Deploy Django Application on AWS using ECS and ECR

Launch one instance in t2.medium.
Install aws cli and configure
#aws configure
access_key=”------------------------------”
secret_access_key=”--------------------”

Install docker for creating docker file
#sudo apt install docker.io

create docker group
#sudo groupadd docker

Add user to the docker group
#sudo usermod -aG docker [user-name]

To activate changes to the group
#newgrp docker

restart docker
#systemctl restart docker

Create a Docker File — Add the “Dockerfile” to the Django application.
#vim Dockerfile

OR -> clone the repository from git.
 #git clone (**git-url**)

Create Repository on AWS ECR – private > Repository name> create -click on created repo and using the below command on local

Retrieve an authentication token and authenticate your Docker client to your registry. Use the AWS CLI:
#aws ecr get-login-password --region us-east-2 | docker login --username AWS --password-stdin 690940206480.dkr.ecr.us-east-2.amazonaws.com

Build your Docker image using the following command

#docker build -t django-repo .

After the build completes, tag your image so you can push the image to this repository

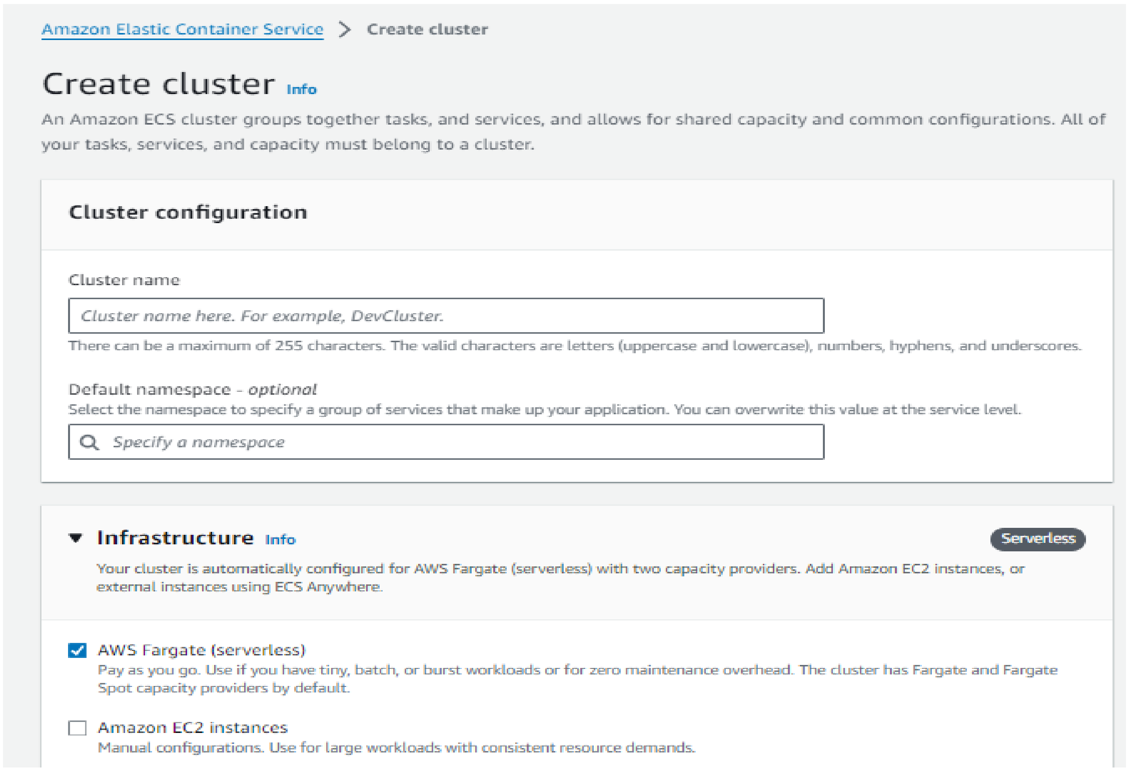
#docker tag django-repo:latest 690940206480.dkr.ecr.us-east-2.amazonaws.com/django-repo:latest

Run the following command to push this image to your newly created AWS repository

#docker push 690940206480.dkr.ecr.us-east-2.amazonaws.com/django-repo:latest

Go to ECS and create Cluster :

Cluster name - django-cluster -> select - AWS Fargate→create



Amazon Elastic Container Service > Create cluster

Create cluster Info

An Amazon ECS cluster groups together tasks, and services, and allows for shared capacity and common configurations. All of your tasks, services, and capacity must belong to a cluster.

