

Day-48 – Deploy App Using ECS And ECR

Today will be a great learning for sure. I know many of you may not know about the term "ECS". As you know, 90 Days of DevOps Challenge is mostly about 'learning new', let's learn then ;)

What is Amazon ECS?

Elastic Container Service (ECS) is a container management service that is highly scalable and fast. Managing containers and performing operations like start/stop is very easy on ECS. Containers in ECS are defined in a task definition within a service and service is a configuration that runs and maintains a specified number of tasks in a cluster. Tasks can be run on a serverless infrastructure that is managed by AWS Fargate or on a cluster of Amazon EC2 instances that is managed by the user.

What is Amazon EKS?

Elastic Kubernetes Service, EKS, is a managed service that can be used to run Kubernetes on AWS. There is no need to install, operate, and maintain the Kubernetes control plane or nodes while using EKS. To ensure high availability, EKS runs Kubernetes control plane instances across multiple Availability Zones. When the nodes are unhealthy, EKS automatically replaces them. EKS provides scalability and security to the applications.

What is meant by Orchestration?

Orchestration is the automated configuration, management, and coordination of computer systems, applications, and services. Orchestration helps IT to more easily manage complex tasks and workflows.

Difference between EKS and ECS?

1] Networking: - EKS allows up to 750 pods per instance whereas ECS accommodates only up to a maximum of 120 tasks per instance.

2] Namespaces: - EKS has the concept of namespaces which isolates workloads running in the same cluster, whereas ECS does not have such concept in it. NameSpaces provide a lot of Advantages.

3] Ease of Use: - ECS is very straight forward and does not have a lot of components to learn whereas EKS is more complex as it uses Kubernetes which is altogether a vast technology to learn and requires expertise for deployments.

4] Pricing and costs: - Amazon EKS consists of a control plane and infrastructure, AWS Fargate or AWS EC2 to host containerized applications. The Pricing of EKS or ECS is dependent of infrastructure (EKS or ECS) being used to host containerized applications.

5] portability or compatibility: - Both EKS or ECS are managed services of AWS. ECS is an AWS proprietary service whereas EKS is a Kubernetes as-a-platform service by AWS. All the applications running on EKS clusters can be run on any other Kubernetes cluster with a little or no change, whereas deploying applications on ECS means using the proprietary container platform of AWS.

Task:

Set up ECS (Elastic Container Service) by setting up Nginx on ECS.

Here for Demo purpose, we use base OS I.e., Linux **Ubuntu**.

Create a new EC2 Instance.

EC2 > Instances > Launch an Instance

Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags [Info](#)

Name

[Add additional tags](#)

▼ Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Recently

Quick Start

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

Red Hat

[Browse more AMIs](#)
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type

Free tier eligible

ami-0557a15b87f6559cf (64-bit (x86)) / ami-0f9bd9098aca2d42b (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-02-08

Architecture

AMI ID

64-bit (x86)

ami-0557a15b87f6559cf

Verified provider

▼ Summary

Number of instances [Info](#)

1

Software Image (AMI)

Canonical, Ubuntu, 22.04 LTS, ...[read more](#)

ami-0557a15b87f6559cf

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

Free tier: in your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million I/Os, 1 GiB of snapshots, and 100 GB of bandwidth to the internet.

