Use case

Docker App Deployment to ECS Fargate

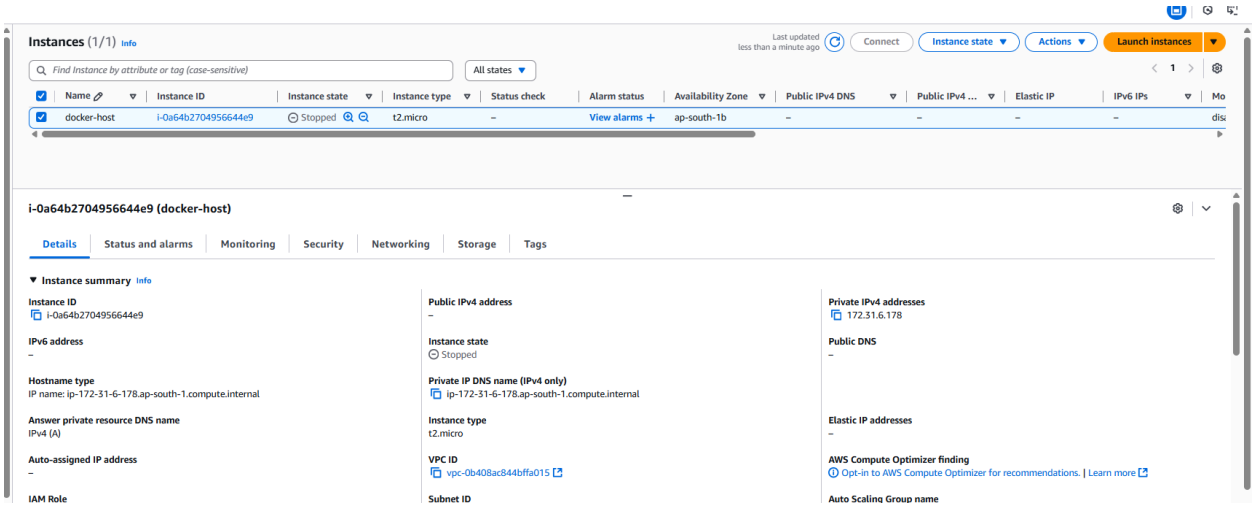
Use case Description

Dockerize a Flask app and push to Amazon ECR.

Deploy app on ECS Fargate with load balancing and auto-scaling.

Approach :

1. Create an EC2 instance
2. SSH into the instance and install docker
3. Create a Flask app
4. Connect to ECR and push the docker container
5. Deploy to Fargate in ECS with autoscaling



