😊 AMAZON ECS (ELASTIC CONTAINER SERVICE)

Amazon Elastic Container Service (Amazon ECS) is a fully managed container orchestration service provided by Amazon Web Services (AWS). It allows you to easily run, stop, and manage Docker containers on a cluster. Containers are a lightweight and portable way to package, distribute, and run applications.

Key features of Amazon ECS include:

1. **Container Orchestration:** ECS simplifies the process of deploying, managing, and scaling containerized applications. It helps you orchestrate the deployment and scaling of Docker containers across a cluster of Amazon EC2 instances.
2. **Scalability:** ECS enables you to scale your application horizontally by adding or removing containers based on demand. It integrates with other AWS services like Auto Scaling, allowing you to automatically adjust the number of container instances based on defined criteria.
3. **Integration with AWS Services:** Amazon ECS seamlessly integrates with other AWS services, such as Amazon Elastic Load Balancing (ELB) for load balancing, Amazon VPC for networking, and AWS Identity and Access Management (IAM) for access control.
4. **Task Definitions:** ECS uses task definitions to specify the configuration details for a set of containers. This includes information such as which Docker image to use, container resource requirements, and data volumes to mount.
5. **Clusters:** ECS organizes container instances into logical groups called clusters. A cluster is a pool of resources that can be used to run containers. You can have multiple clusters in an AWS account.
6. **Service Definitions:** ECS allows you to define and run long-running applications using services. A service allows you to define the desired number of tasks and how they should be distributed across your cluster.
7. **Fargate:** In addition to running ECS on EC2 instances, AWS offers a service called AWS Fargate that allows you to run containers without managing the underlying infrastructure. With Fargate, you only pay for the vCPU and memory that you use, making it a serverless option for running containers.

Amazon ECS is suitable for a variety of applications, including microservices architectures, monolithic applications, and batch processing workloads. It provides a flexible and scalable platform for containerized applications in the AWS cloud environment.

🤓 USE CASES OF AMAZON ECS:

Amazon ECS (Elastic Container Service) is a versatile container orchestration service that can be used for various use cases across different industries. Here are some common use cases for Amazon ECS:

1. **Microservices Architecture:** ECS is well-suited for deploying and managing microservices-based applications. It allows you to break down your application into

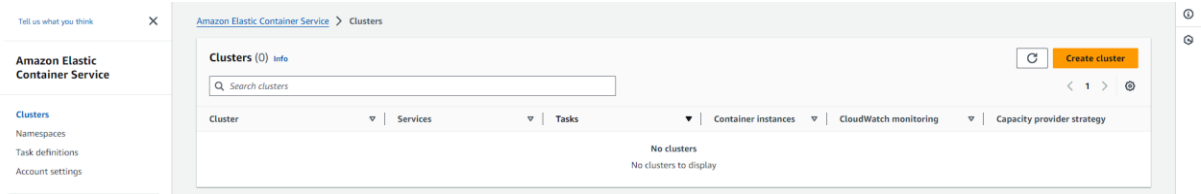
smaller, independent services that can be developed, deployed, and scaled independently.

2. **Web Applications:** Deploying and scaling web applications is a common use case for ECS. By defining services and tasks in ECS, you can easily manage the deployment of web servers, load balancers, and other components of your application.
3. **Batch Processing:** ECS can be used for running batch processing workloads where you need to process large amounts of data periodically. You can define ECS tasks to execute batch jobs, and ECS can scale the number of tasks based on demand.
4. **Continuous Integration/Continuous Deployment (CI/CD):** Integrate ECS into your CI/CD pipeline to automate the building, testing, and deployment of containerized applications. ECS can be used to manage the deployment of new versions of your application, making it easy to achieve continuous delivery.
5. **Machine Learning Inference:** For deploying and scaling machine learning models in production, ECS can be a suitable choice. You can containerize your machine learning models and use ECS to manage the deployment and scaling of inference services.
6. **DevOps Workflows:** ECS can be integrated into DevOps workflows to streamline the process of deploying and managing applications. It facilitates the automation of tasks related to containerized application deployment and updates.
7. **Hybrid Architectures:** For organizations with hybrid cloud architectures, ECS can be used to manage containerized applications both on-premises and in the cloud. This flexibility allows for a seamless integration of container workloads across different environments.
8. **Stateless Applications:** ECS is particularly well-suited for stateless applications where each instance of the application can handle requests independently. Statelessness makes it easier to scale applications horizontally by adding or removing containers.
9. **Multi-Region Deployments:** If your application needs to be deployed across multiple AWS regions for high availability and disaster recovery, ECS can help manage the deployment and scaling of containers in each region.
10. **Cost-Effective Scaling:** ECS, especially when combined with AWS Fargate, provides a cost-effective solution for running containers. With Fargate, you don't need to manage the underlying infrastructure, and you only pay for the resources your containers consume.