×

Cancel

Launch instance

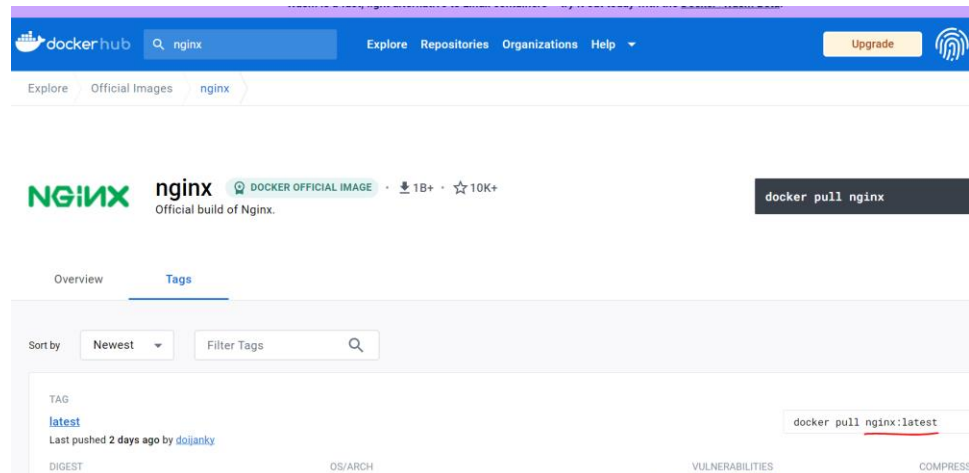
Here we use Containerization tool I.e., **Docker** to make a container for an App.

```

reinstalling package 1.5.9-0ubuntu3.1 done
ubuntu@ip-172-31-61-108:~$ sudo apt-get update && sudo apt-get install docker.io
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 38 not upgraded.
Need to get 66.8 MB of archives.
After this operation, 287 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 bridge-utils amd64 1.7-1ubuntu3 [34.4 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 runc amd64 1.1.0-0ubuntu1.1 [4242 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 containerd amd64 1.5.9-0ubuntu3.1 [28.1 MB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 dns-root-data all 2021011101 [5256 B]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dnsmasq-base amd64 2.86-1.1ubuntu0.1 [354 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 docker.io amd64 20.10.12-0ubuntu4 [34.0 MB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 ubuntu-fan all 0.12.16 [35.2 kB]
Fetched 66.8 MB in 2s (34.2 MB/s)
Preconfiguring packages ...
Selecting previously unselected package pigz.
(Reading database ... 63605 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.6-1_amd64.deb ...
Unpacking pigz (2.6-1) ...
Selecting previously unselected package bridge-utils.
Preparing to unpack .../1-bridge-utils_1.7-1ubuntu3_amd64.deb ...
Unpacking bridge-utils (1.7-1ubuntu3) ...
Selecting previously unselected package runc.
Preparing to unpack .../2-runc_1.1.0-0ubuntu1.1_amd64.deb ...
Unpacking runc (1.1.0-0ubuntu1.1) ...
Selecting previously unselected package containerd.
Preparing to unpack .../3-containerd_1.5.9-0ubuntu3.1_amd64.deb ...
Unpacking containerd (1.5.9-0ubuntu3.1) ...
Selecting previously unselected package dns-root-data.
Preparing to unpack .../4-dns-root-data_2021011101_all.deb ...
Unpacking dns-root-data (2021011101) ...
Selecting previously unselected package dnsmasq-base.
Preparing to unpack .../5-dnsmasq-base_2.86-1.1ubuntu0.1_amd64.deb ...
Unpacking dnsmasq-base (2.86-1.1ubuntu0.1) ...
Selecting previously unselected package docker.io.

```

Copy the latest Nginx image from Docker Hub.



The screenshot shows the Docker Hub interface for the nginx image. The top navigation bar includes the Docker Hub logo, a search bar with 'nginx' entered, and links for 'Explore', 'Repositories', 'Organizations', and 'Help'. Below the navigation bar, the 'nginx' repository is displayed, featuring the NGINX logo and the text 'Official build of Nginx.' A 'docker pull nginx' button is located in the top right corner. The 'Tags' tab is selected, showing a list of tags. The 'latest' tag is highlighted, with a note indicating it was last pushed 2 days ago by 'doijanky'. A search bar for tags is also present. At the bottom of the page, there are tabs for 'Overview', 'Tags', 'Digest', 'OS/ARCH', 'Vulnerabilities', and 'Compress'.

Make a DockerFile.

```

ubuntu@ip-172-31-61-108:~$ vi Dockerfile
ubuntu@ip-172-31-61-108:~$ cat Dockerfile
From nginx:latest
EXPOSE 80

```

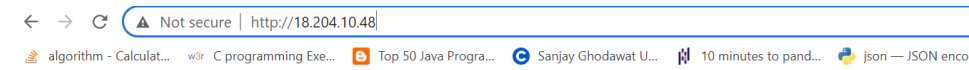
Build and Run Docker File and generate a Docker Image and Docker Container.

```

root@nginx-172-17-0-100:~# sudo docker build -t nginx-image
Sending build context to Docker daemon 11.82kB
Step 1/2 : FROM nginx:latest
latest: Pulling from library/nginx
1f9582a2c3a7: Pull complete
ba6c4f28b71d: Pull complete
d118d798bc21: Pull complete
73a6d4d31109: Pull complete
d95ba36e4ed1: Pull complete
ba18f20135d1: Pull complete
Digest: sha256:a6bf6e6b3c4e7589f421d932b37b73ed2372a75a65cd88d4753e32e72
Status: Downloaded newer image for nginx:latest
--> 98d8d313b93
Step 2/2 : EXPOSE 80
--> Running in f283bc48bd
Removing intermediate container f283bc48bd
--> ba193bcc3119
Successfully built ba193bcc3119
Successfully tagged nginx-image:latest
root@nginx-172-17-0-100:~# sudo docker run -d -p 80:80 --name nginx-ctr nginx-image:latest
c1e0ef9d313e2a86848770a7f725f505e4d31c8e13d8d754ec13263a1
root@nginx-172-17-0-100:~# docker ps
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://fd32-var92-run92f-docker.sock/v1.24/containers/json": dial unix /var/run/docker.sock: connect: permission denied
root@nginx-172-17-0-100:~# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED    STATUS    PORTS    NAMES
c1e0ef9d31     nginx:latest  "/docker-entrypoint..." 30 seconds ago    Up 9 seconds    0.0.0.0:80->80/tcp, :::80->80/tcp    nginx-ctr
root@nginx-172-17-0-100:~#

```

Copy the EC2 Public Address and Hit on Browser and First Check Your App is Running on Local Machine.



