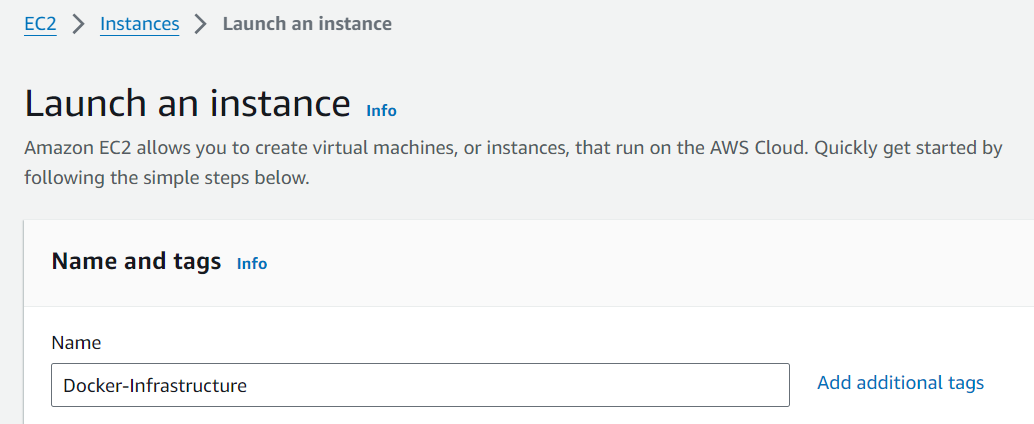
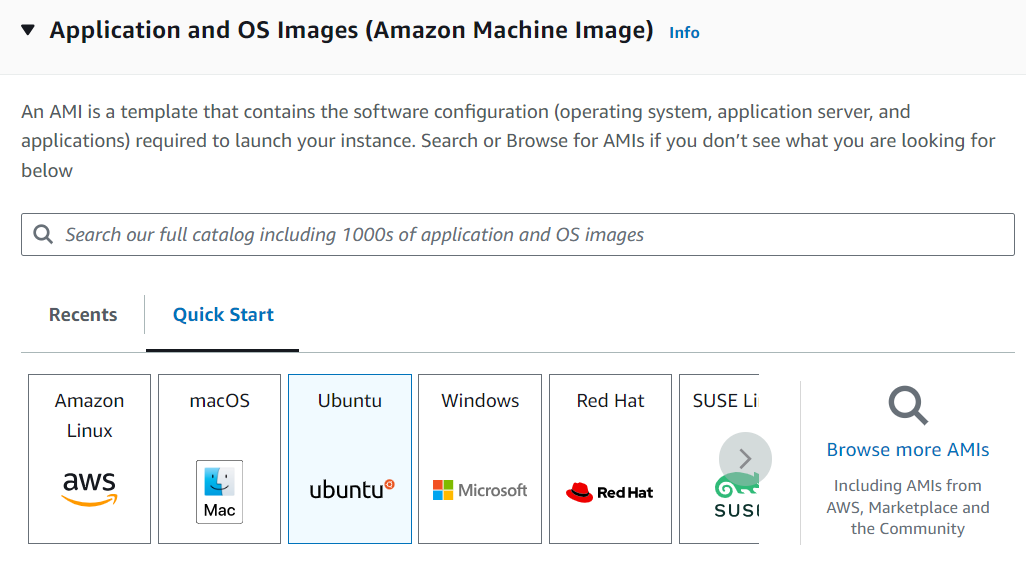
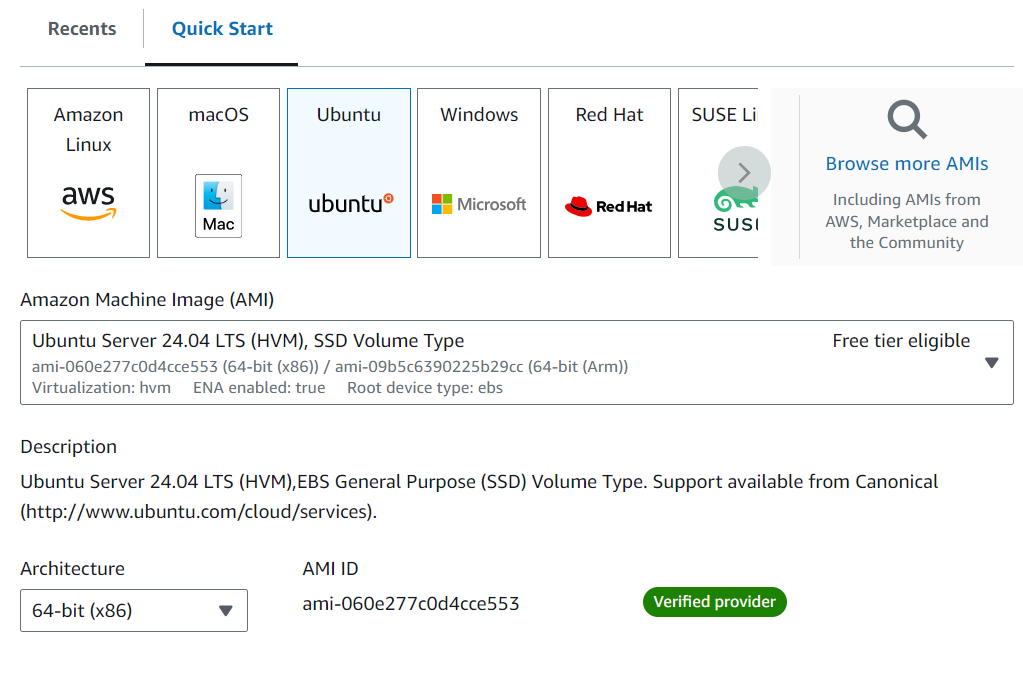
**AIM : Performing DevOps (Development Operations) with AWS Elastic Container Service (ECS)**

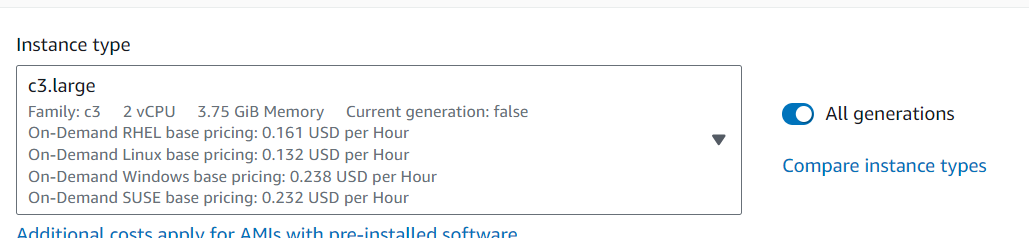
1. **Amazon ECS**:
   * Objective: Orchestrate the deployment and management of containerized applications, enabling easy scaling and efficient resource management.
2. **Amazon Elastic Container Registry (ECR)**:
   * Objective: Store, version, and manage Docker container images securely for seamless integration into ECS deployments.
3. **AWS Fargate**:
   * Objective: Run containers without managing the underlying infrastructure, providing a serverless environment that simplifies scaling and cost optimization.
4. **Amazon CloudWatch**:
   * Objective: Monitor and log ECS tasks, gather performance metrics, and set up alarms to ensure operational health and performance insights.
5. **IAM (Identity and Access Management)**:
   * Objective: Control access to ECS, ECR, and related AWS services with least-privilege permissions for secure, role-based access to resources.

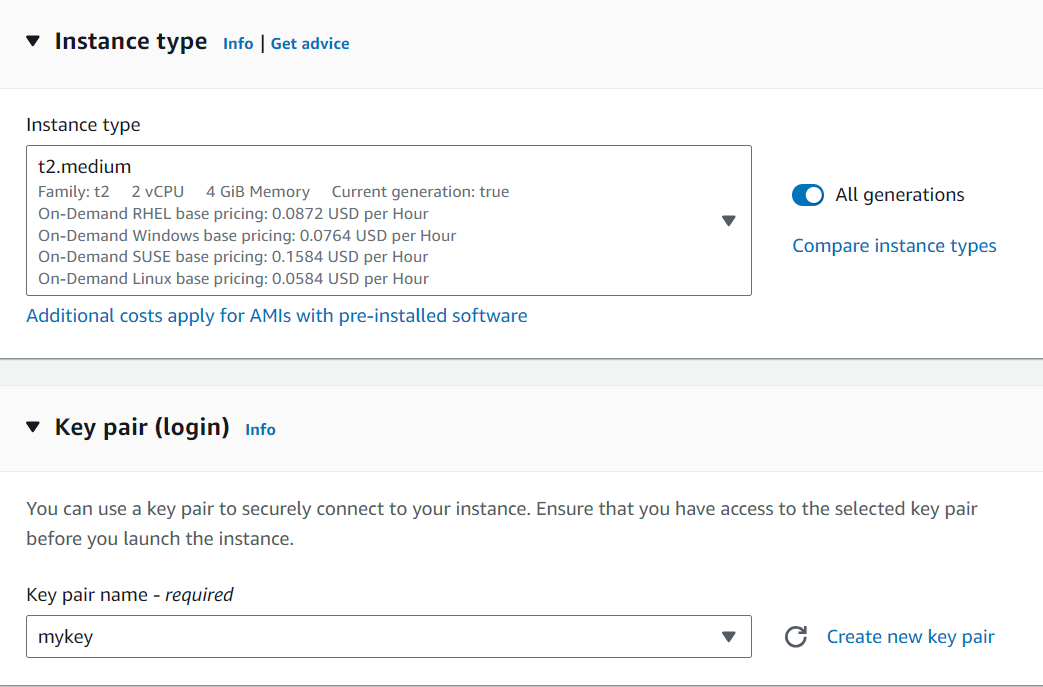
These services work together to create a scalable, secure, and easy-to-manage DevOps pipeline for deploying containerized applications.