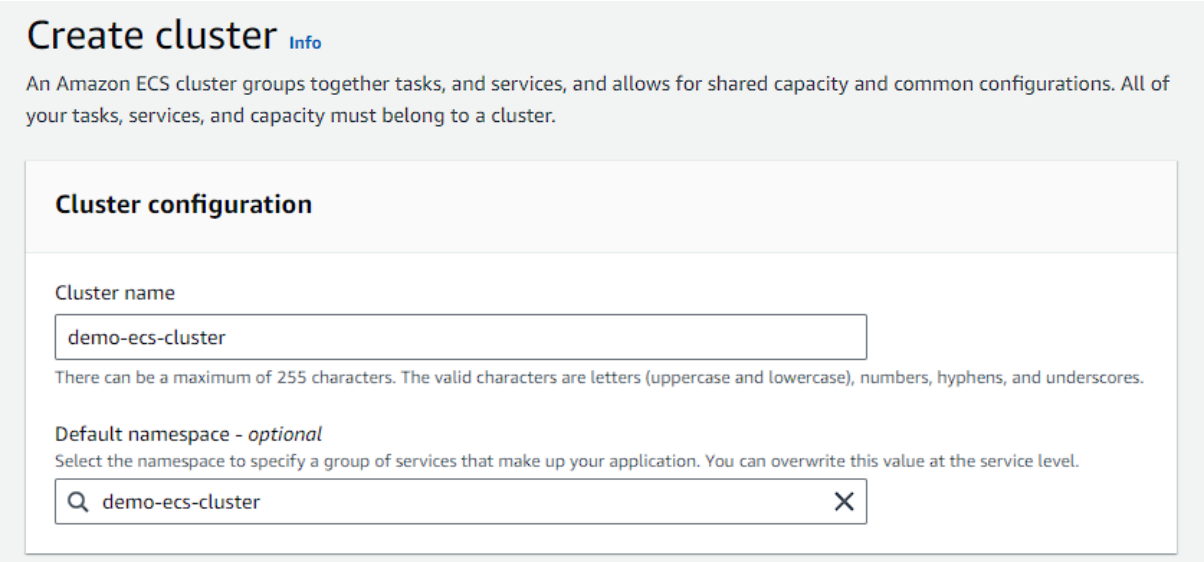
These use cases highlight the flexibility of Amazon ECS, making it suitable for a wide range of applications and scenarios in the cloud environment.

😁 TO BEGIN WITH THE LAB:

1. Log in to Amazon Console. There you need to search ECS (Elastic Container Service)

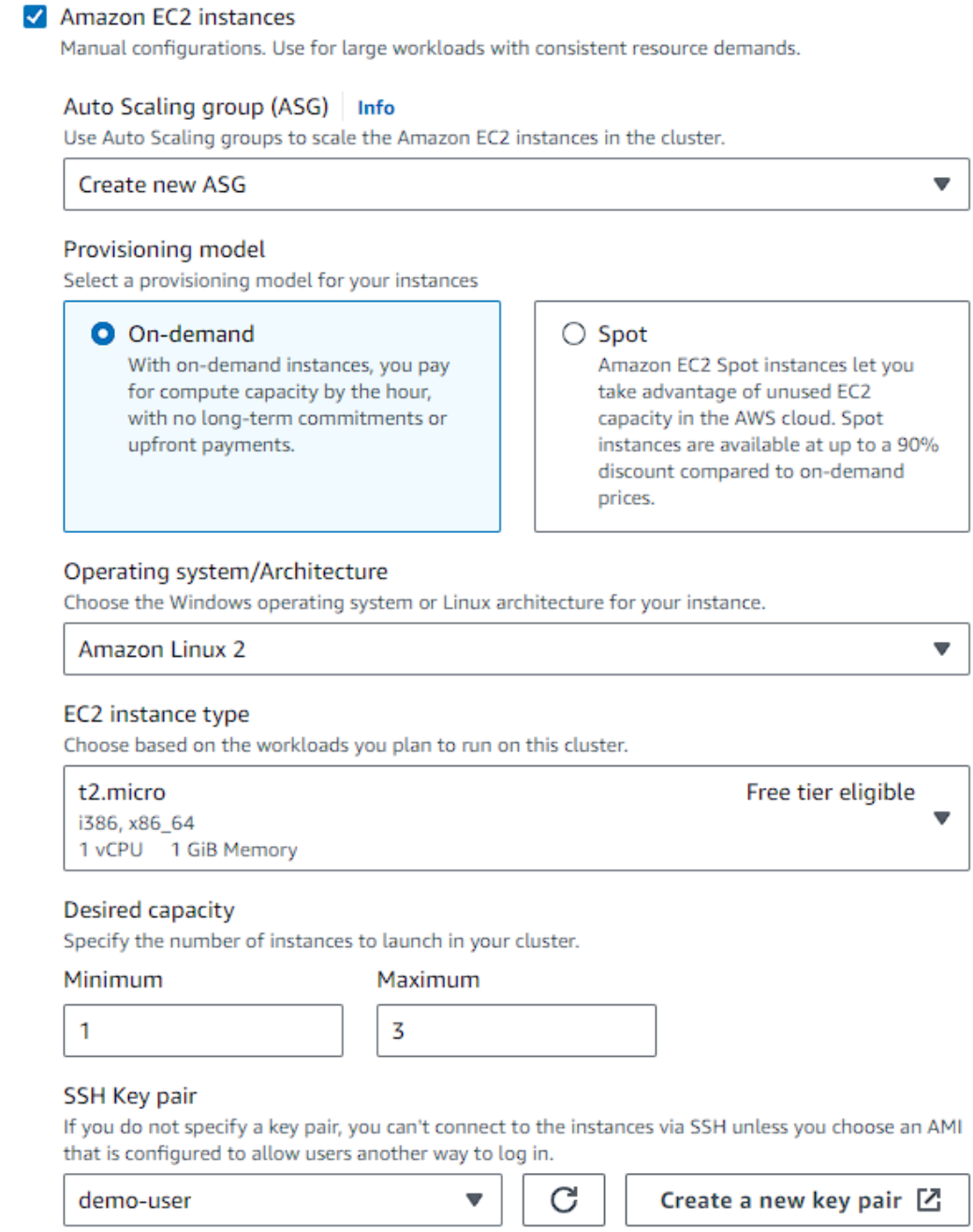


2. Now you need to create a cluster. Click on create cluster.
3. Give it a name, then move forward.

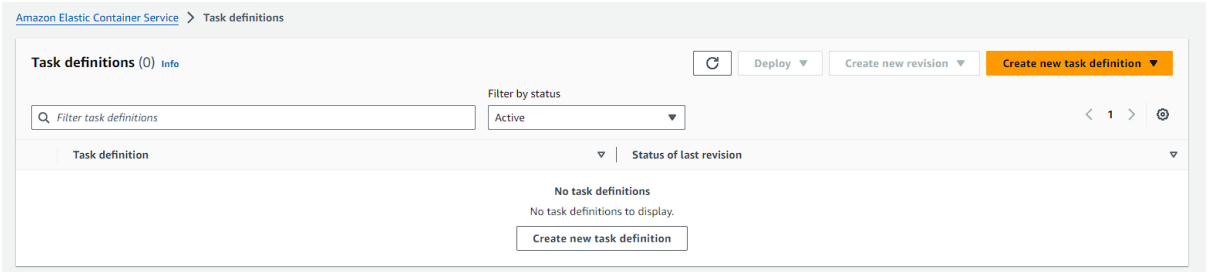


4. In the infrastructure part, AWS Fargate will be selected by default so uncheck Fargate. You need to select Amazon EC2 instances only.