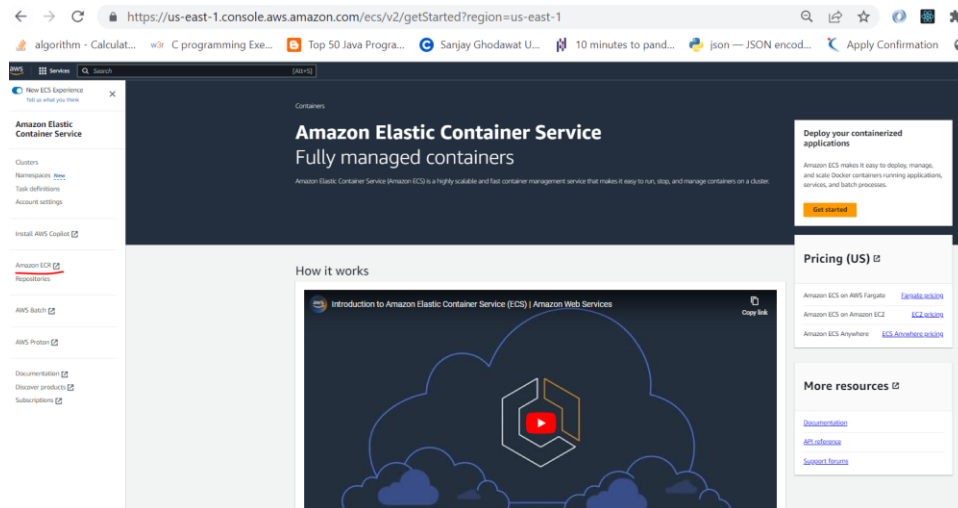
Welcome to nginx!

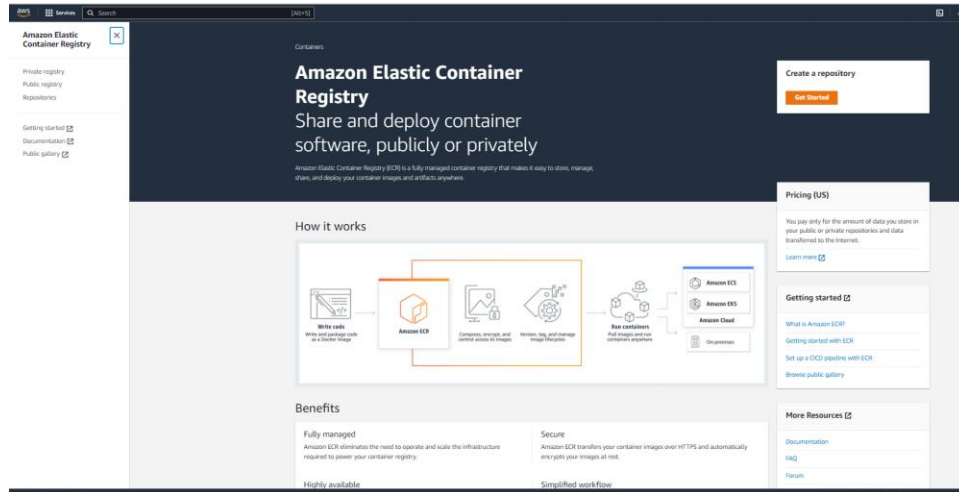
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

If Your App is Perfectly Run on Local, then you can Push This image to ECR (Elastic Container Registry)





First Make a Registry in ECR, Follow the Below Steps to Create a Registry.

Create repository

General settings

Visibility settings [Info](#)

Choose the visibility setting for the repository.

☒ **Private**
Access is managed by IAM and repository policy permissions.

☐ **Public**
Publicly visible and accessible for image pulls.

Repository name
Provide a concise name. A developer should be able to identify the repository contents by the name.

534497365850.dkr.ecr.us-east-1.amazonaws.com/ nginx-registry

14 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

Tag immutability [Info](#)

Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.

☒ **Enabled**

Once a repository is created, the visibility setting of the repository can't be changed.

Image scan settings

Deprecation warning
ScanOnPush configuration at the repository level is deprecated in favor of registry level scan filters.

Scan on push
Enable scan on push to have each image automatically scanned after being pushed to a repository. If disabled, each image scan must be manually started to get scan results.

