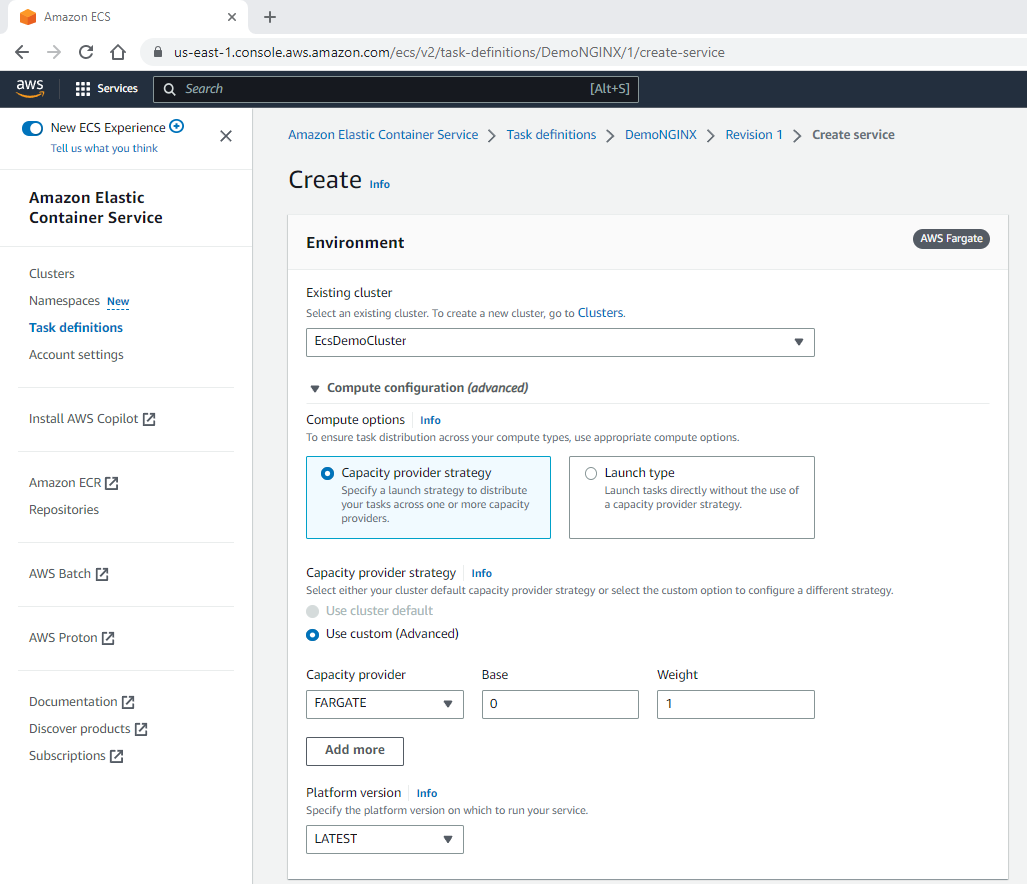


## Deploy the Application

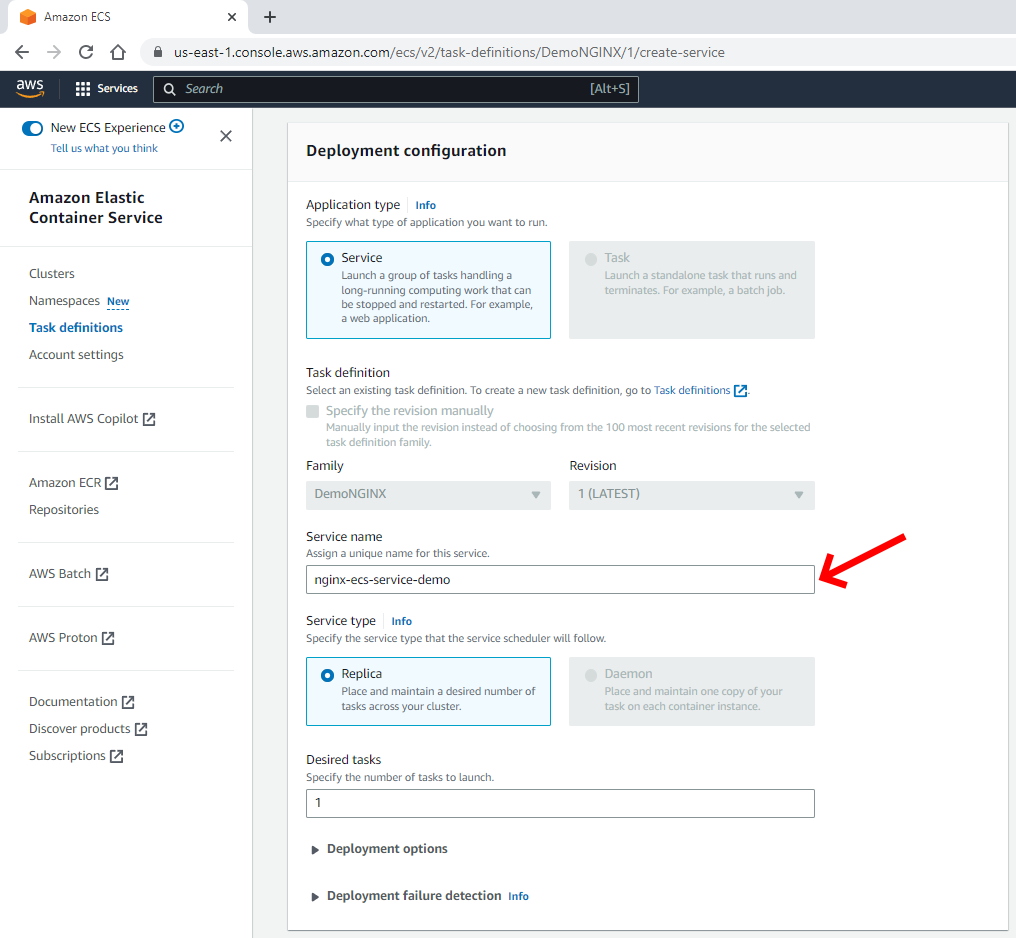
Now you need to create a service to run the application. Click on the **Deploy** drop-down in the upper right-hand corner and select **Create service**:



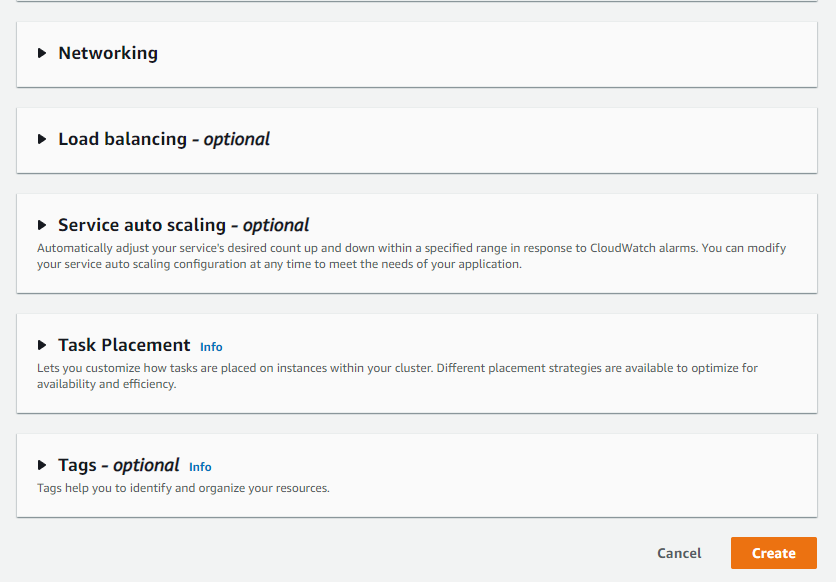
This will open up the **Create service** page. Under **Environment**, select the cluster that you created previously (*ie* “EcsDemoCluster”) and keep everything else as it is:

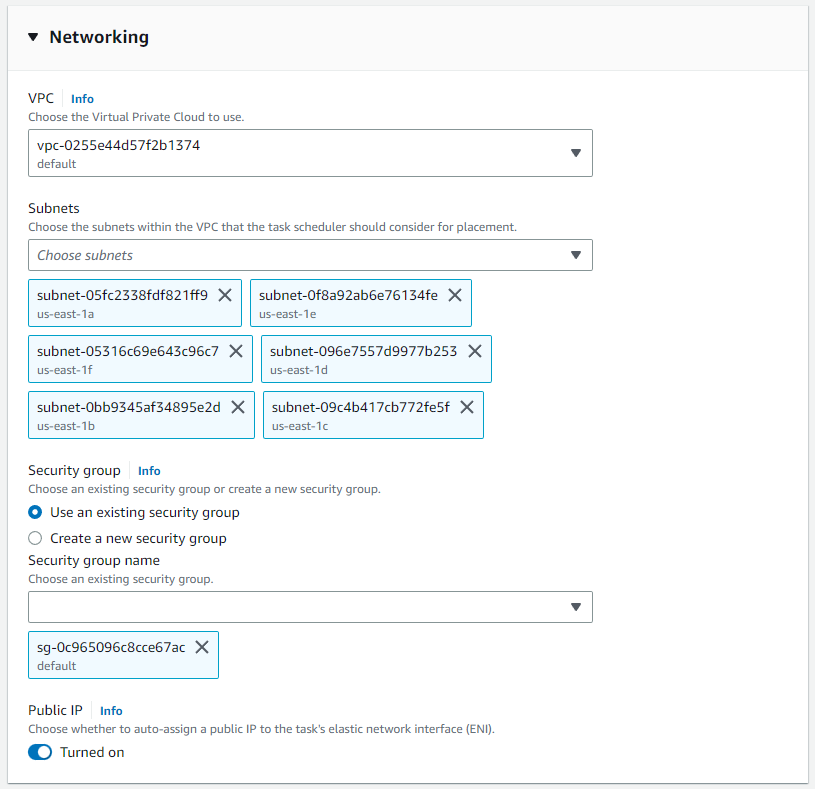


Scroll down to **Deployment Configuration** and give your service a name. Here, “nginx-ecs-service-demo” was used:



Scroll down and expand the **Networking** section and enable the **Public IP**:

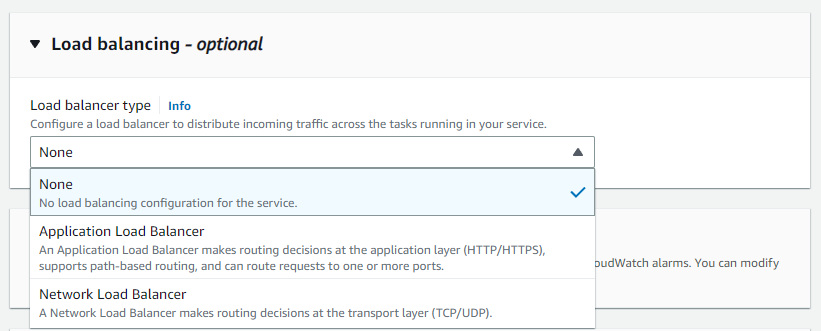
**Networking** section



Enable Public IP for service

Enabling the Public IP allows the container to communicate directly with the internet without any intermediate services or gateways:

Scroll down and expand the **Load balancing** section:

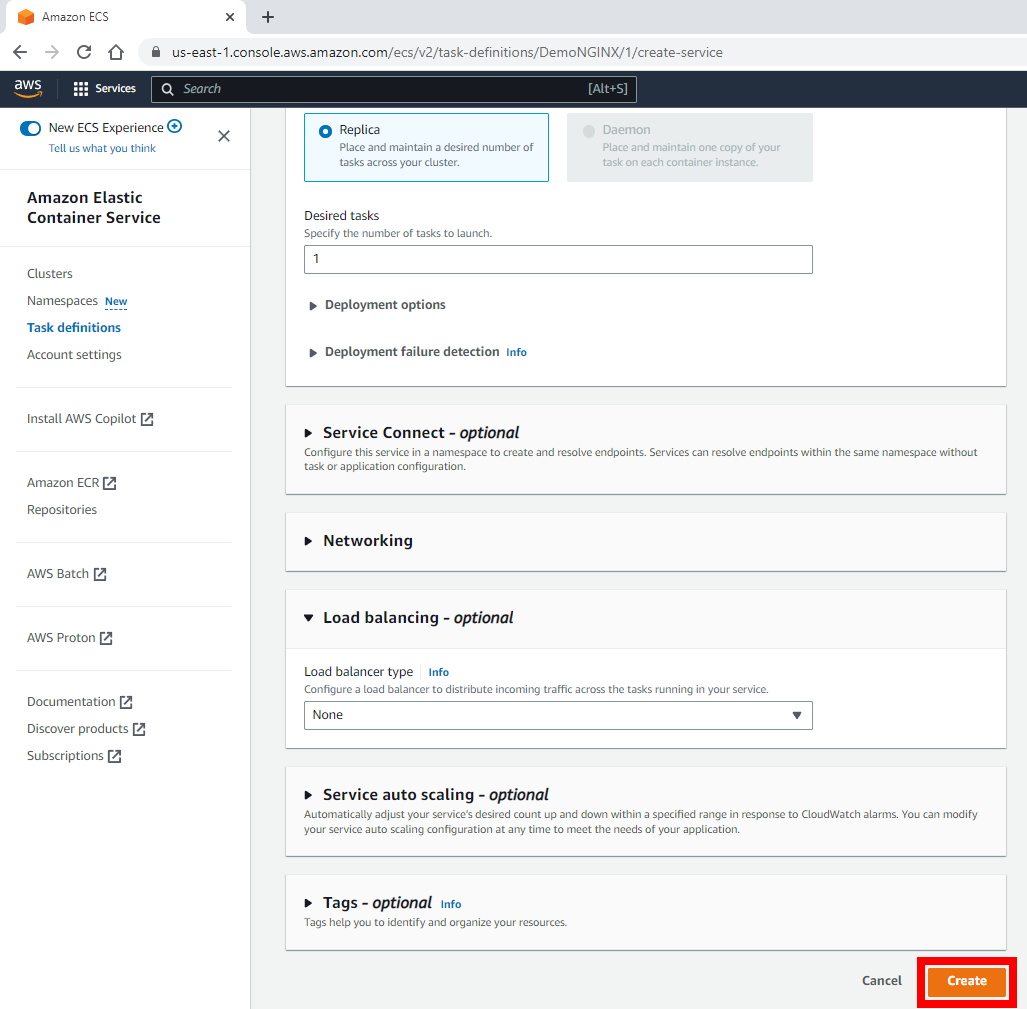


**Load balancing** section

This is where you can configure a load balancer to distribute incoming traffic across the tasks running in your service.

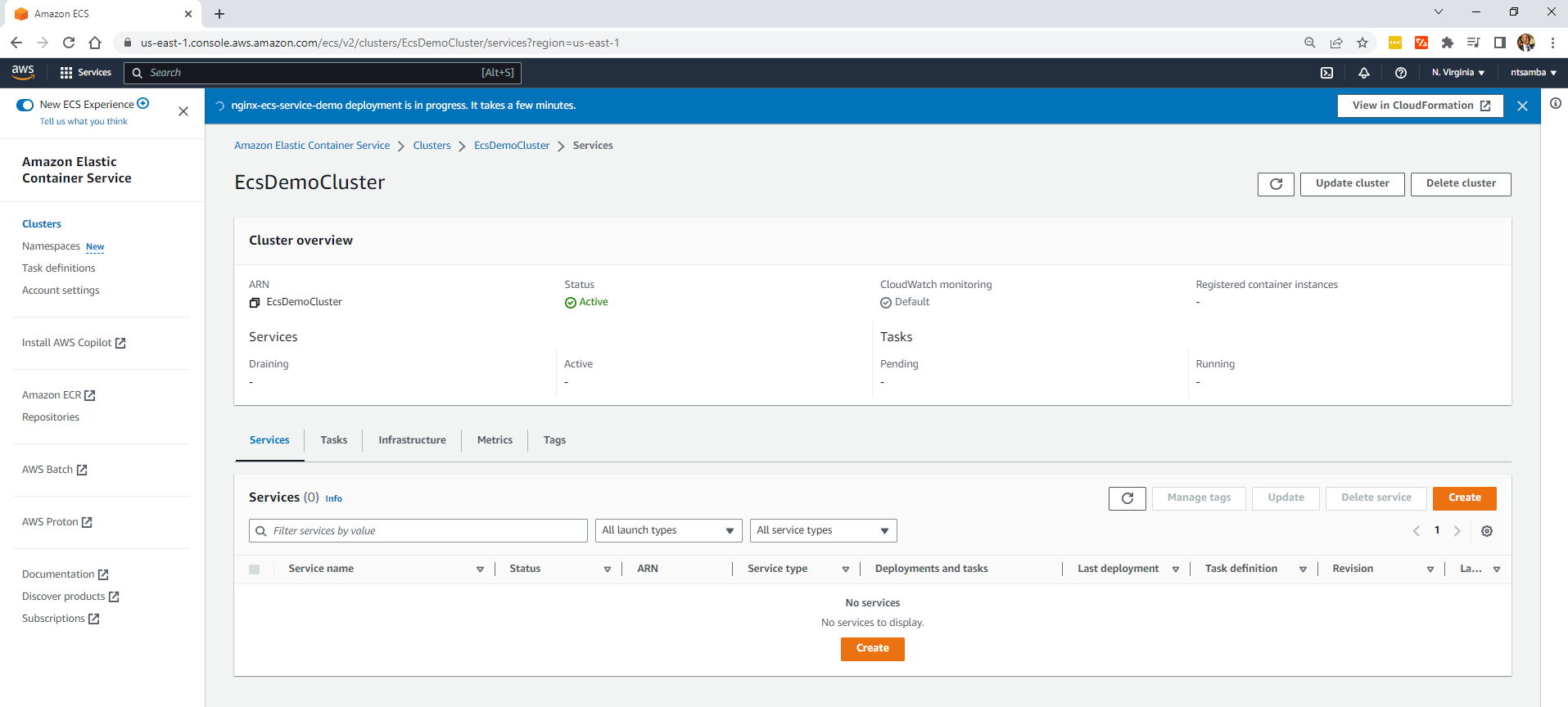
Click on the drop-down, and you’ll see two types of load balancers you can set up. The first option is an **Application Load Balancer**, which makes routing decisions based on the application layer (*ie* you can route traffic based on whether it is HTTP or HTTPS). The second option is a **Network Load Balancer**, which makes routing decisions on the transport layer (TCP or UDP). This demo application is small and, therefore, does not need a load balancer.