5. As soon as you click on ec2, it will ask you to create an ASG (Auto Scaling Group).
6. Now you need to select On-demand for provisioned model.
7. Then for the operating system select Amazon Linux 2.
8. For the instance type select t2.micro
9. Then in the desired capacity select minimum to be 1 and maximum to be 3.
10. In the last select your key pair. If you don't have a key pair no issues just create new.

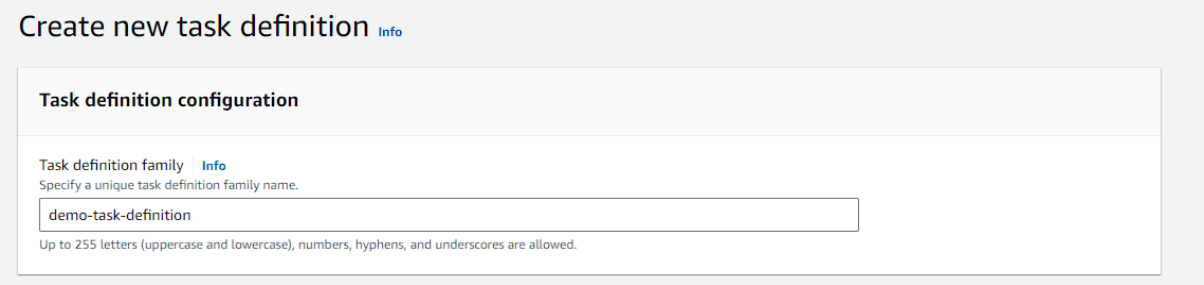


11. So, when you will scroll down you will see the network configuration leave them to default. And create your cluster.
12. Till your cluster is creating, switch to task definition and create it.