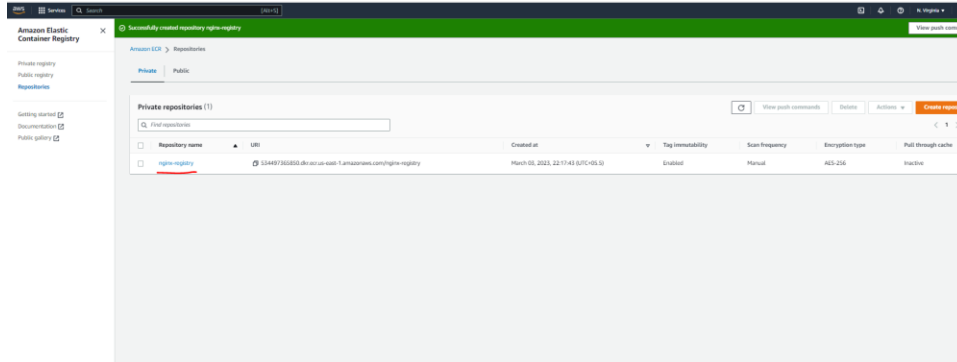
☒ **Disabled**

Encryption settings

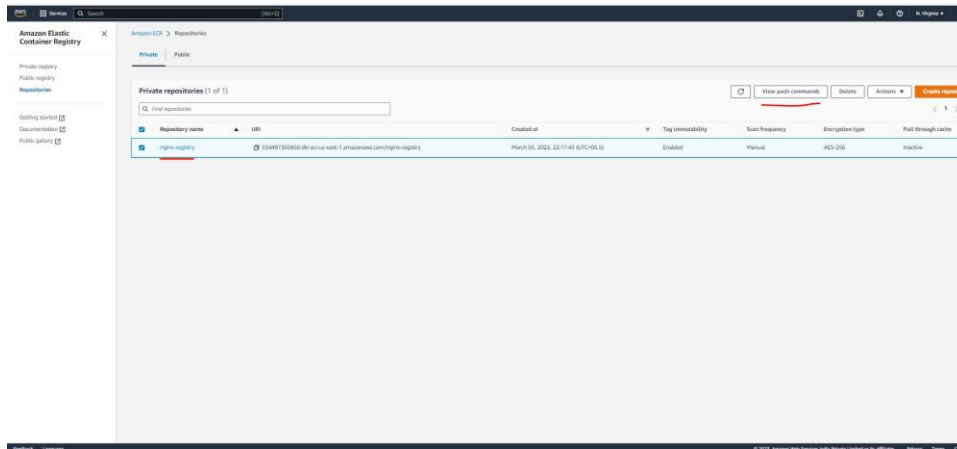
KMS encryption
You can use AWS Key Management Service (KMS) to encrypt images stored in this repository, instead of using the default encryption settings.

☒ **Disabled**

ECR Repository is Created Successfully.



Click on View Push Commands, To see all Push Commands.



Go to **Ubuntu cmd prompt** and install **awscli**, for communicating to ECR.

sudo apt install awscli -y ---> Only for Ubuntu OS

```
ubuntu@ip-172-31-61-108:~$ aws --version
aws-cli/2.11.0 Python/3.11.2 Linux/5.15.0-1028-aws exe/x86_64.ubuntu.22 prompt/off
ubuntu@ip-172-31-61-108:~$
```

ECR Push Commands

Push commands for docker-nginx-repo

macOS / LinuxWindows

Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see [Getting Started with Amazon ECR](#).

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see [Registry Authentication](#).

- Retrieve an authentication token and authenticate your Docker client to your registry.
Use the AWS CLI:

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 534497365850.dkr.ecr.us-east-1.amazonaws.com
```

Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.
- Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions [here](#). You can skip this step if your image is already built:

```
docker build -t docker-nginx-repo .
```
- After the build completes, tag your image so you can push the image to this repository:

```
docker tag docker-nginx-repo:latest 534497365850.dkr.ecr.us-east-1.amazonaws.com/docker-nginx-repo:latest
```
- Run the following command to push this image to your newly created AWS repository:

```
docker push 534497365850.dkr.ecr.us-east-1.amazonaws.com/docker-nginx-repo:latest
```

Close

Login in to ECR using aws cli

```
ubuntu@ip-172-31-14-114:~$ aws --version
aws-cli/1.22.34 Python/3.10.6 Linux/5.15.0-1028-aws boto3core/1.23.34
ubuntu@ip-172-31-14-114:~$ aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 534497365850.dkr.ecr.us-east-1.amazonaws.com
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post "http://82fvarh2f2runh2f2docker.sock/v1.24/auth": dial unix /var/run/docker.sock: connect: permission denied
ubuntu@ip-172-31-14-114:~$ sudo su
root@ip-172-31-14-114:/home/ubuntu# aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 534497365850.dkr.ecr.us-east-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@ip-172-31-14-114:/home/ubuntu#
```

Push commands for docker-nginx-repo

macOS / Linux

Windows

Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see [Getting Started with Amazon ECR](#).

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see [Registry Authentication](#).

- Retrieve an authentication token and authenticate your Docker client to your registry.
Use the AWS CLI:

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 534497365850.dkr.ecr.us-east-1.amazonaws.com
```

Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.
- Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions [here](#). You can skip this step if your image is already built:

```
docker build -t docker-nginx-repo .
```
- After the build completes, tag your image so you can push the image to this repository:

```
docker tag docker-nginx-repo:latest 534497365850.dkr.ecr.us-east-1.amazonaws.com/docker-nginx-repo:latest
```
- Run the following command to push this image to your newly created AWS repository:

```
docker push 534497365850.dkr.ecr.us-east-1.amazonaws.com/docker-nginx-repo:latest
```

Close

After ECR Login, try to tag an image name.