Cluster configuration

Cluster name

Cluster name here. For example, DevCluster.

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

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.

Specify a namespace

▼ Infrastructure Info Serverless

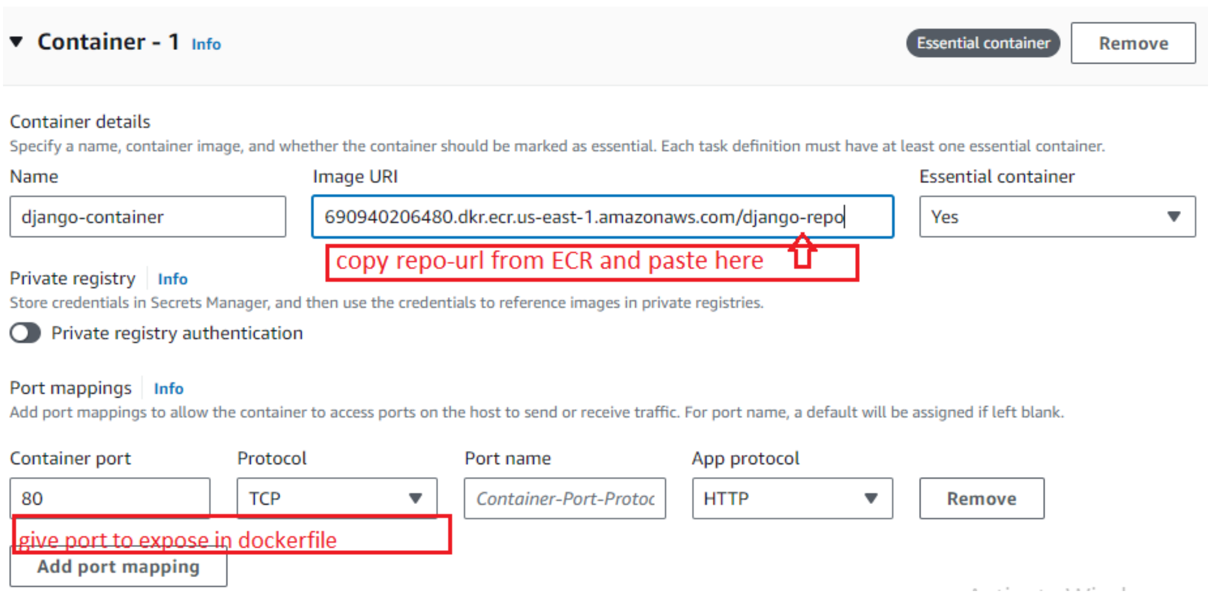
Your cluster is automatically configured for AWS Fargate (serverless) with two capacity providers. Add Amazon EC2 instances, or external instances using ECS Anywhere.

☑ 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
Manual configurations. Use for large workloads with consistent resource demands.

Create Task definations -
Create new task definition -
Task definition family - —----------------------------- > Select AWS Fargate
Task size
-Select CPU - 0.5 vCPU
-Select Memory - 1 GB

Container details -


▼ Container - 1 Info

Essential container | Remove

Container details

Specify a name, container image, and whether the container should be marked as essential. Each task definition must have at least one essential container.

Name	Image URI	Essential container
django-container	690940206480.dkr.ecr.us-east-1.amazonaws.com/django-repo	Yes

copy repo-url from ECR and paste here

Private registry | Info

Store credentials in Secrets Manager, and then use the credentials to reference images in private registries.

Private registry authentication

Port mappings | Info

Add port mappings to allow the container to access ports on the host to send or receive traffic. For port name, a default will be assigned if left blank.

Container port	Protocol	Port name	App protocol
80	TCP	Container-Port-Protoc	HTTP

give port to expose in dockerfile

After creating task definitions go to deploy and create service
Service name - django-service
Desired tasks - 2 **—> create**

After creating the service go to cluster open service
Go to task and check the status of task is running.
-> copy the public ip and heat on the browser.