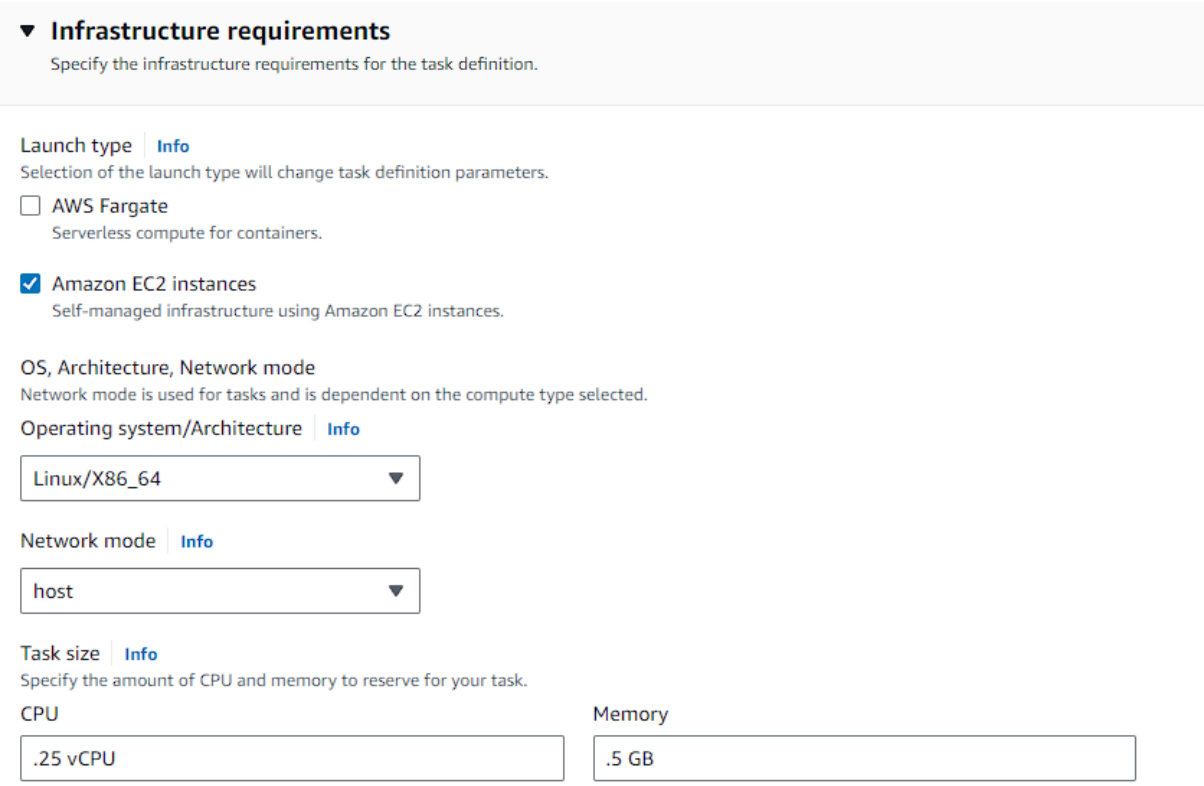


13. Click on create new task definition.

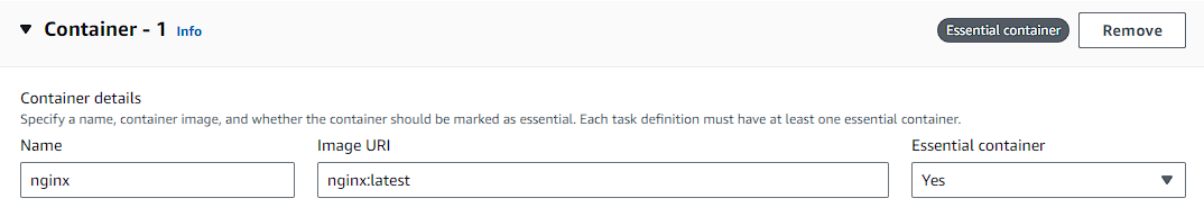
14. In there now you need to give it a name.



15. Then in infrastructure requirements, select launching type as ec2 instances.
16. The OS should be Linux.
17. Then the select host in the network mode.
18. Now in the task size you need to select CPU as 0.25 and memory as 0.5gb, so that it is compatible with our instance type that is t2.micro.

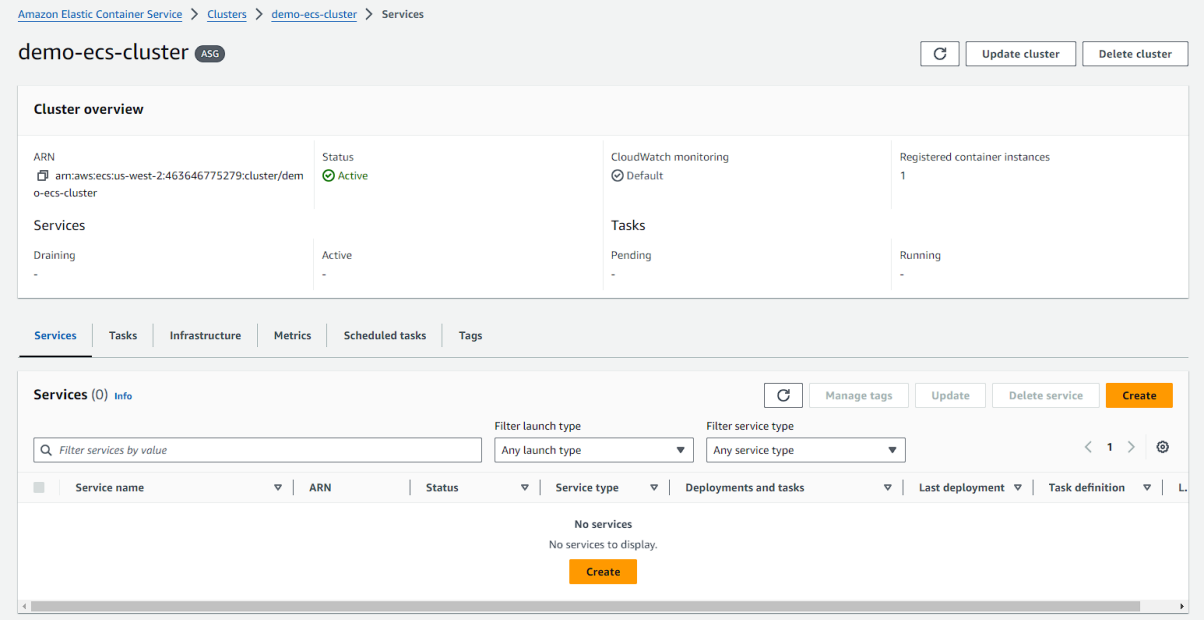


19. And within here you'll have to specify the details related to what are the container images that you'd like. You'll be using a sample nginx container image.