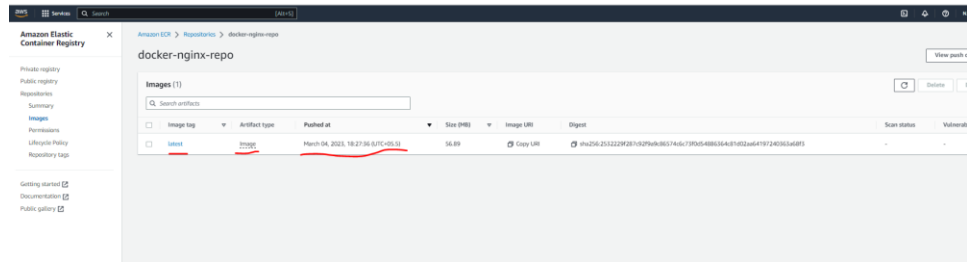
Syntax is: -

Docker tag <image_name><Above image mentioned Full path>

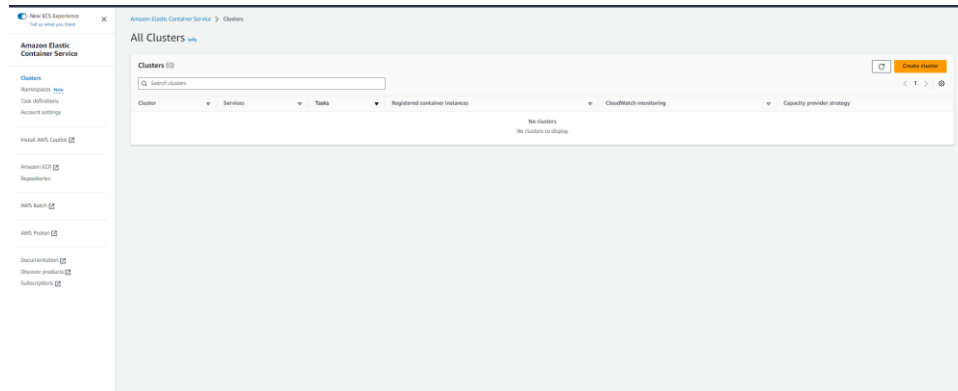
After that Using Push Command, Push the Docker tagged image to ECR Repository.

```
root@ip-172-31-14-114:/home/ubuntu# sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
nginx-image   latest    b46d13006aa2   15 minutes ago 142MB
nginx         latest    904b8cb13b93   2 days ago    142MB
root@ip-172-31-14-114:/home/ubuntu# sudo docker tag nginx-image:latest 534497365850.dkr.ecr.us-east-1.amazonaws.com/docker-nginx-repo:latest
root@ip-172-31-14-114:/home/ubuntu# sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
nginx-image    534497365850.dkr.ecr.us-east-1.amazonaws.com/docker-nginx-repo   latest    b46d13006aa2   16 minutes ago 142MB
nginx         534497365850.dkr.ecr.us-east-1.amazonaws.com/docker-nginx-repo   latest    b46d13006aa2   16 minutes ago 142MB
nginx         nginx                    latest    904b8cb13b93   2 days ago    142MB
root@ip-172-31-14-114:/home/ubuntu# docker push 534497365850.dkr.ecr.us-east-1.amazonaws.com/docker-nginx-repo:latest
The push refers to repository [534497365850.dkr.ecr.us-east-1.amazonaws.com/docker-nginx-repo]
101af4ba983b: Pushed
d8466e142d87: Pushed
83ba6d8ffb8c: Pushed
e161c82b34d2: Pushed
4dc5cd799a08: Pushed
650abce4b096: Pushed
latest: digest: sha256:2532229f287c92f9a9c86574c6c73f8d54886364c81d02aa64197240363a68f3 size: 1570
root@ip-172-31-14-114:/home/ubuntu#
```

See Our Docker nginx image is Successfully pushed into ECR Repository.



Goto ECS (Elastic Container Service) and Follow the Below Steps to create a Cluster.



Put necessary information and click on next and create.

New ECS Experience

Tell us what you think

Amazon Elastic Container Service

Clusters

Namespaces New

Task definitions

Account settings

Install AWS Copilot

Amazon ECR

Repositories

AWS Batch

AWS Proton

Documentation

Discover products

Subscriptions

Amazon Elastic Container Service > Create cluster

Create cluster

Cluster configuration

Cluster name

nginxcluster

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

▼ Networking

By default tasks and services run in the default subnets for your default VPC. To use the non-default VPC, specify the VPC and subnets.

VPC

Use a VPC with public and private subnets. By default, VPCs are created for your AWS account. To create a new VPC, go to the VPC Console.

vpc-09231f74fee21c955

default

Subnets

Select the subnets where your tasks run. We recommend that you use three subnets for production.