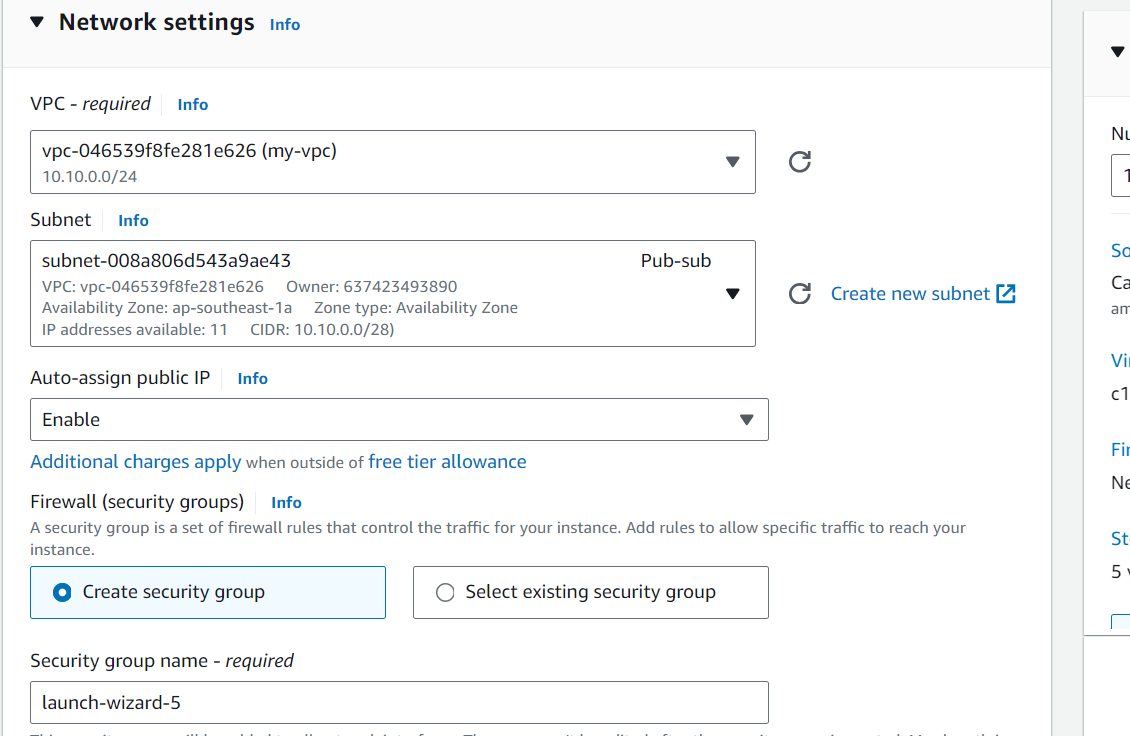
****

****

****

****

****

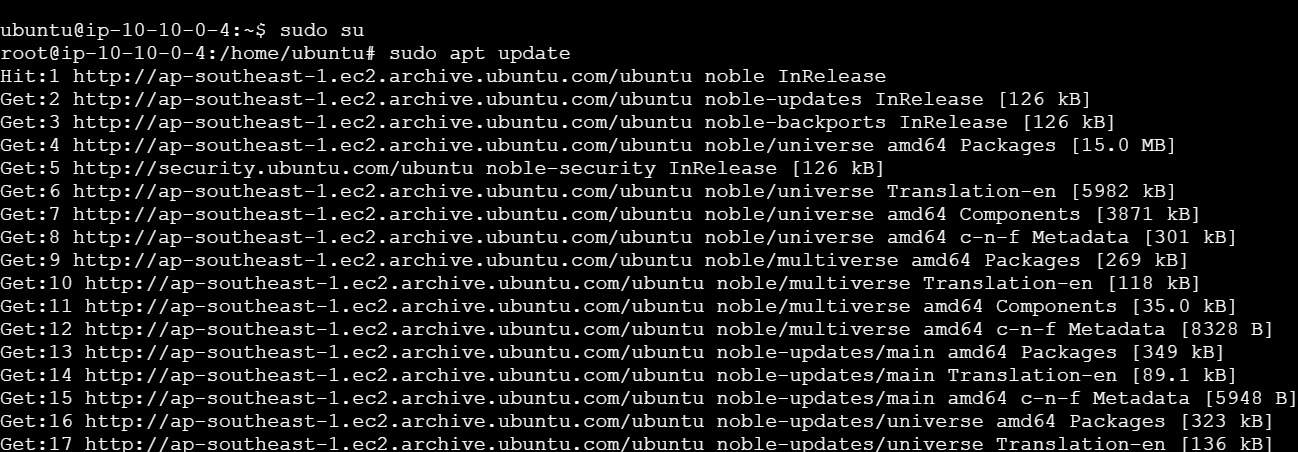
****

**Step 1 : Installing Docker**

The Docker installation package available in the official Ubuntu repository may not be the latest version. To ensure we get the latest version, we’ll install Docker from the official Docker repository. To do that, we’ll add a new package source, add the GPG key from Docker to ensure the downloads are valid, and then install the package.

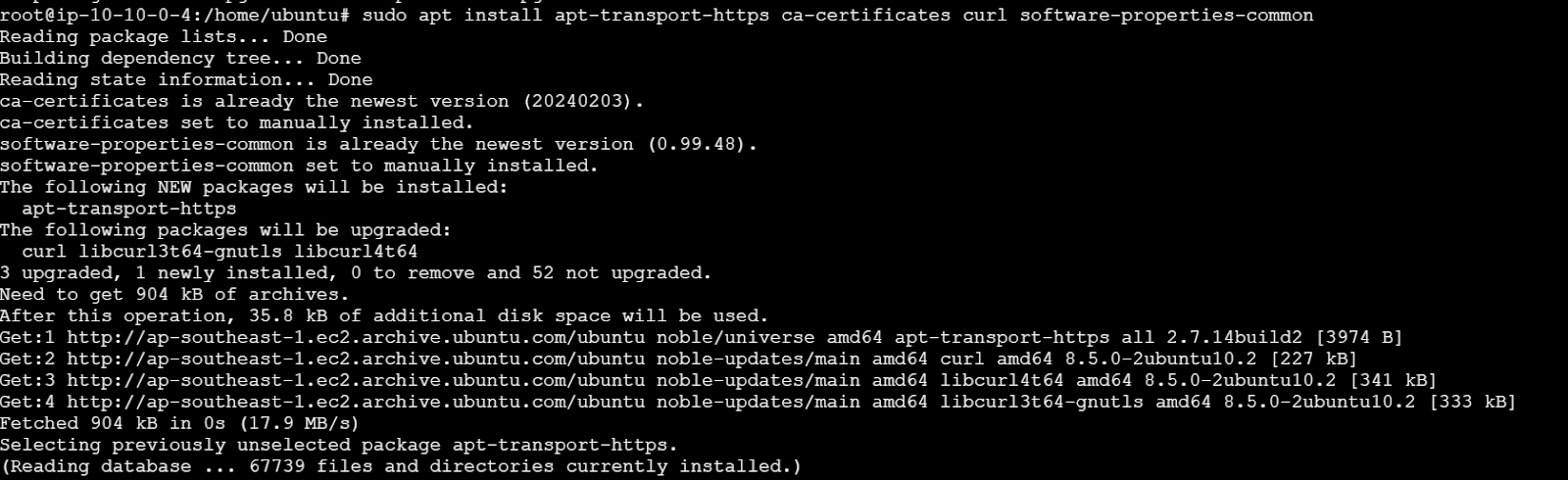
First, update your existing list of packages:

**> sudo apt update**

****

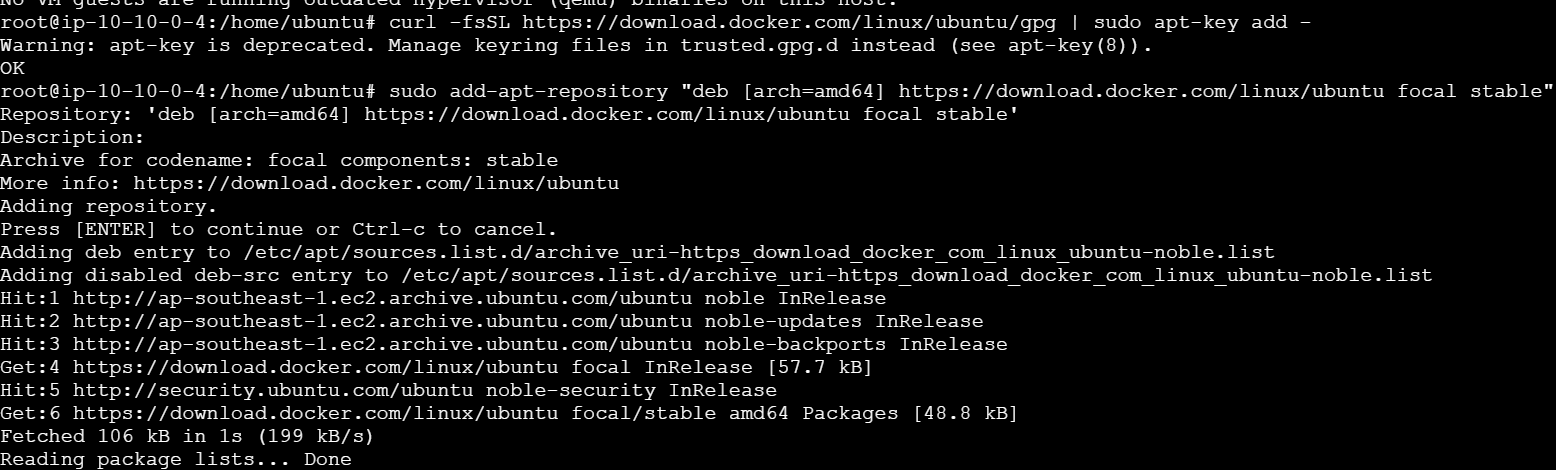
**Next, install a few prerequisite packages which let apt use packages over HTTPS:**

**sudo apt install apt-transport-https ca-certificates curl software-properties-common**

****

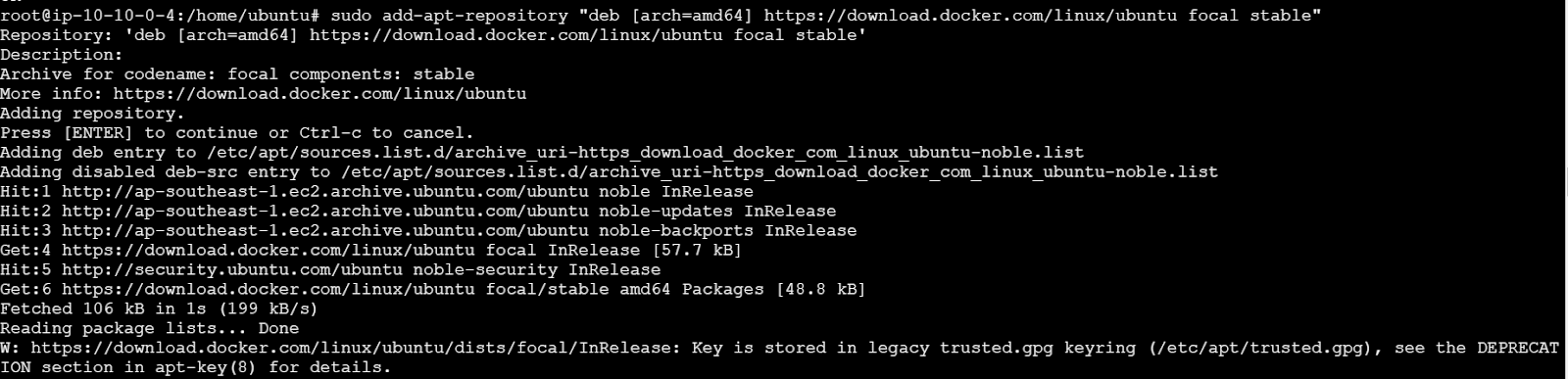
**Then add the GPG key for the official Docker repository to your system:**

**curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -**

****

Add the Docker repository to APT sources:

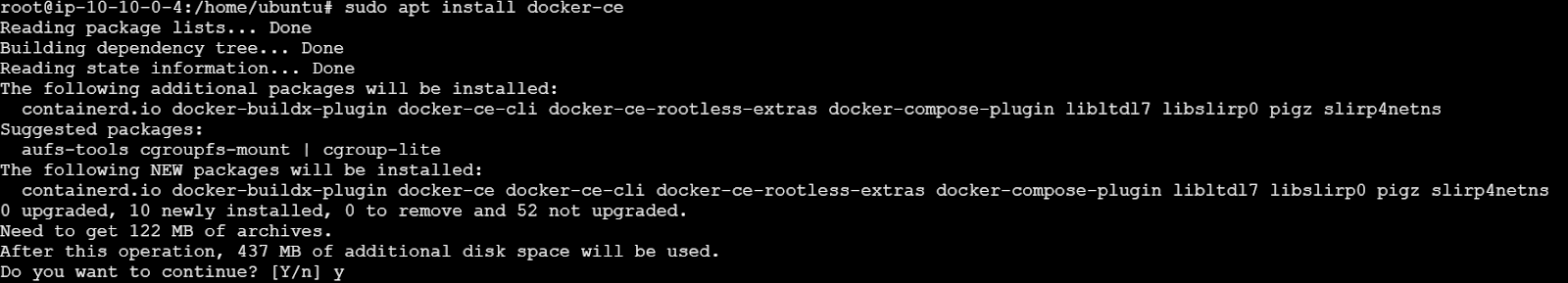
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"



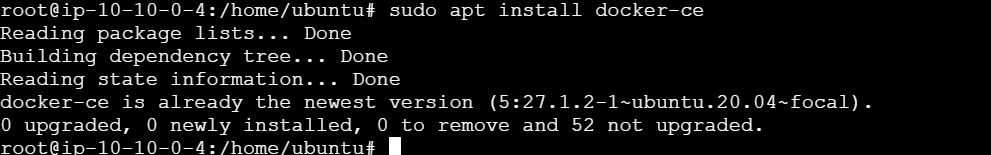
This will also update our package database with the Docker packages from the newly added repo.