To create the service, click on **Create** at the bottom of your screen:



## Create the service

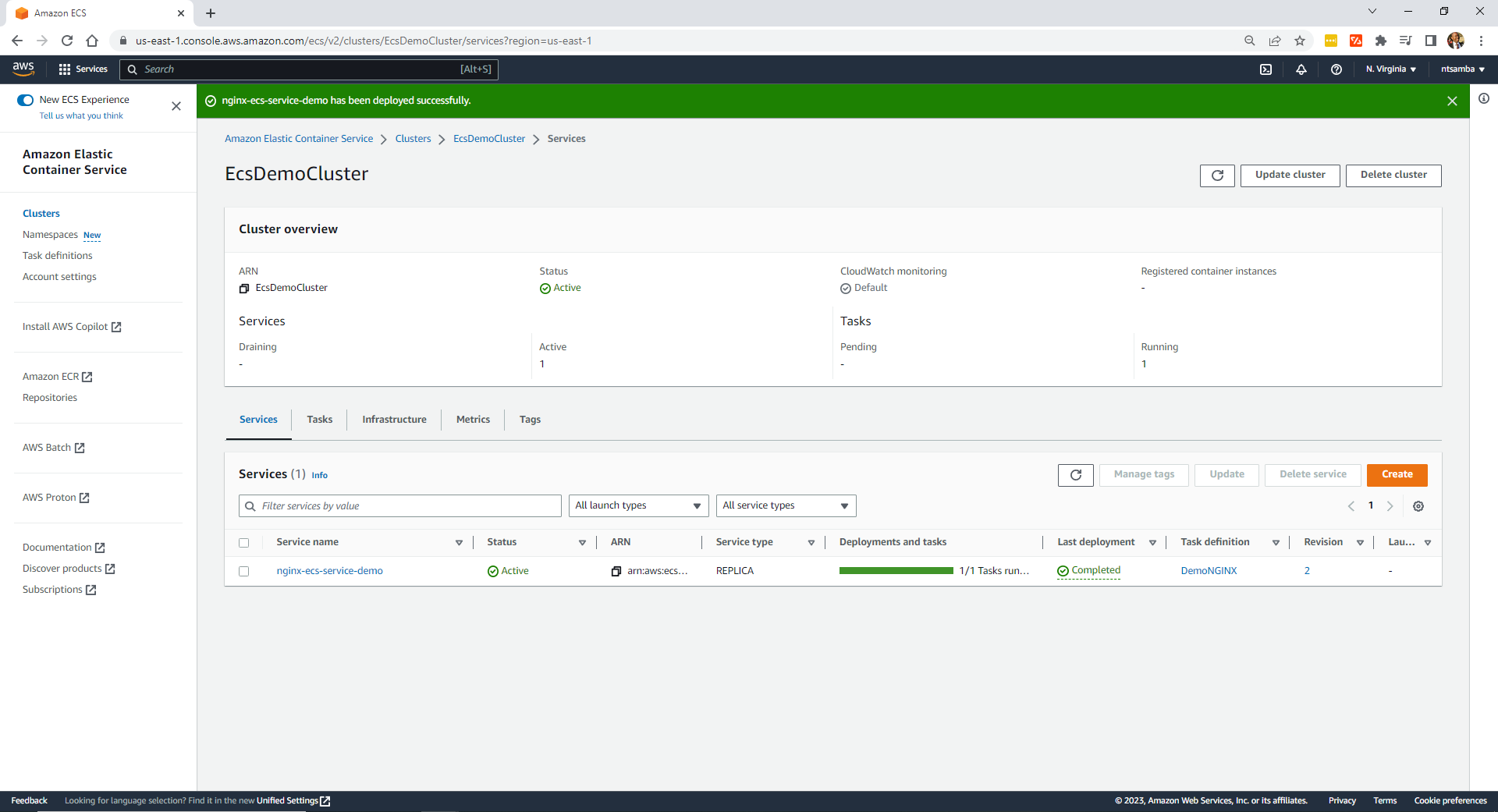
Like before, you should see that the service creation is in progress:



Service creation in progress

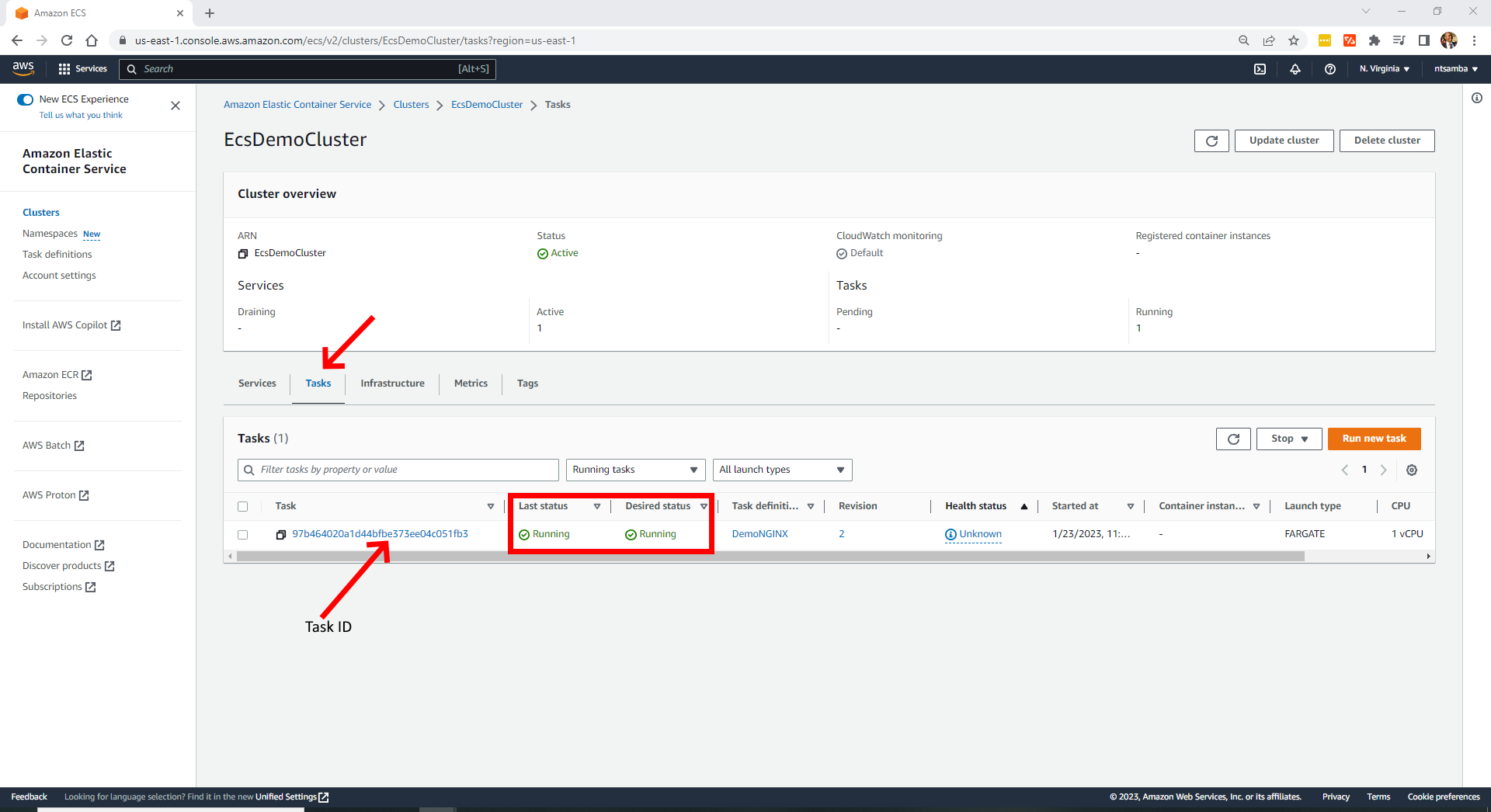
After a few minutes, you’ll see the “nginx-ecs-service-demo” service.