Choose subnets

subnet-0ca6c35505f0dbc3a

us-east-1a

X

subnet-0633758dc97b52565

us-east-1a | RDS-Pvt-subnet-1

X

subnet-080f8a290b88a37aa

us-east-1c | RDS-Pvt-subnet-3

X

subnet-0d45765f4e6806bb4

us-east-1f

X

subnet-0683fd96c823fee5

us-east-1d

X

subnet-0cbafab997b0df475

us-east-1a | RDS-Pvt-subnet-2

X

subnet-0b0999bab47a5589d

us-east-1b

X

subnet-02253e08a14bc8e56

us-east-1e

X

subnet-0690ec282ae44b7be

us-east-1c

X

subnet-05d168156fae31183

us-east-1b | RDS-Pvt-subnet-5

X

subnet-062358dec36165c64

us-east-1f | RDS-Pvt-subnet-4

X

Default namespace - optional

Select the namespace to specify a group of services that make up your application. You can overwrite this value at the service level.

nginxcluster

▼ Infrastructure

Serverless

AWS Fargate (serverless)

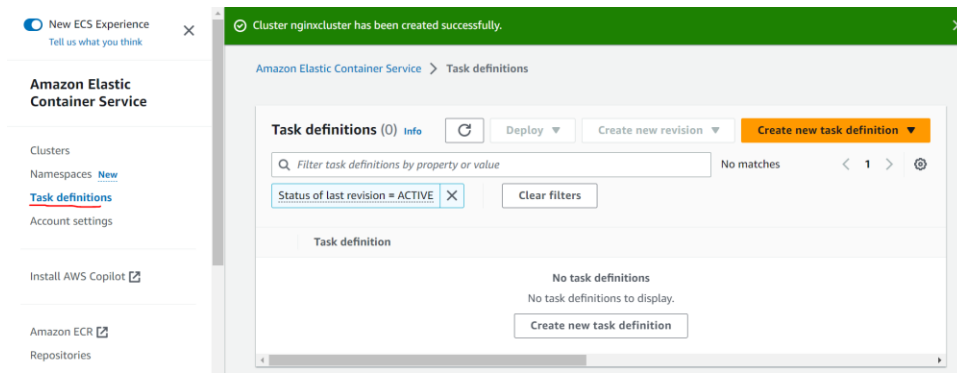
Pay as you go. Use if you have tiny, batch, or burst workloads or for zero maintenance overhead. The cluster has Fargate and Fargate Spot capacity providers by default.

Amazon EC2 instances

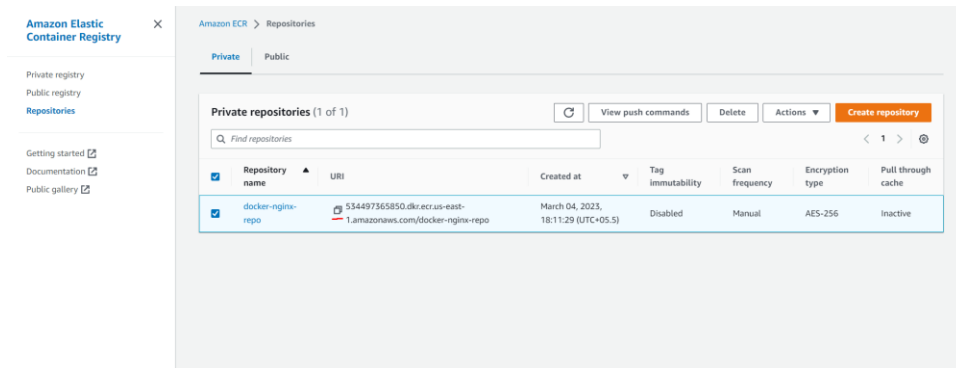
ECS Cluster is Created successfully.

After Creating cluster, you need to define a **TASK**.

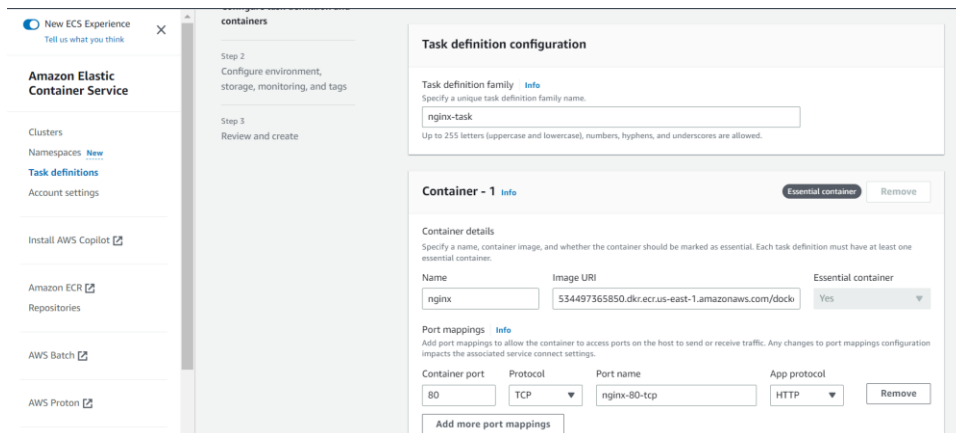
So, click on **Create a new task definition**.



Copy This ECR Image URI and paste in below **container Image URI** Section.



Give Task name and add Container port Name.



Select **Fargat** as an Environment Type.