Make sure you are about to install from the Docker repo instead of the default Ubuntu repo:

apt-cache policy docker-ce

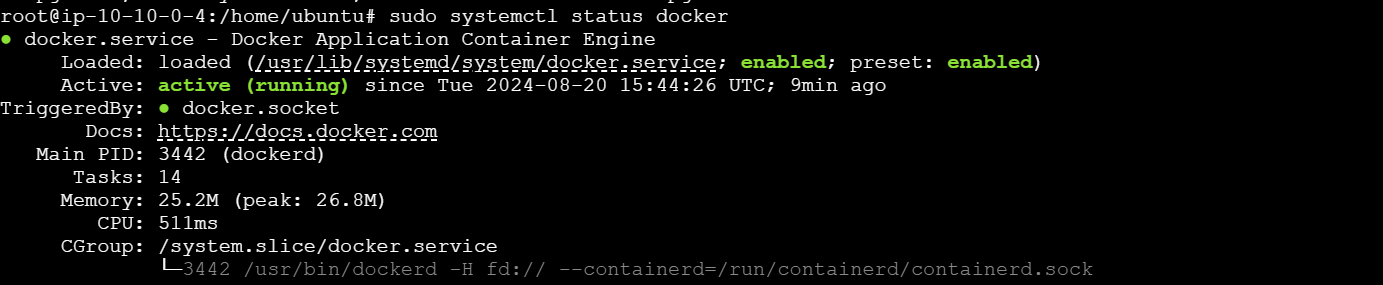


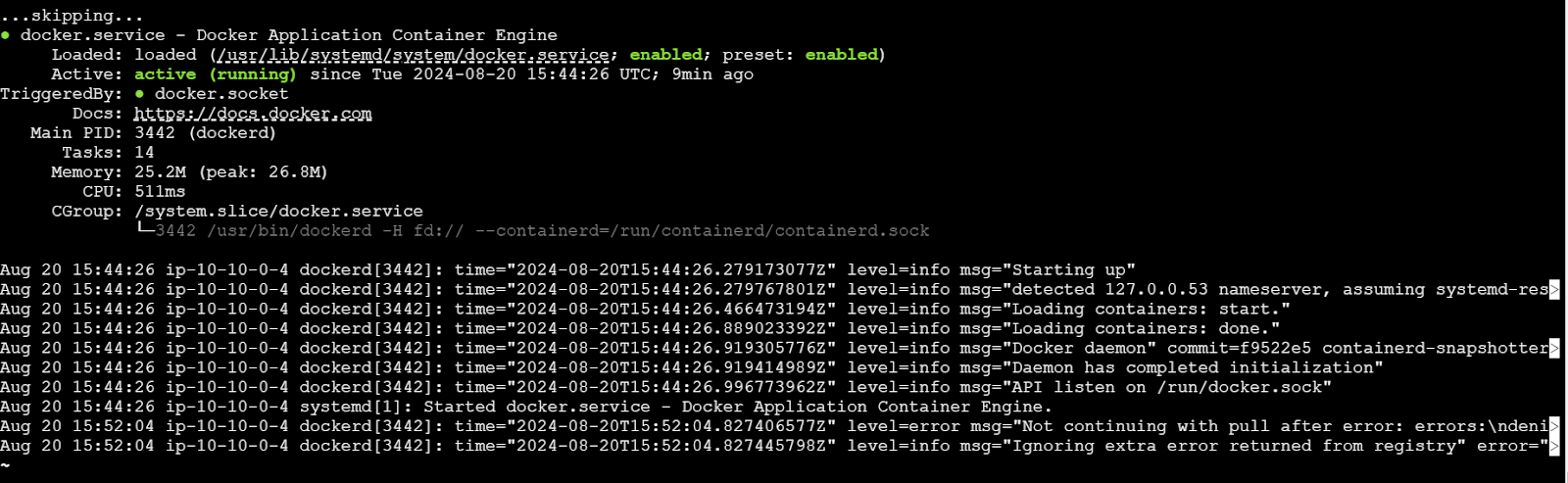
Finally, install Docker:

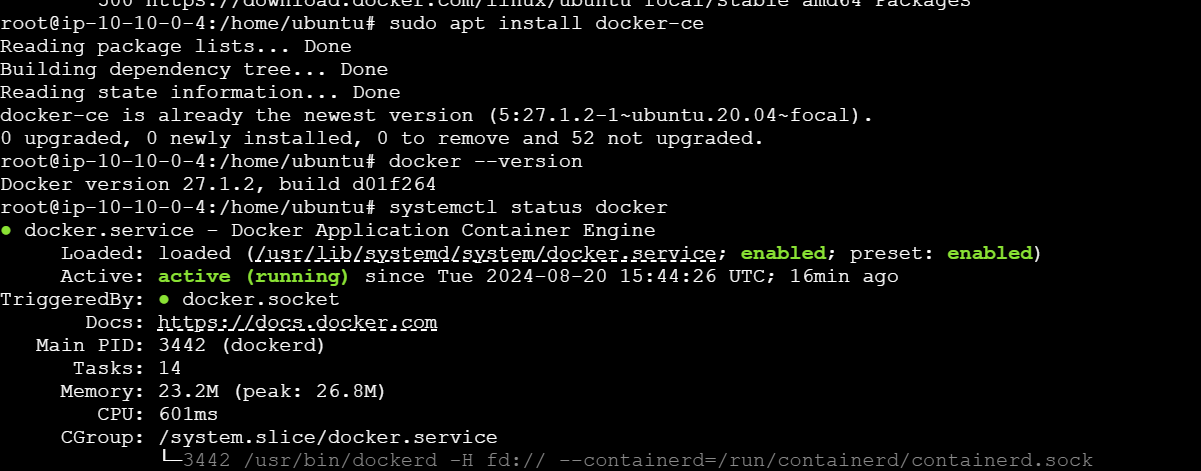
****

**Docker should now be installed, the daemon started, and the process enabled to start on boot. Check that it’s running:**

**sudo systemctl status docker**

****

****

****

**# Use Ubuntu 22.04 as the base image**

**FROM ubuntu:22.04**

**# Update installed packages and install Apache**

**RUN apt update && \**

**apt install -y apache2 && \**

**rm -rf /var/lib/apt/lists/\* && \**

**service apache2 start**

**# Write hello world message**

**RUN echo 'Hello World' > /var/www/html/index.html**

**# Create a custom Apache configuration file to set the ServerName**

**RUN echo 'ServerName localhost' > /etc/apache2/conf-available/servername.conf && \**

**a2enconf servername**

**# Configure Apache**

**RUN mkdir -p /var/run/apache2 /var/lock/apache2 && \**

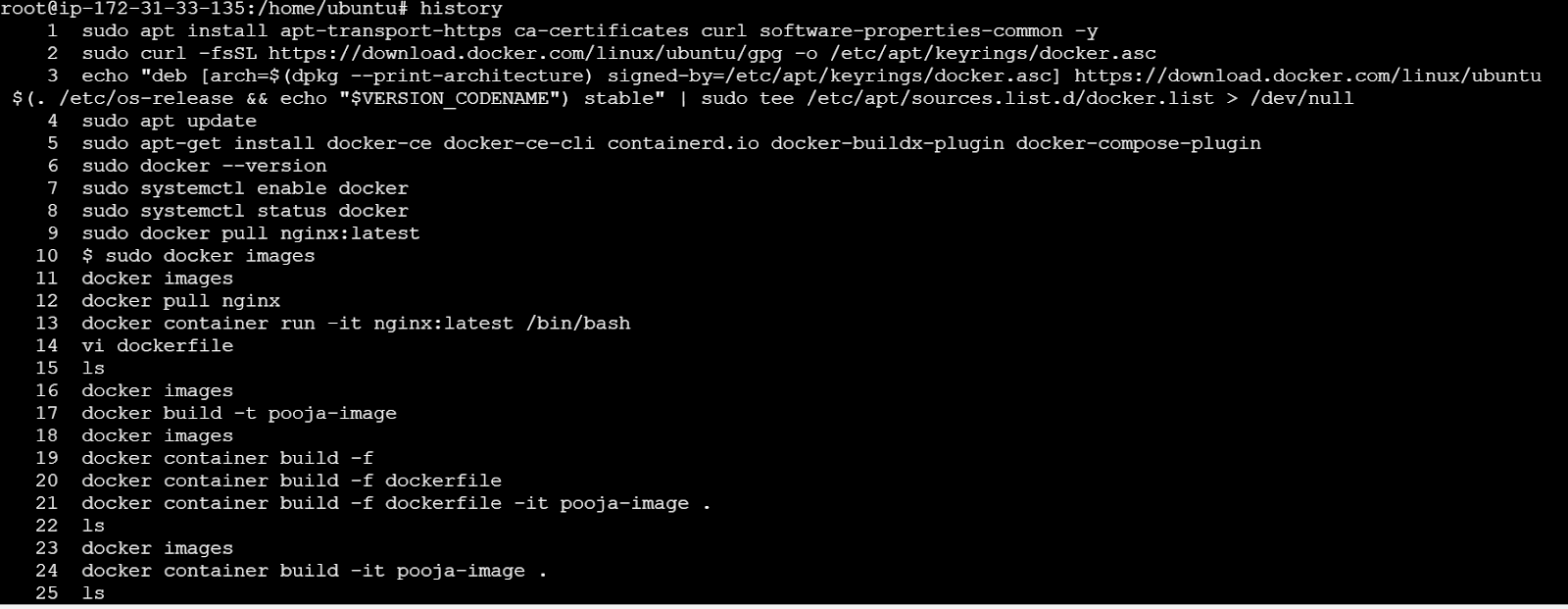
**echo '#!/bin/bash\n' \**

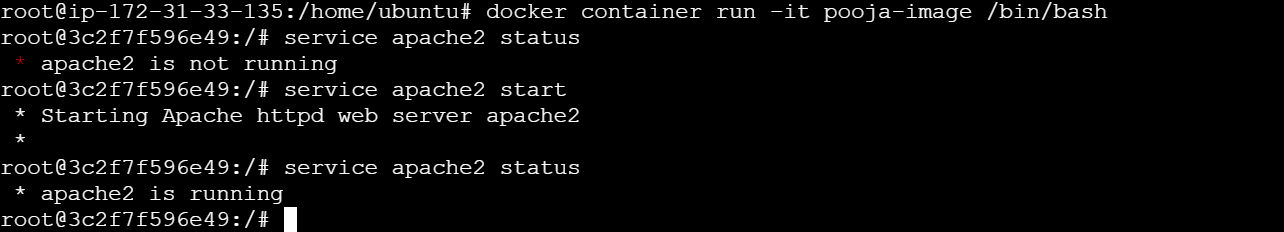
**'service apache2 start && /usr/sbin/apachectl -D FOREGROUND' > /usr/local/bin/run\_apache.sh && \**