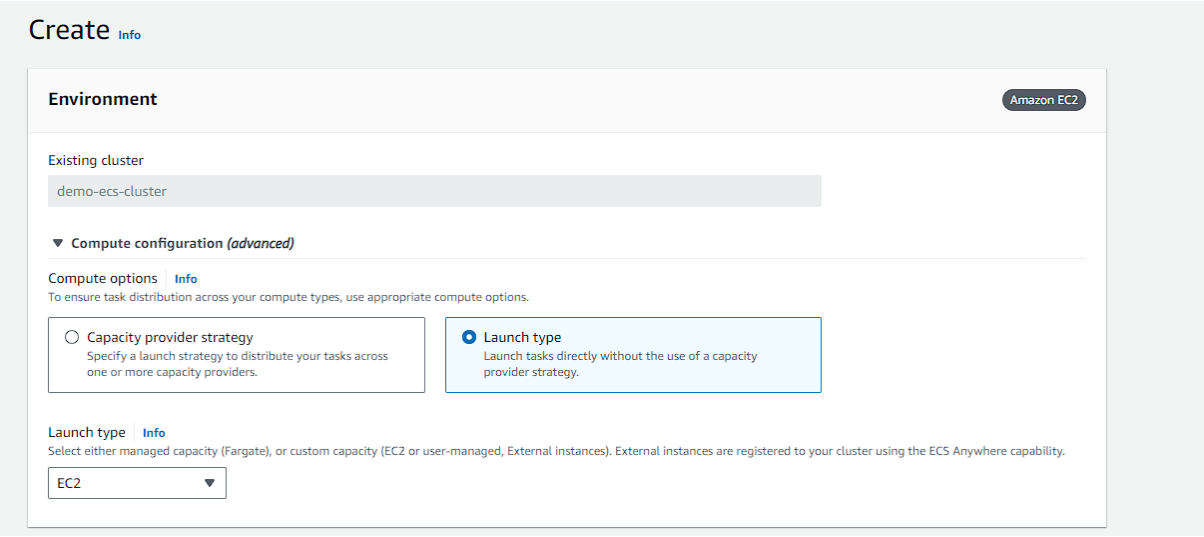


20. So, now just scroll down to the bottom to create your task definition.

21. If you go back to your cluster and open it, you'll see that it is empty. There is no service running.
22. So, quickly click on create, to create a service.



23. In here you need to select compute options as Launch type.
24. Then select Launch type as EC2.



25. In the deployment configuration select Service as application type.
26. In the family select your task definition that you created.
27. Give a service name to it.
28. In the desired task select it to 1.
29. Then keep everything to default and click on create.

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: demo-task-definition

Revision: 1 (LATEST)

Service name

Assign a unique name for this service.

nginx-service

Service type | Info

Specify the service type that the service scheduler will follow.

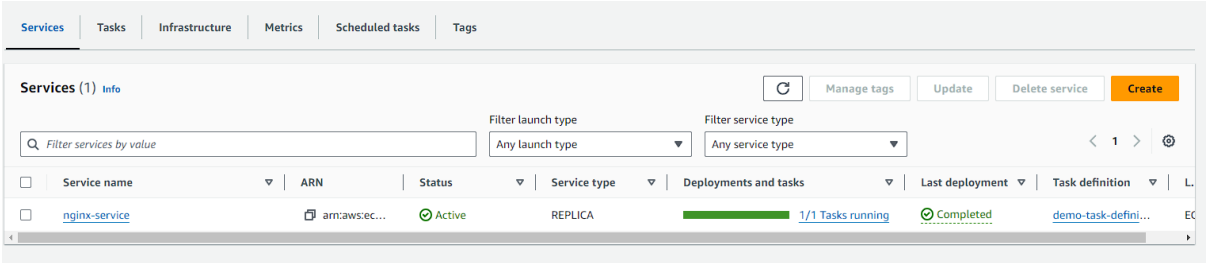
- (•) Replica — Place and maintain a desired number of tasks across your cluster.
- ( ) Daemon — Place and maintain one copy of your task on each container instance.

Desired tasks

Specify the number of tasks to launch.

1

30. After some time, you'll see that your service is in active state.



31. Now if go to your ec2 instance, there you have to copy your public IP address and paste it in the new tab or browser.