New ECS Experience
Tell us what you think

Amazon Elastic Container Service

Clusters

Namespaces [New](#)

Task definitions

Account settings

Install AWS Copilot [↗](#)

Amazon ECR [↗](#)

Repositories

AWS Batch [↗](#)

AWS Proton [↗](#)

Documentation [↗](#)

Discover products [↗](#)

Amazon Elastic Container Service > Create new task definition

Step 1
Configure task definition and containers

Step 2
Configure environment, storage, monitoring, and tags

Step 3
Review and create

Configure environment, storage, monitoring, and tags

▼ Environment [Info](#)

Specify the infrastructure requirements for the task definition.

App environment [Info](#)
Specify the infrastructure for the task definition.

Add an option

AWS Fargate (serverless) [✕](#)

Operating system/Architecture [Info](#)

Linux/x86_64

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

1 vCPU

 Memory

3 GB

► Container size - optional [Info](#)

▼ Task roles, network mode - conditional [Info](#)

Task role [Info](#)
A task IAM role allows containers in the task to make API requests to AWS services. You can create a task IAM role from the IAM console [↗](#).

None

Ephemeral storage [Info](#)
The amount of ephemeral storage, in GiB, to allocate for the task. By default, your tasks hosted on AWS Fargate receive a maximum of 20 GiB of ephemeral storage.
Amount

21

To specify a custom amount of ephemeral storage, specify a value between 17 GiB up to a maximum of 200 GiB.

No storage associated with this task definition.

Container mount points [Info](#)

For each data volume associated with the task, add a container mount point to determine where the data volume is mounted.

No storage associated with this task definition. To configure storage at a container level, first add storage.

Add mount point

▼ Monitoring and logging - optional [Info](#)

Configure your container logging options and your application trace and metric collection settings using the AWS Distro for OpenTelemetry integrations.

CPU and memory allocation for a sidebar
There are monitoring and logging options that will automatically add a sidebar to your task definition if it does not already exist. AWS provides CPU and memory adjustment recommendations based on the selected options.

We recommend that you use log collection for tasks running on AWS Fargate. Learn more about log collection.

☒ Use log collection [Info](#)
Configure your task to send container logs to a logging destination using a default configuration. See pricing information on Amazon CloudWatch [↗](#).

Amazon CloudWatch

Key	Value type	Value
awslogs-group	<div>Value</div>	<div>/ecs/engine-task</div>
awslogs-region	<div>Value</div>	<div>us-east-1</div>
awslogs-stream-prefix	<div>Value</div>	<div>ecs</div>
awslogs-create-group	<div>Value</div>	<div>true</div>

Add

☐ Use trace collection [Info](#)
Amazon ECS creates an AWS Distro for OpenTelemetry sidebar to route traces from your application to AWS X-Ray. See pricing information on AWS X-Ray [↗](#).

☐ Use metric collection [Info](#) [Preview](#)
Amazon ECS creates an AWS Distro for OpenTelemetry sidebar to route custom container and application metrics to Amazon CloudWatch or Amazon Managed Service for Prometheus.

► Tags - optional [Info](#)

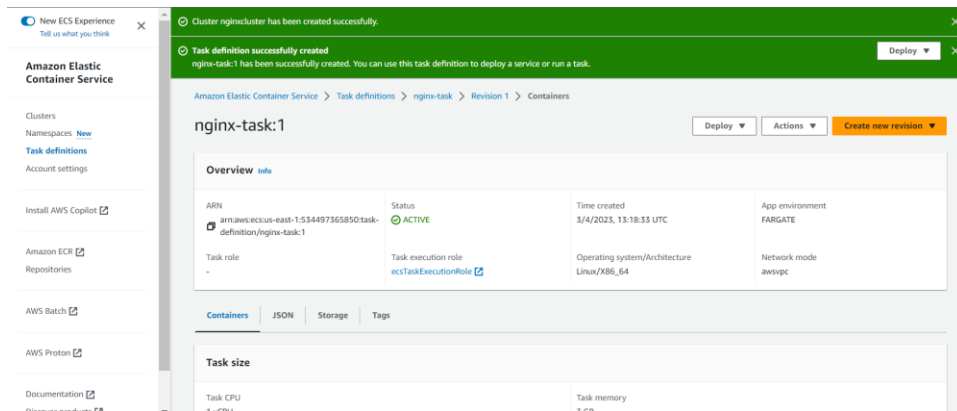
Tags help you to identify and organize your task definitions.