**chmod +x /usr/local/bin/run\_apache.sh**

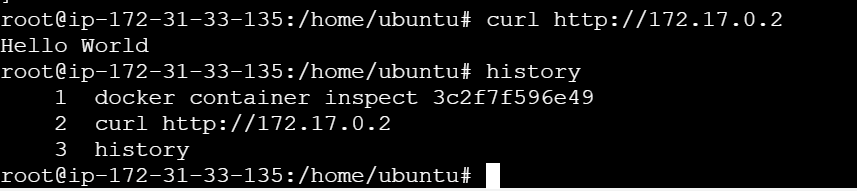
**EXPOSE 80**

**CMD ["/usr/local/bin/run\_apache.sh"]**

****

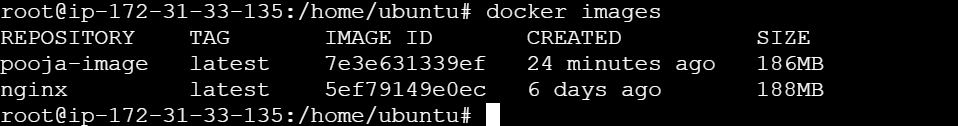
****

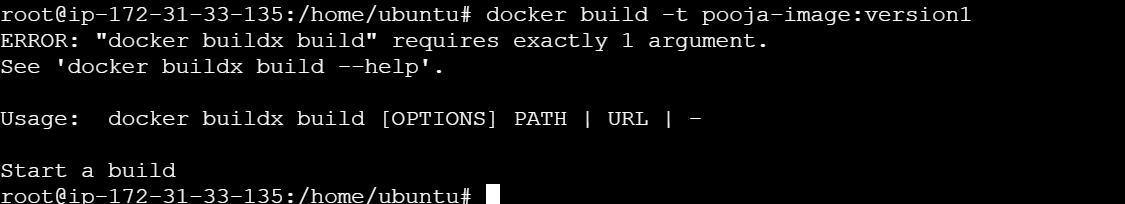
**In another session**

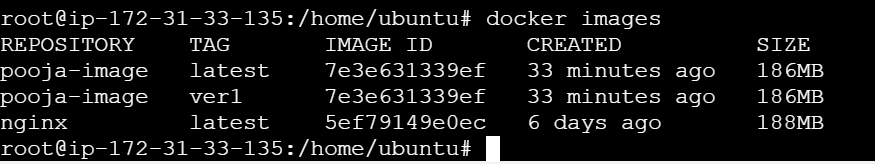
****

**Vi dockerfile**

****

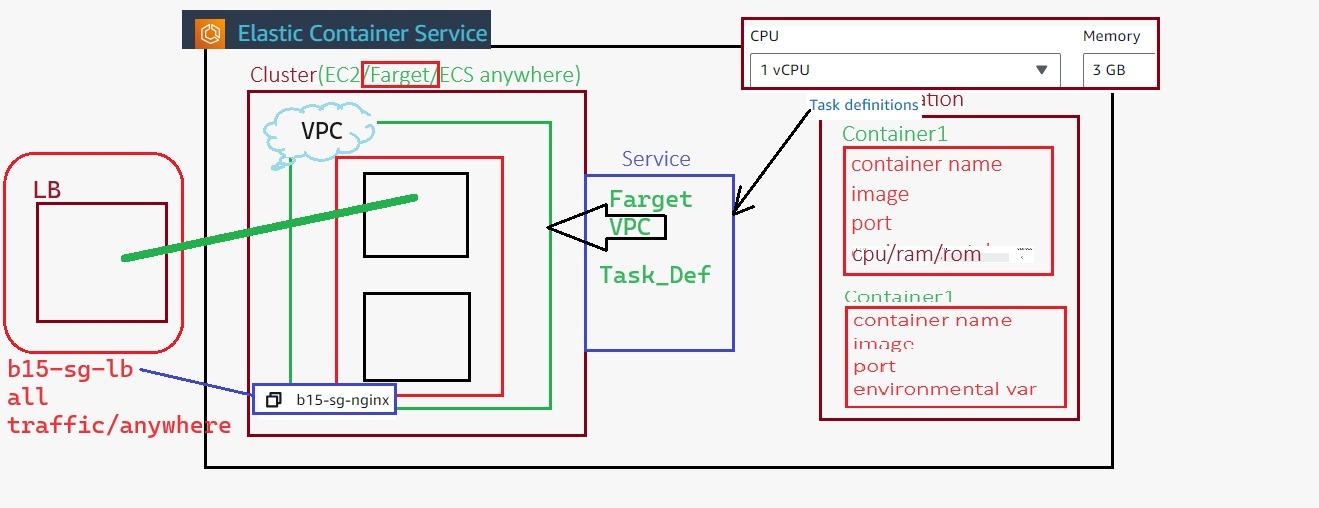
****

****

****

**Aim : configuring AWS Elastic Container Service (ECS)**

**Architecture**



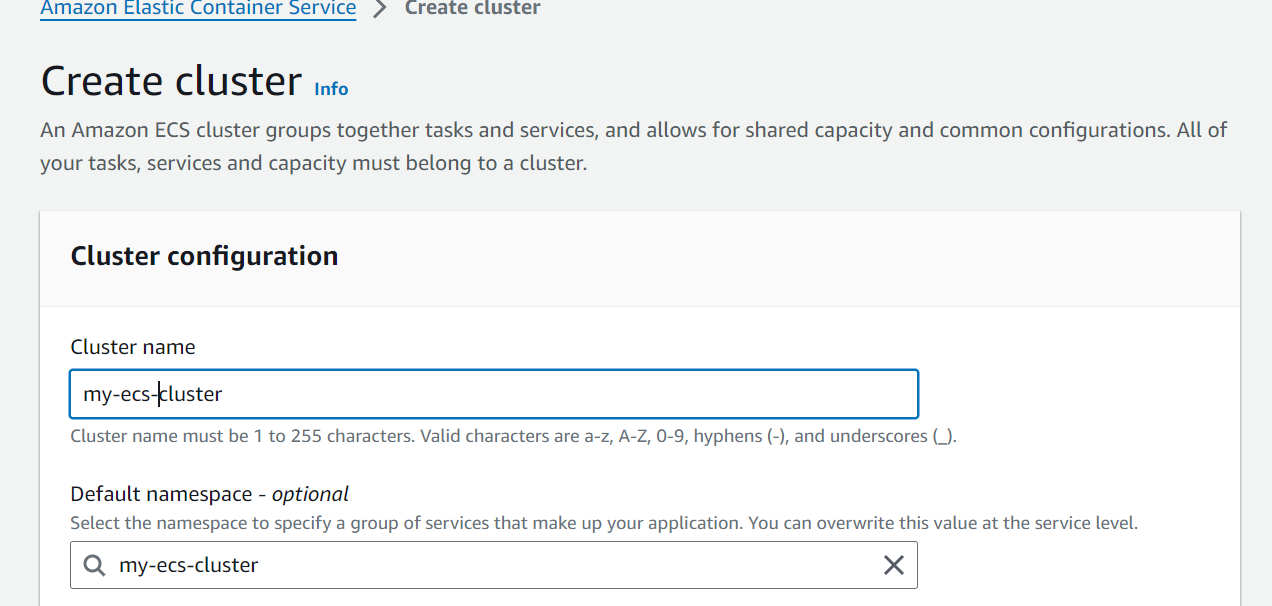
### 

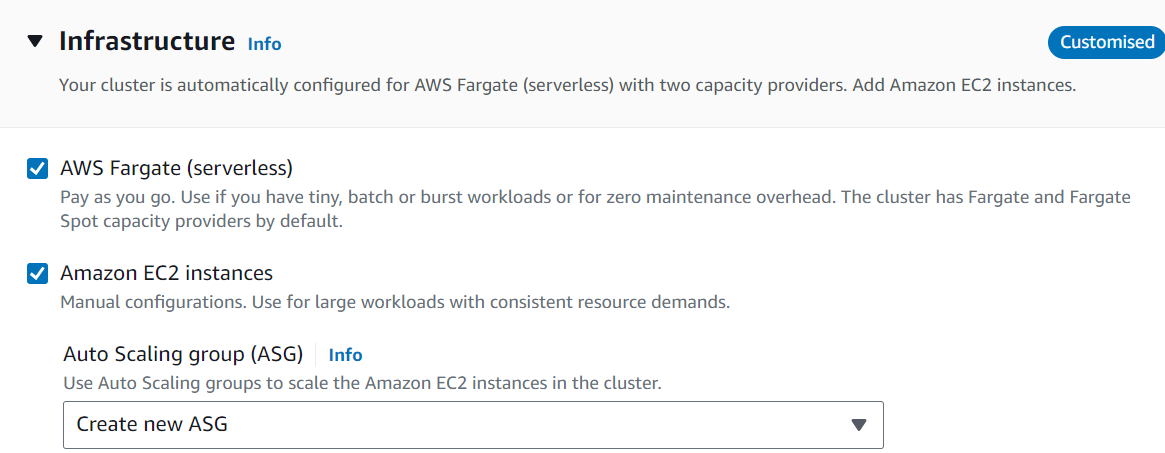
### **1. Set Up the AWS Environment**

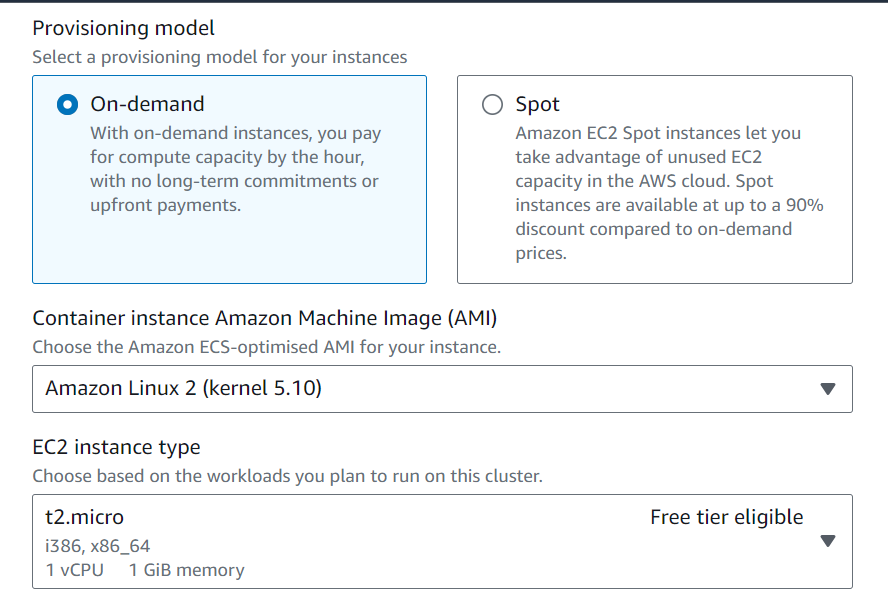
* **Log in to AWS Console: Log in to your AWS account.**
* **Navigate to ECS: Go to the AWS Management Console and search for "Elastic Container Service" (ECS).**

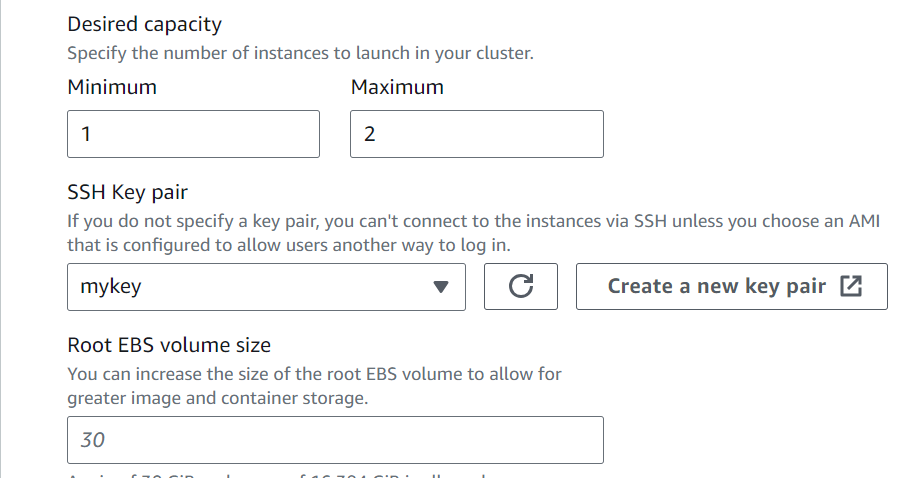
### **2. Create a Cluster**

* **Launch ECS Cluster:**
  + **Select "Clusters" from the ECS menu.**
  + **Click on "Create Cluster."**