If the service deployment goes through successfully, it will notify you and run the task:

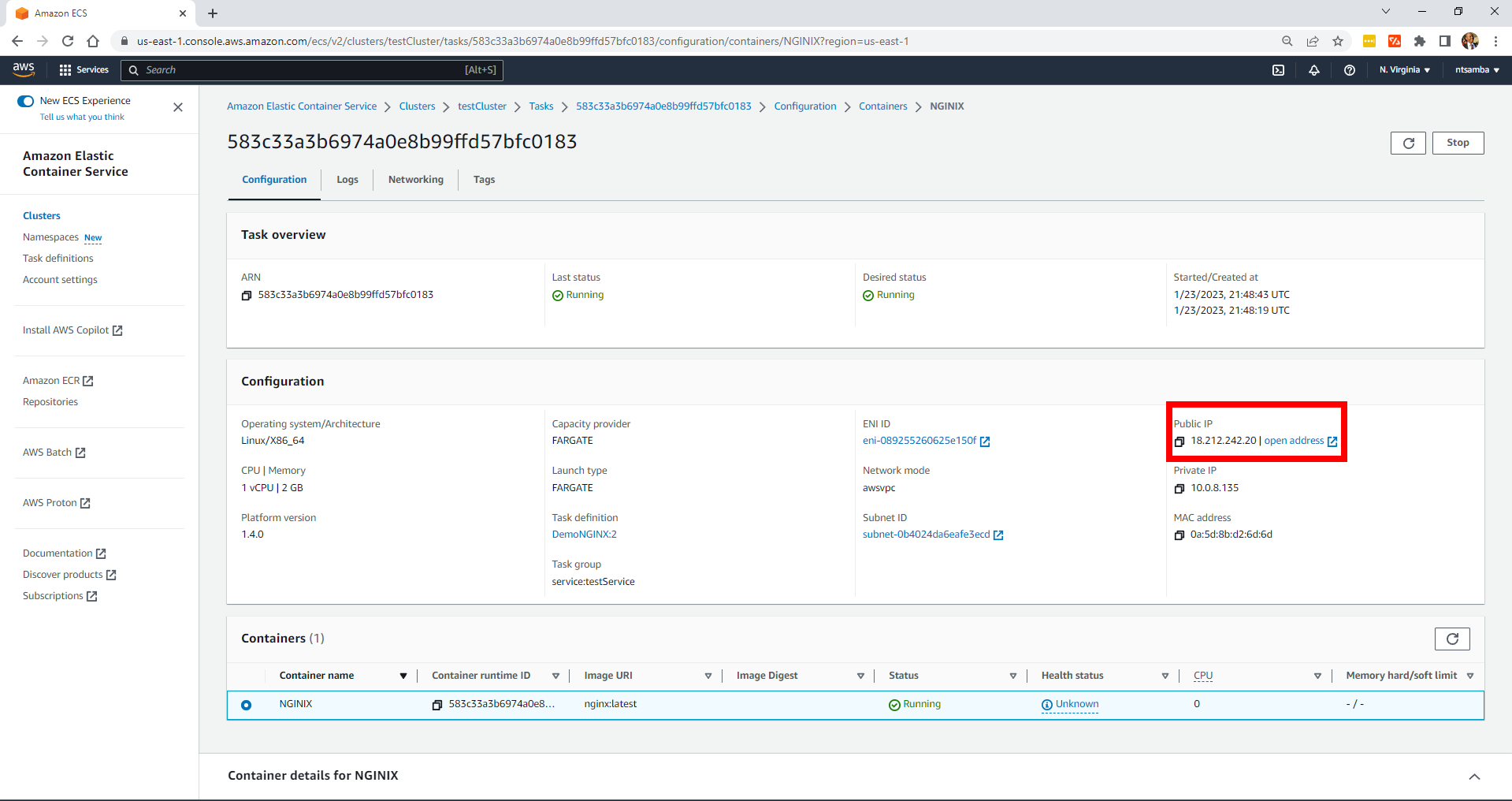


Service deployment was successful

To verify the service is running, select the **Tasks** tab:

**Tasks** tab

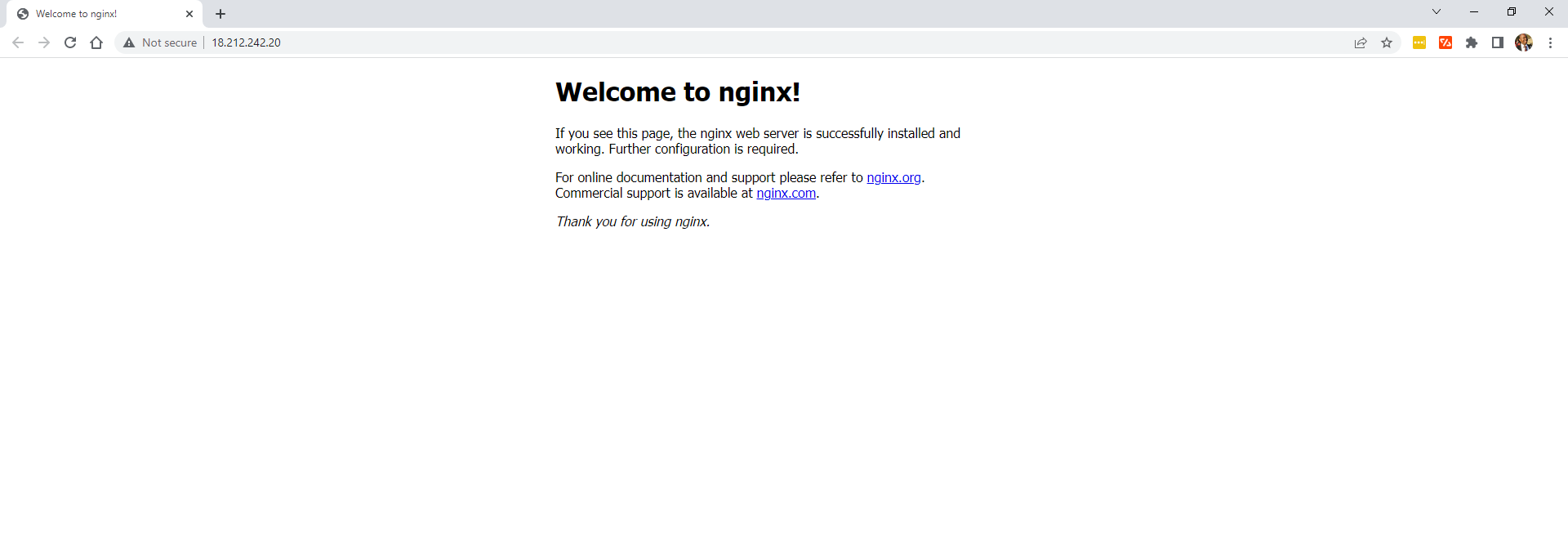
Check that both the **Last status** and **Desired status** are running and click on the **Task ID** to open the task. Then under **Configuration**, click on **open address** or copy the public IP address and paste it into your browser:



Public IP

## Access website

You should see your application running, and if it is i.e. showing you a *Welcome to NGINX page*, you’ve successfully deployed a container on Amazon ECS:



NGINX running. DONE