Cancel

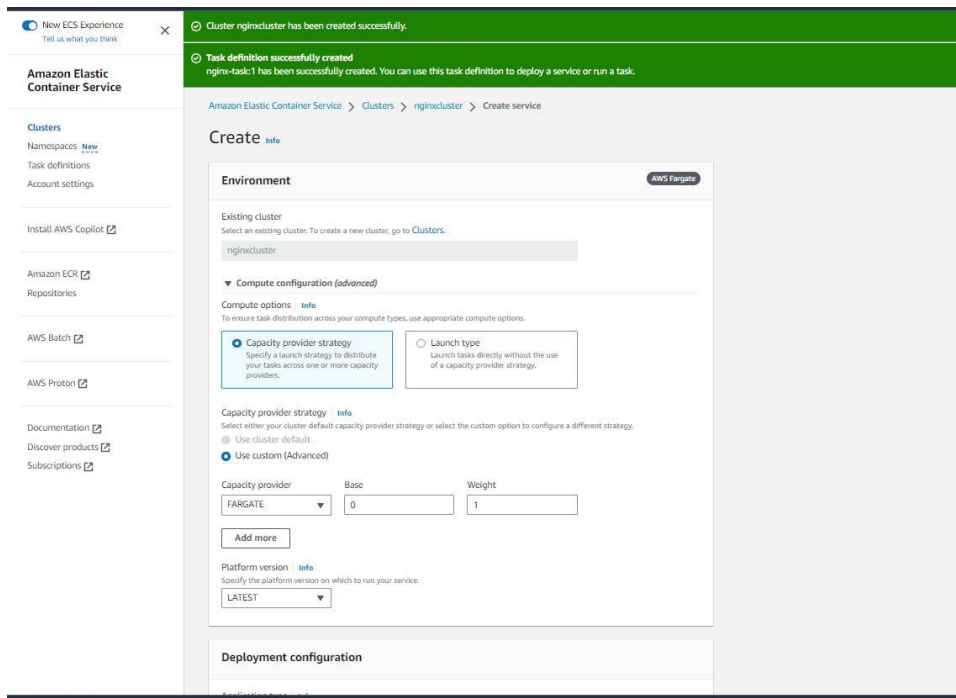
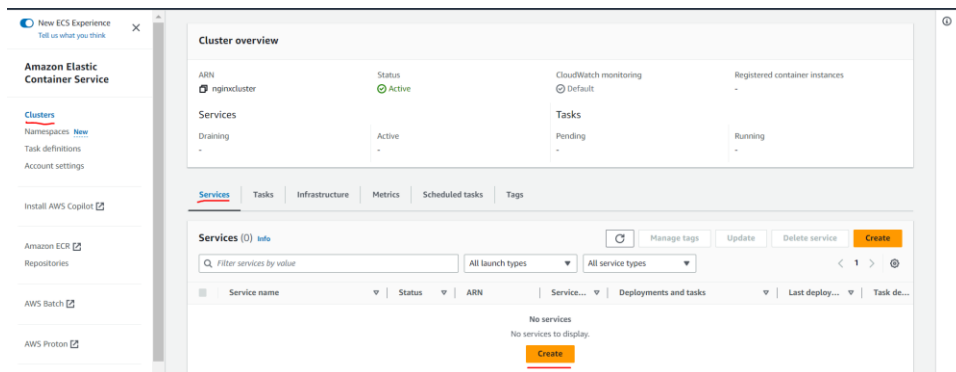
Previous

Next

Your Task Definition is Created Successfully.



After Creating Task Definitions, Create A service for Deployment of App.



Select Already Created Task Name and Give the Service name.

New ECS Experience

Tell us what you think

Amazon Elastic Container Service

Clusters

Namespaces [New](#)

Task definitions

Account settings

Install AWS Copilot [↗](#)

Amazon ECR [↗](#)

Repositories

AWS Batch [↗](#)

AWS Proton [↗](#)

Documentation [↗](#)

Discover products [↗](#)

Subscriptions [↗](#)

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

nginx-task

Revision

1 (LATEST)

Service name

Assign a unique name for this service.

nginx-service-2

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

► Deployment options

► Deployment failure detection [Info](#)

▼ Service Connect - optional

Configure this service in a namespace to create and resolve endpoints. Services can resolve endpoints within the same namespace without task or application configuration.

☐ Turn on Service Connect [Info](#)

Turn off Service Connect to remove the configuration.

Your Service is Created.

New ECS Experience

Tell us what you think

Amazon Elastic Container Service

Clusters

Namespaces [New](#)

Task definitions

Account settings

Install AWS Copilot [↗](#)

Amazon ECR [↗](#)

Repositories

AWS Batch [↗](#)

AWS Proton [↗](#)

Documentation [↗](#)

Discover products [↗](#)

Subscriptions [↗](#)

nginx-service-2 has been deployed successfully

Amazon Elastic Container Service > Clusters > nginxcluster > Services

nginxcluster

Update cluster Delete cluster

Cluster overview

ARN

nginxcluster

Status

Active

CloudWatch monitoring

Default

Registered container instances

0

Services

Draining

Active

Tasks

Pending

Running

0

Services (1) info

Filter services by value

All launch types

All service types

Message logs

Update

Delete service

Start

Service name	Status	ARN	Service type	Deployments and tasks	Last deployment	Task definition	Revision	Launch type
nginx-service-2	Active	arn:aws:ecs:us-east-1:123456789012:cluster/nginxcluster/service/nginx-service-2	REPLICA	1/1 Tasks running	Completed	nginx-task	1	-

Now, Your App is Deployed.

Check your app is Running, follow below path to copy the Public Address.

Goto Cluster, click on Tasks/Configuration

Inside this Configuration, you will find the Public Address.

Thank you for using nginx.