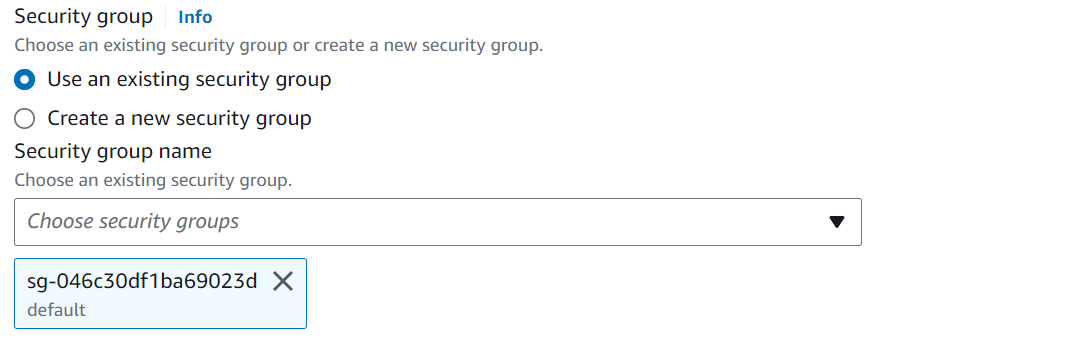


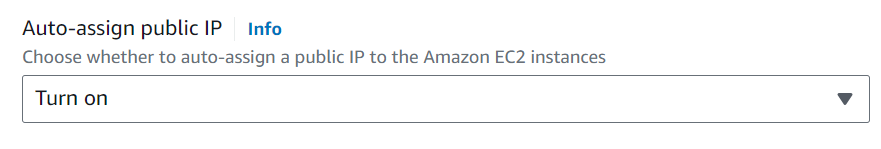


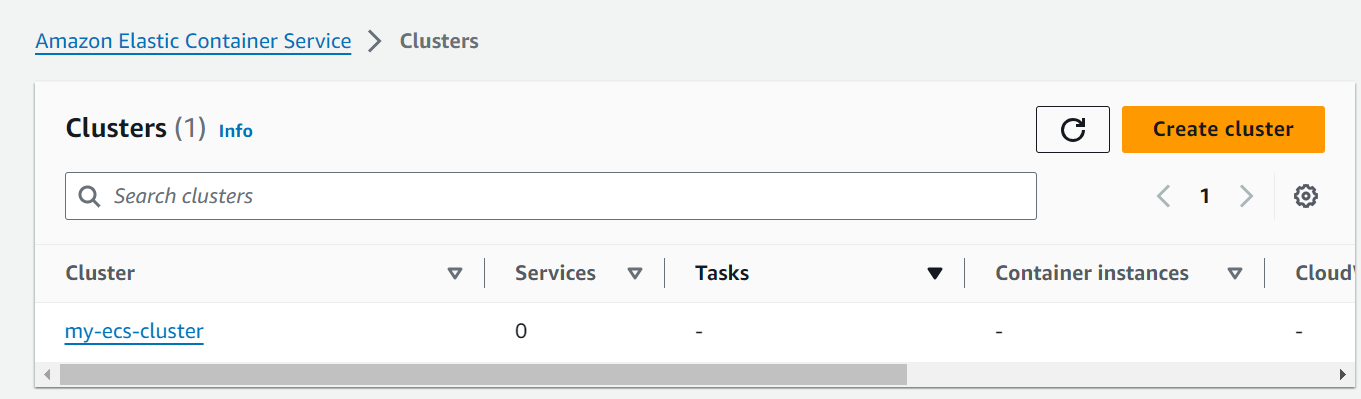




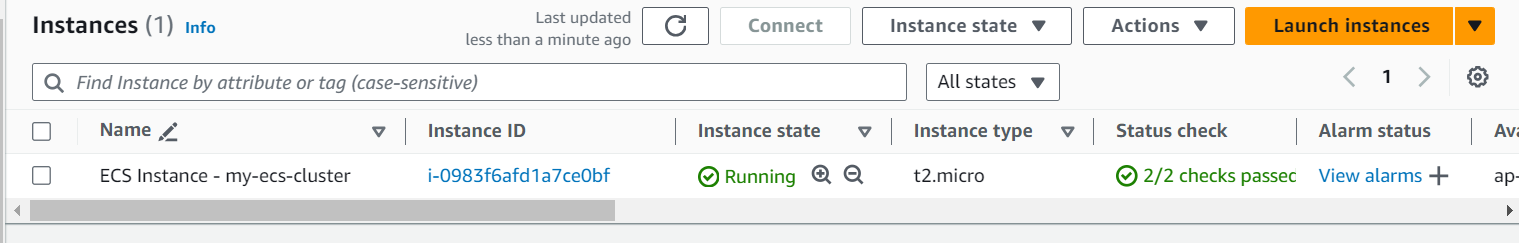






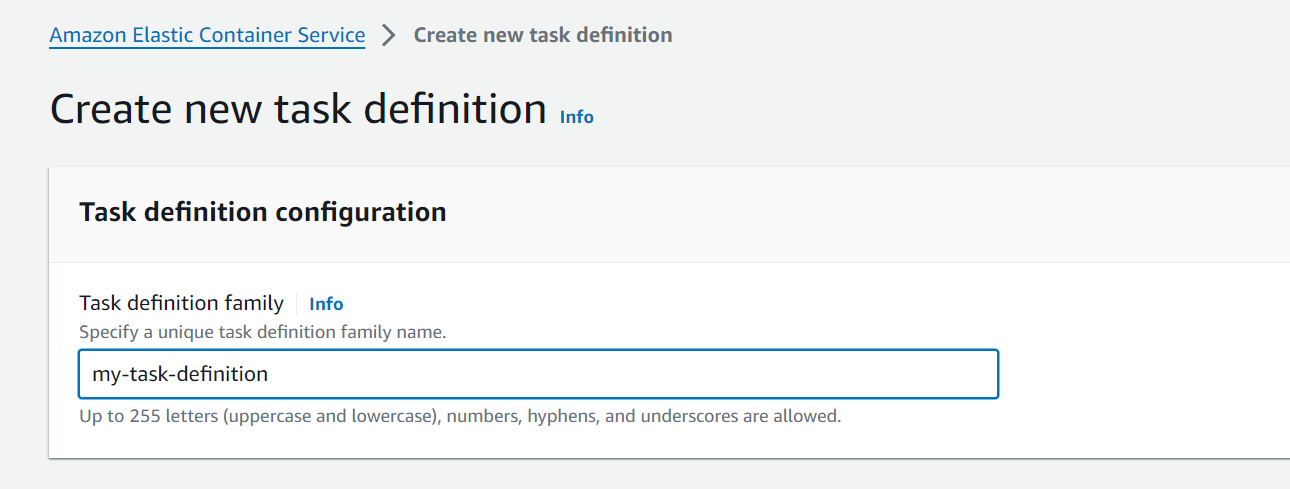


An EC2 instance will be created

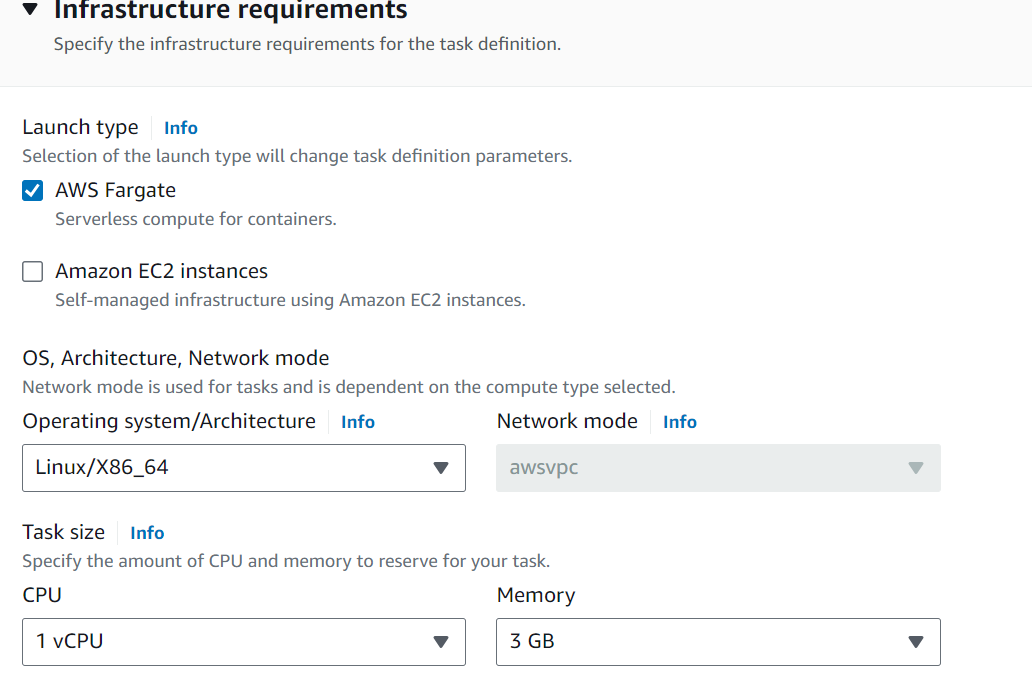


### **3. Define a Task Definition**

* **Create a New Task Definition**:
  + In the ECS menu, select "Task Definitions."
  + Click on "Create new Task Definition."



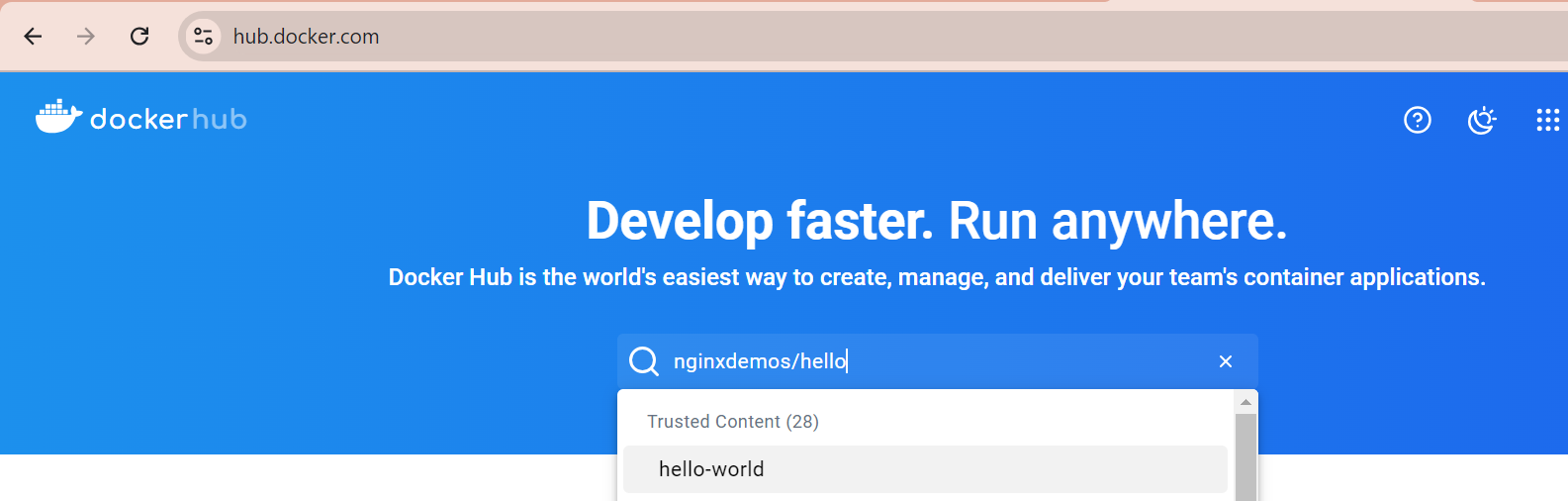
* + Choose "Fargate" as the launch type.



**Configure Task Settings**:

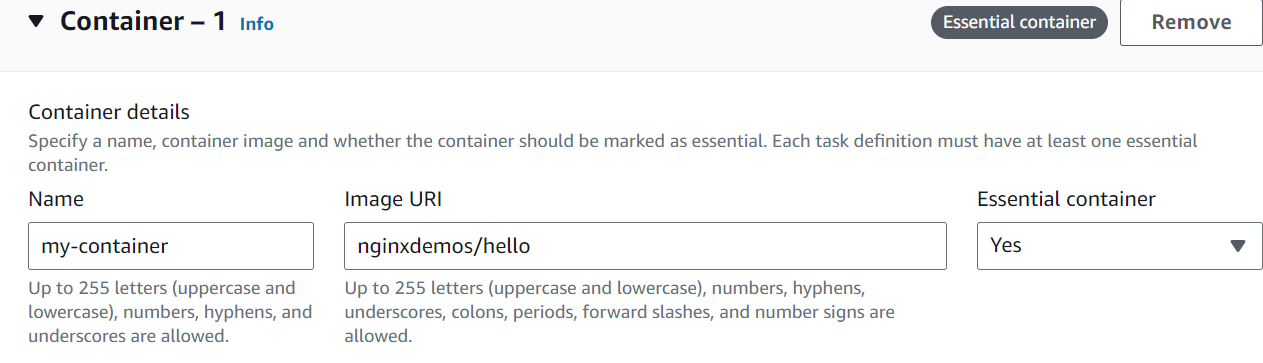
* Specify the task definition name (e.g., Task\_Def).
* Set the vCPU and memory according to your requirements. In the diagram, it's shown as 1 vCPU and 3 GB memory.

**Docker Hub >**



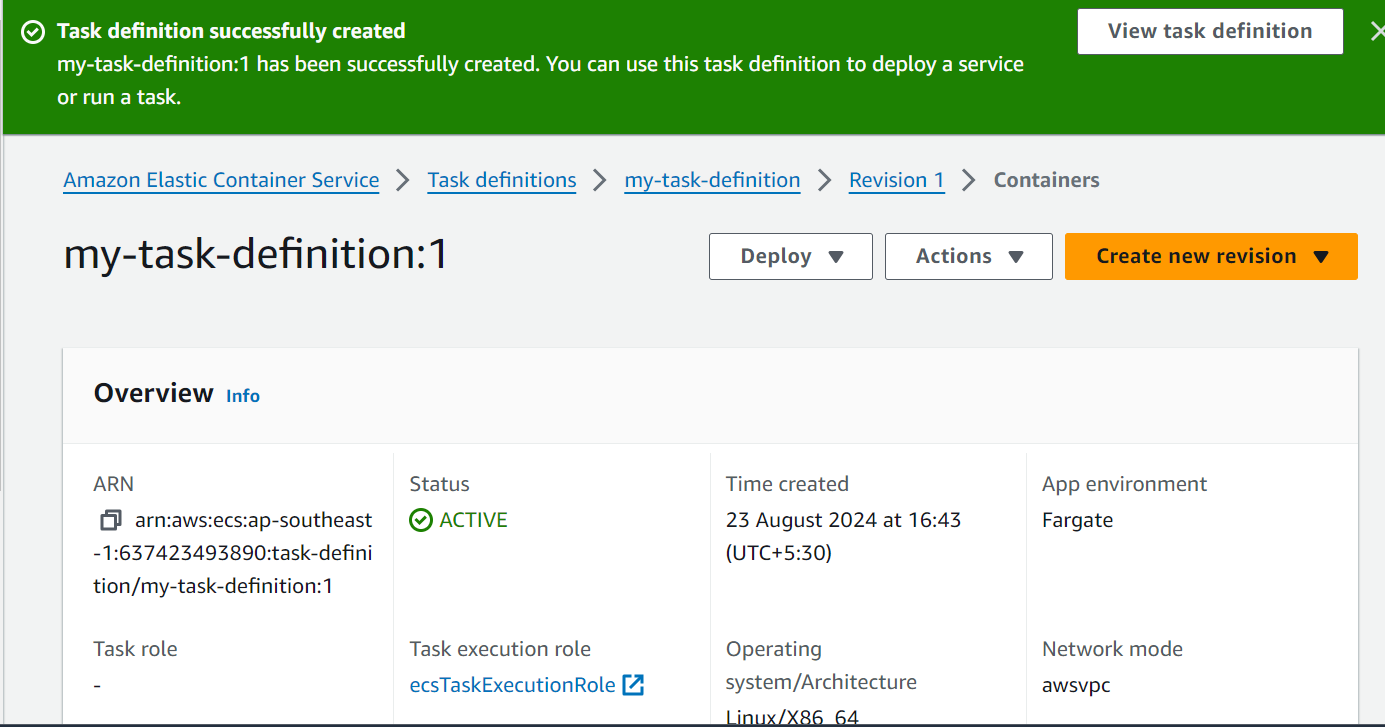
**Add Container Definitions**:

* Add a container (e.g., Container1) by providing the container name, image (e.g., nginx), and port mapping. Include any environmental variables if necessary.
* Repeat for additional containers if needed.



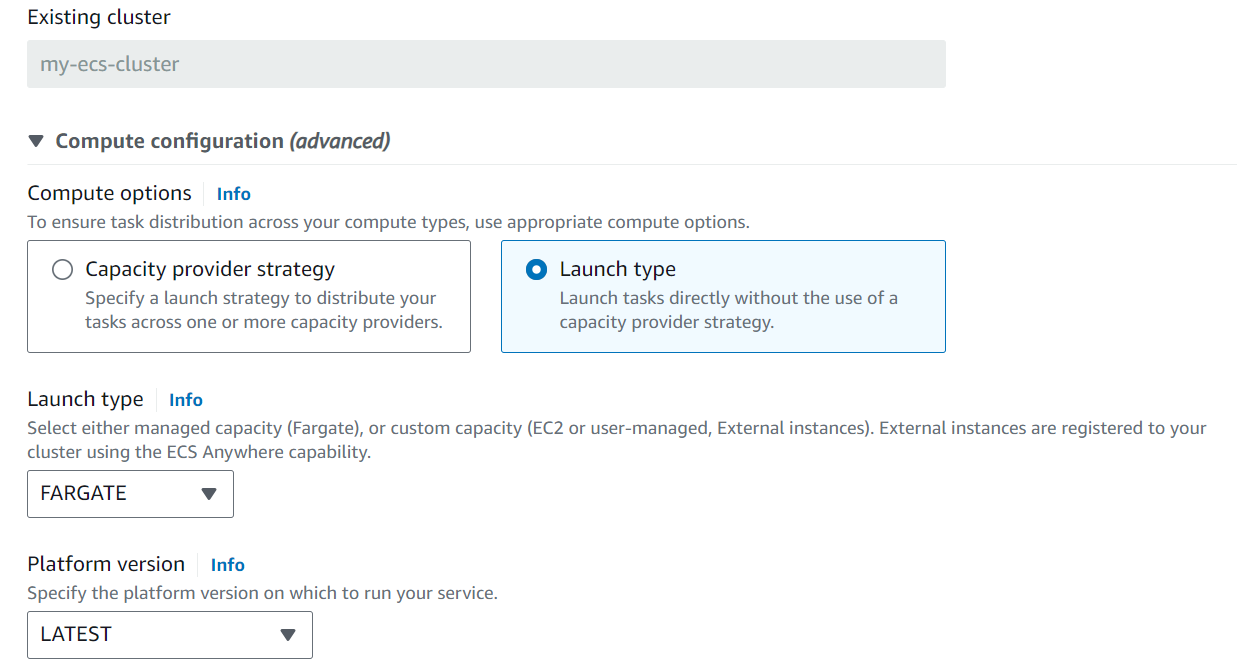
my-container-80-tcp

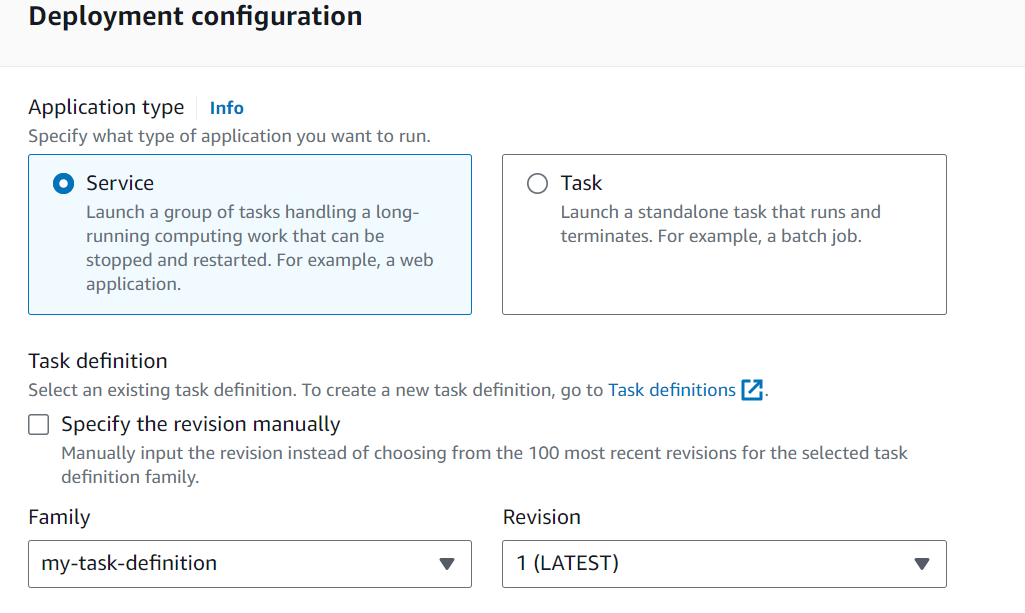




### **4. Create and Configure a Service**

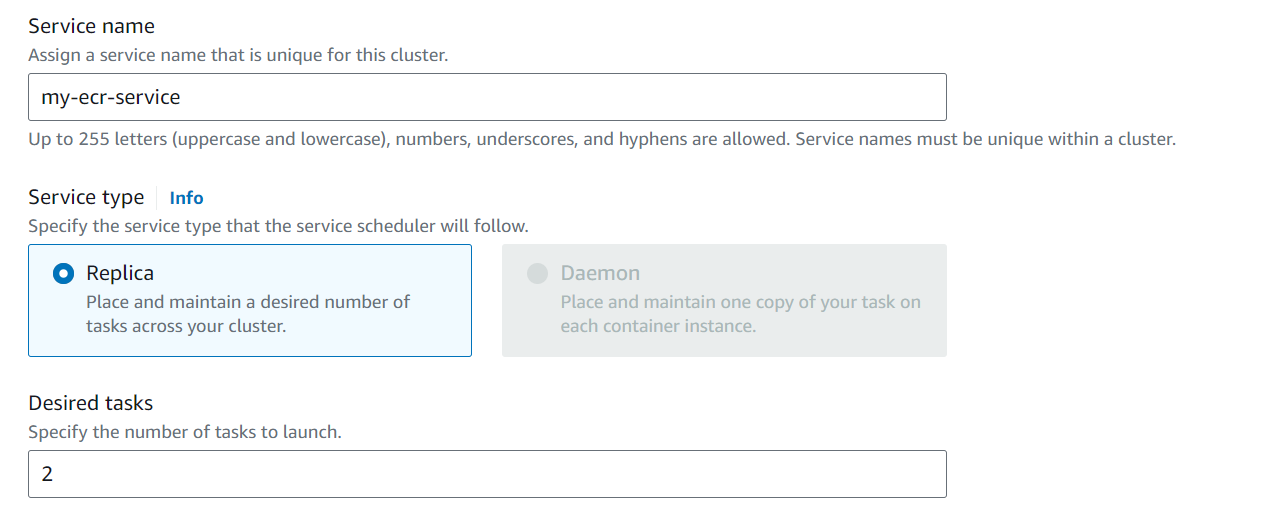
* **Service Creation**:
  + Go back to the ECS dashboard and select "Clusters."
  + Choose the cluster you created earlier.
  + Click on "Create" under "Services."

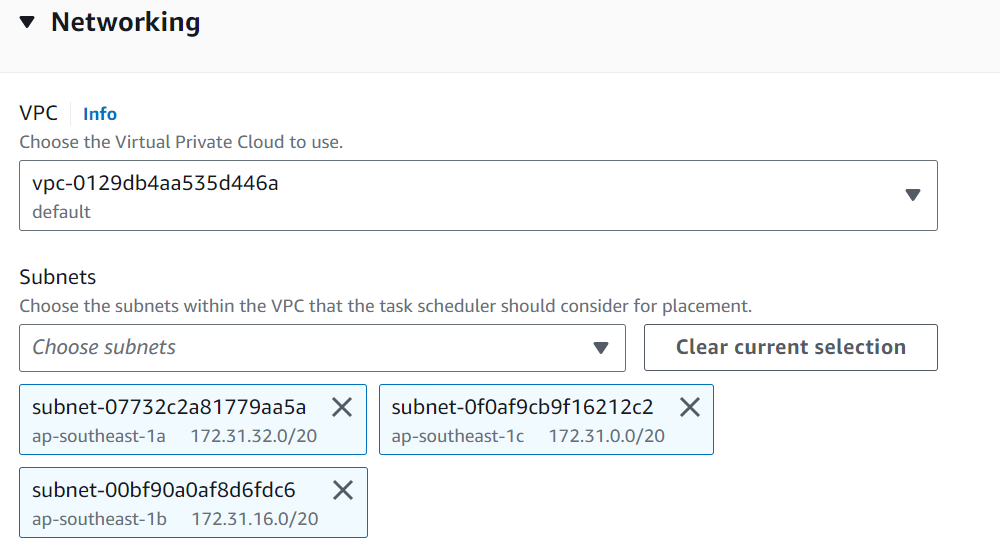




**Service Settings:**

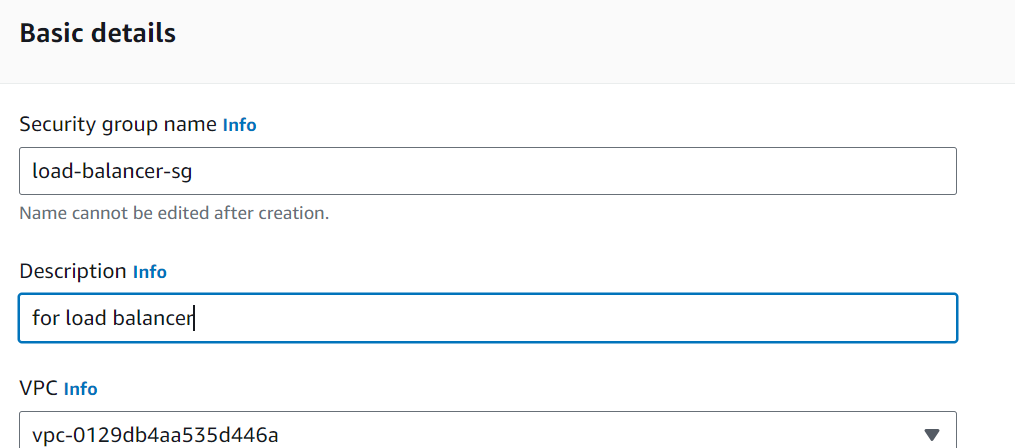
* **Select the launch type as "Fargate."**
* **Use the task definition created in the previous step (e.g., Task\_Def).**
* **Set the number of tasks (for high availability, set at least 2 tasks).**
* **Assign a VPC (Virtual Private Cloud) and subnets as per the diagram.**
* **Choose a load balancer configuration. The diagram shows a load balancer (LB) associated with security group b15-sg-lb**

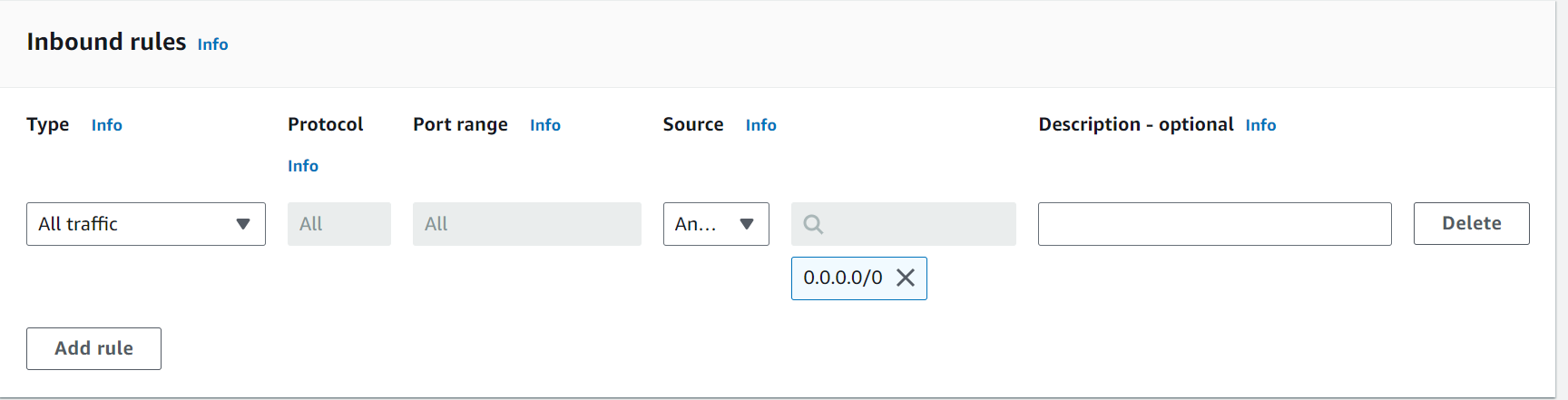




### **5. Configure Security Groups**

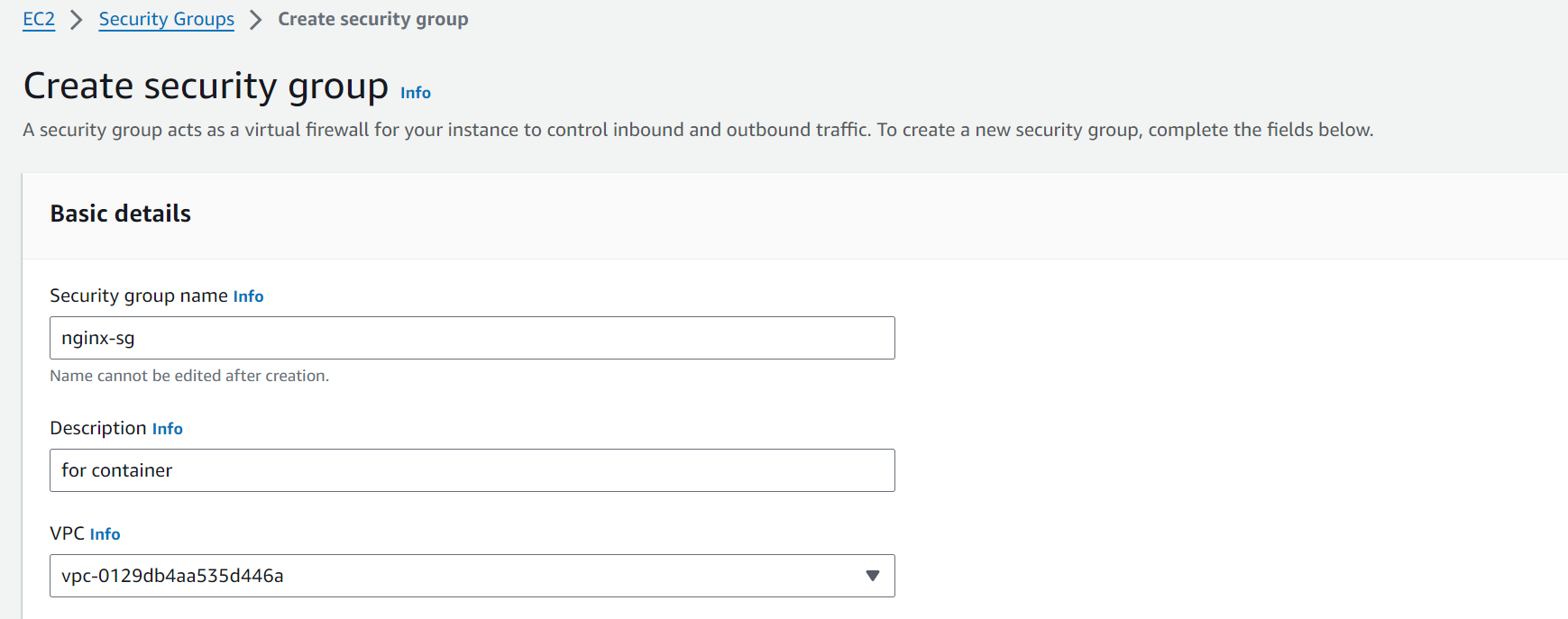
* **Load Balancer Security Group**:
  + Navigate to the "EC2" dashboard and select "Security Groups."
  + Create a new security group (e.g., b15-sg-lb) allowing inbound traffic on necessary ports (e.g., HTTP/HTTPS).
* **Container Security Group**:
  + Create a new security group (e.g., b15-sg-nginx) for your ECS tasks, allowing inbound traffic only from the load balancer.

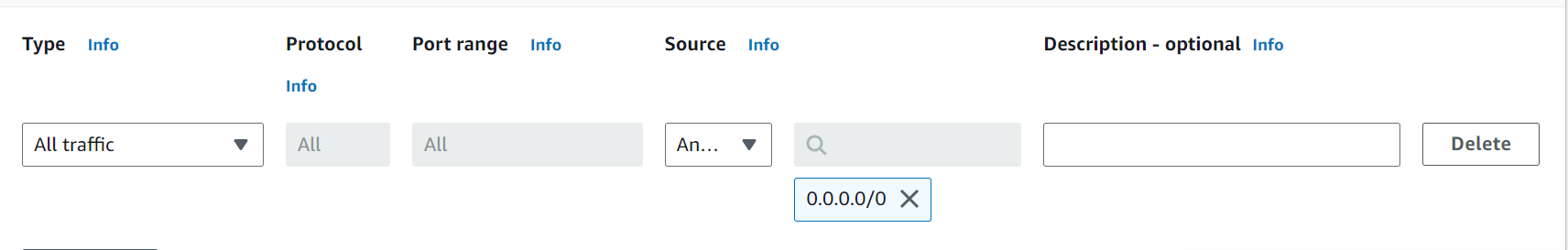


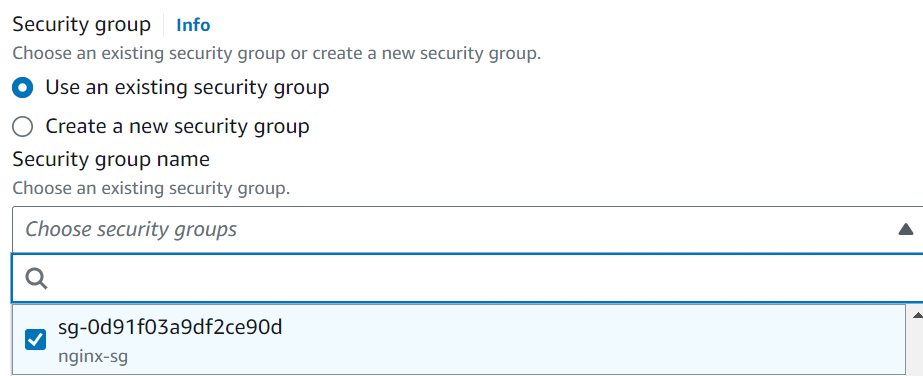


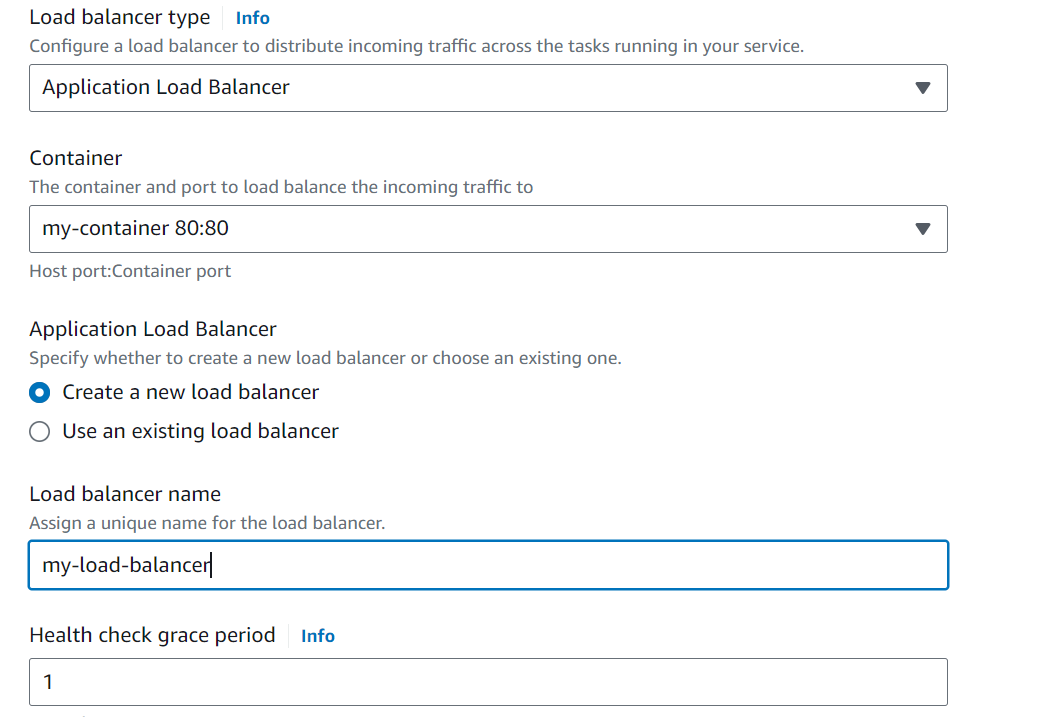
### **6. Attach Load Balancer to Service**

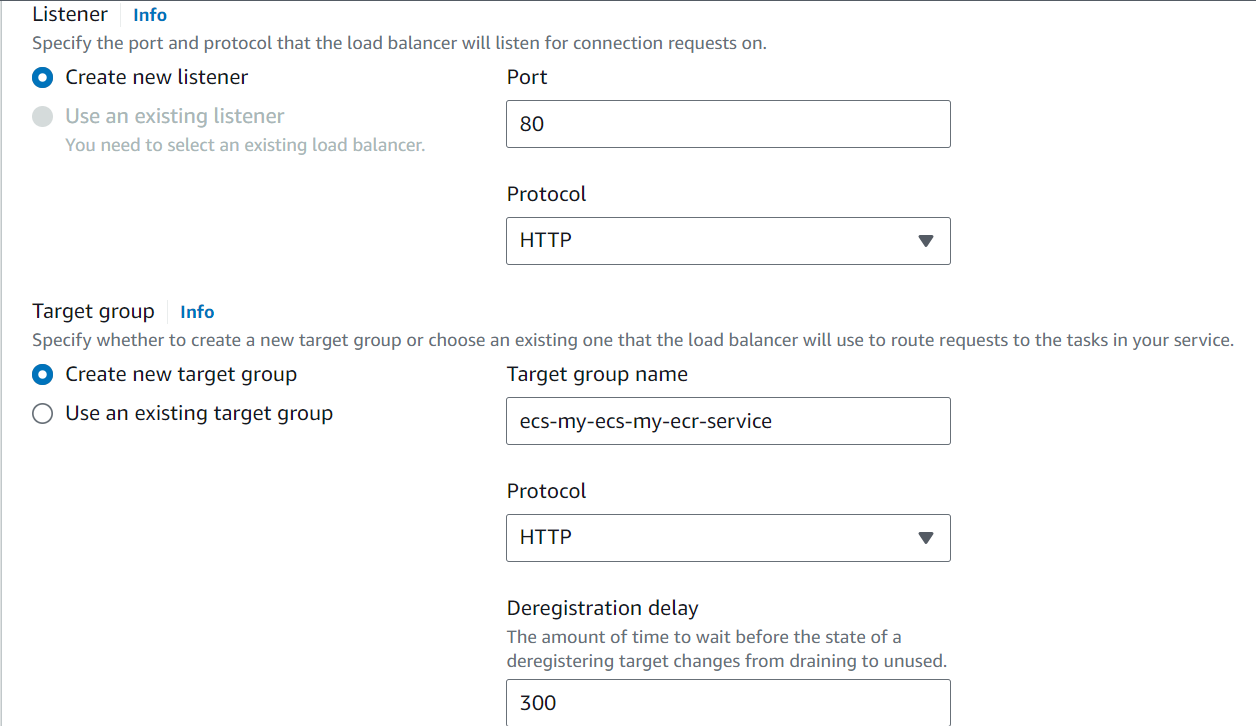
* **Register Load Balancer with ECS**:
  + In the service creation wizard, select "Application Load Balancer."
  + Choose the load balancer you set up earlier.
  + Define listeners for the ports and target groups as per your ECS service requirements.

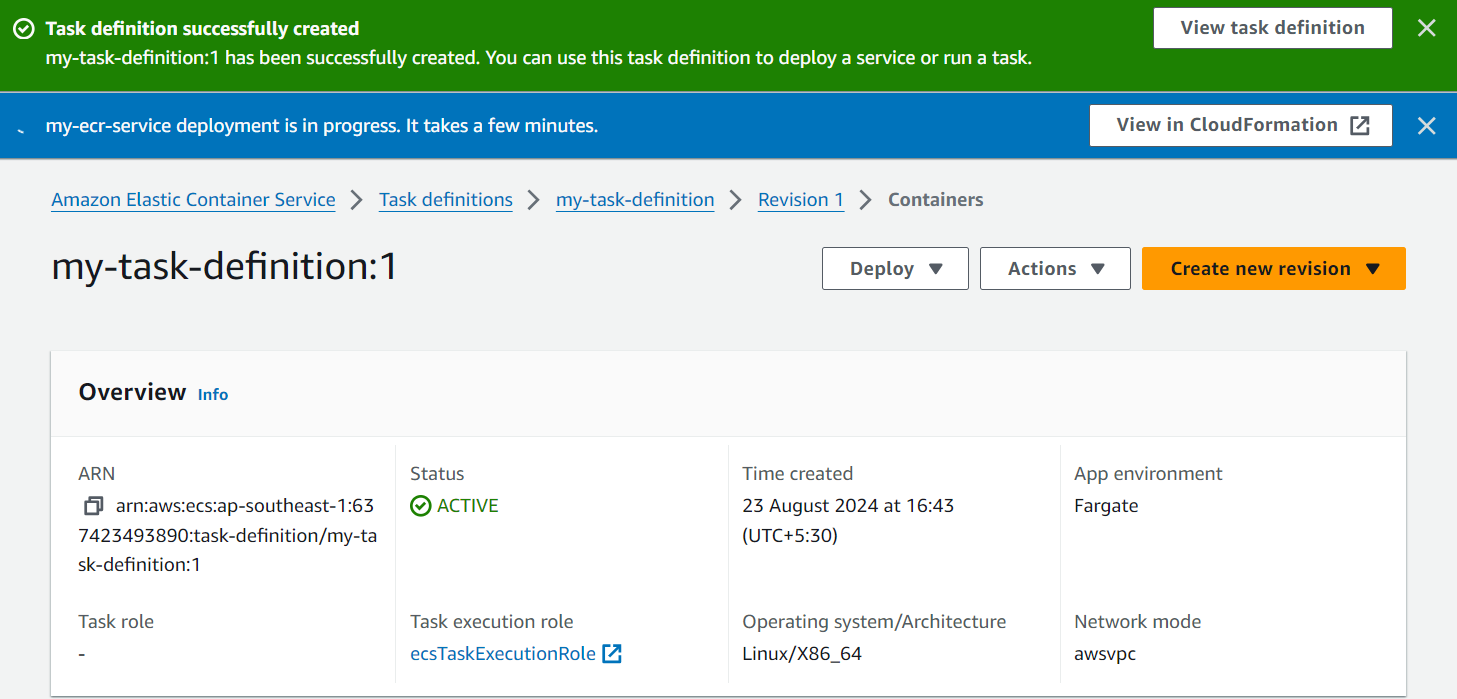




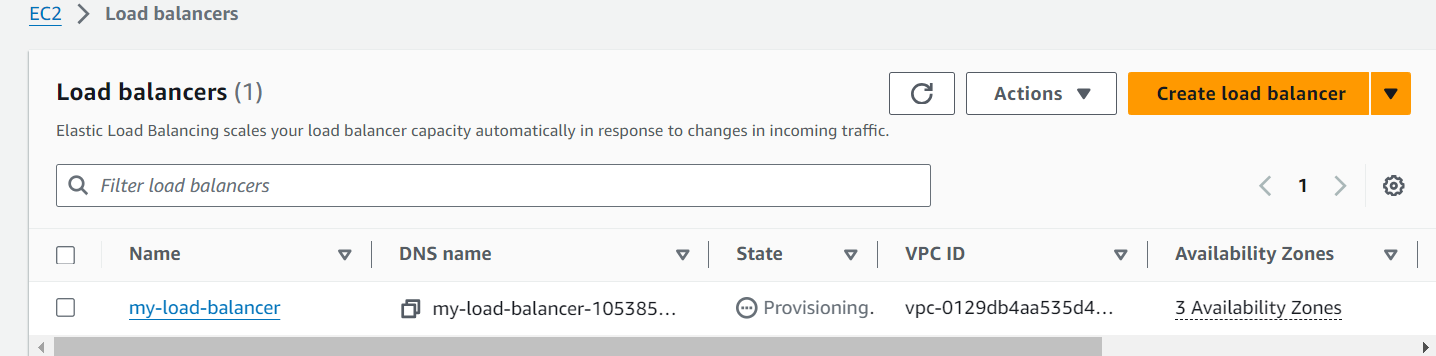




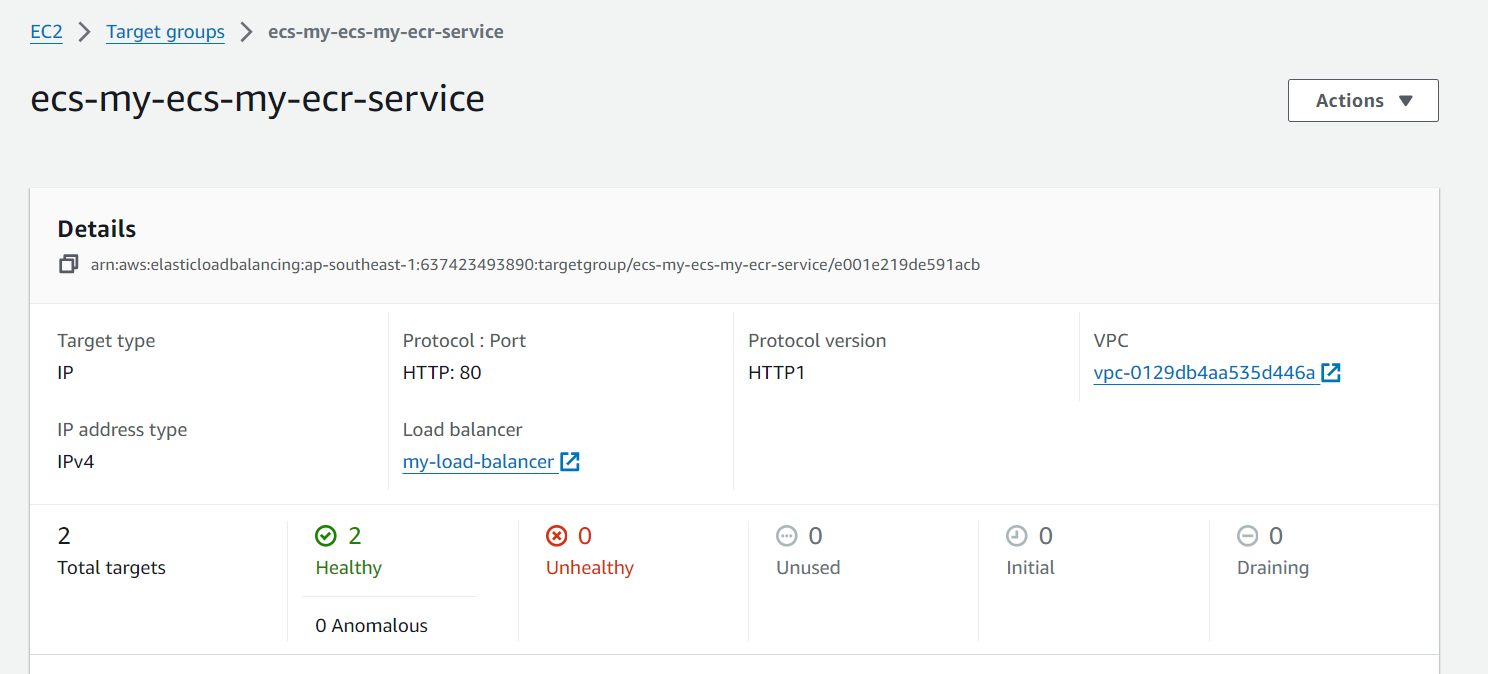


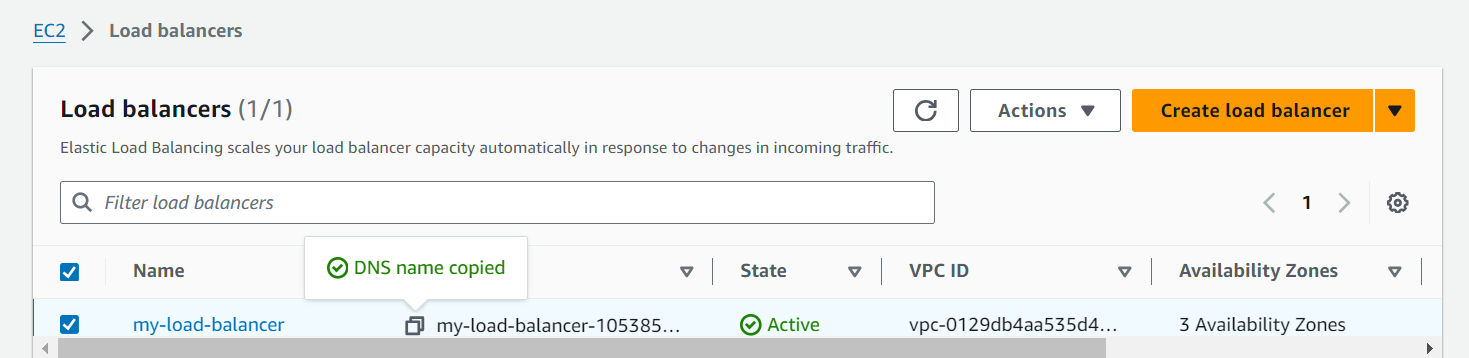


**Load balancer will be created**



**Load Balancer > Targets > check if the load-balancer is healthy or not**





### **8. Test and Monitor**

* **Access the Application**:
  + Once the service is up, you can access the application using the DNS name of the load balancer.
* **Monitor ECS**:
  + Use the ECS dashboard to monitor the health of tasks, scaling, and other metrics.