SSH & INSTALL DOCKER ON EC2 INSTANCE

```
PS C:\Users\dkarasu\desktop\AWSCloudPractitioner\UseCasesDocx\UC-20> ssh -i
.\MumbaiRegion.pem ec2-user@43.204.110.200
The authenticity of host '43.204.110.200 (43.204.110.200)' can't be established.
ED25519 key fingerprint is SHA256:lKKghvDJf/wSnFWhgaXq0n8BkApKrOPeQQbZ+kB9EL0.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '43.204.110.200' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
```

```
    ~~       V~' '->
     ~~~         /
       ~~._.   _/
          _/ _/
        _/m/'
[ec2-user@ip-172-31-6-178 ~]$ sudo yum update -y
Amazon Linux 2023 Kernel Livepatch repository                    161 kB/s |  17
kB     00:00
Last metadata expiration check: 0:00:01 ago on Wed Jul  9 11:35:02 2025.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-6-178 ~]$ sudo yum install docker -y
Last metadata expiration check: 0:00:12 ago on Wed Jul  9 11:35:02 2025.
Dependencies resolved.
================================================================================
================================================================
 Package                  Architecture       Version                  Repository
Size
================================================================================
================================================================
Installing:
 docker                   x86_64             25.0.8-1.amzn2023.0.4    amazonlinux
44 M
Installing dependencies:
 container-selinux        noarch             3:2.233.0-1.amzn2023     amazonlinux
```

```
[ec2-user@ip-172-31-6-178 ~]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[ec2-user@ip-172-31-6-178 ~]$ sudo usermod -aG docker ec2-user
[ec2-user@ip-172-31-6-178 ~]$ exit
logout
```

```
PS C:\Users\dkarasu\desktop\AWSCloudPractitioner\UseCasesDocx\UC-20> ssh -i
.\MumbaiRegion.pem ec2-user@43.204.110.200
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Wed Jul  9 11:34:38 2025 from 103.183.203.20
[ec2-user@ip-172-31-6-178 ~]$ docker --version
Docker version 25.0.8, build 0bab007
[ec2-user@ip-172-31-6-178 ~]$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:940c619fbd418f9b2b1b63e25d8861f9cc1b46e3fc8b018ccfe8b78f19b8cc4f
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

SETUP FLASK APP

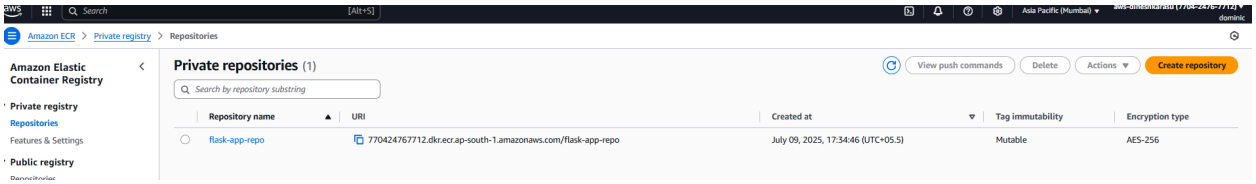
```
[ec2-user@ip-172-31-6-178 ~]$ mkdir flask-app
[ec2-user@ip-172-31-6-178 ~]$ cd flask-app
[ec2-user@ip-172-31-6-178 flask-app]$ nano app.py
[ec2-user@ip-172-31-6-178 flask-app]$ nano app.py
[ec2-user@ip-172-31-6-178 flask-app]$ echo "Flask==2.0.1" > requirements.txt
[ec2-user@ip-172-31-6-178 flask-app]$ nano Dockerfile
[ec2-user@ip-172-31-6-178 flask-app]$ docker build -t flask-app .
[+] Building 10.6s (10/10) FINISHED
```

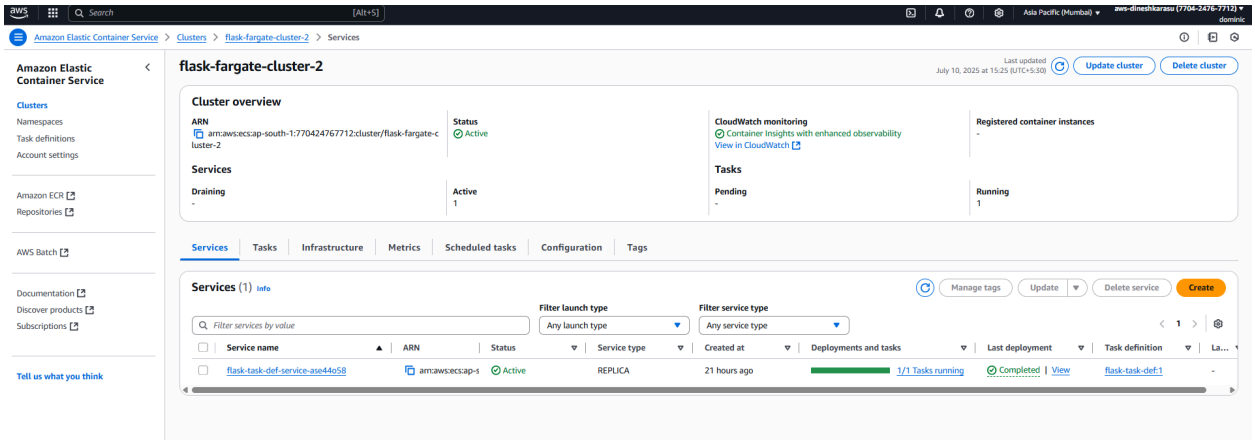
```python
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
    return "Hello from Docker on EC2!"

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000)
```

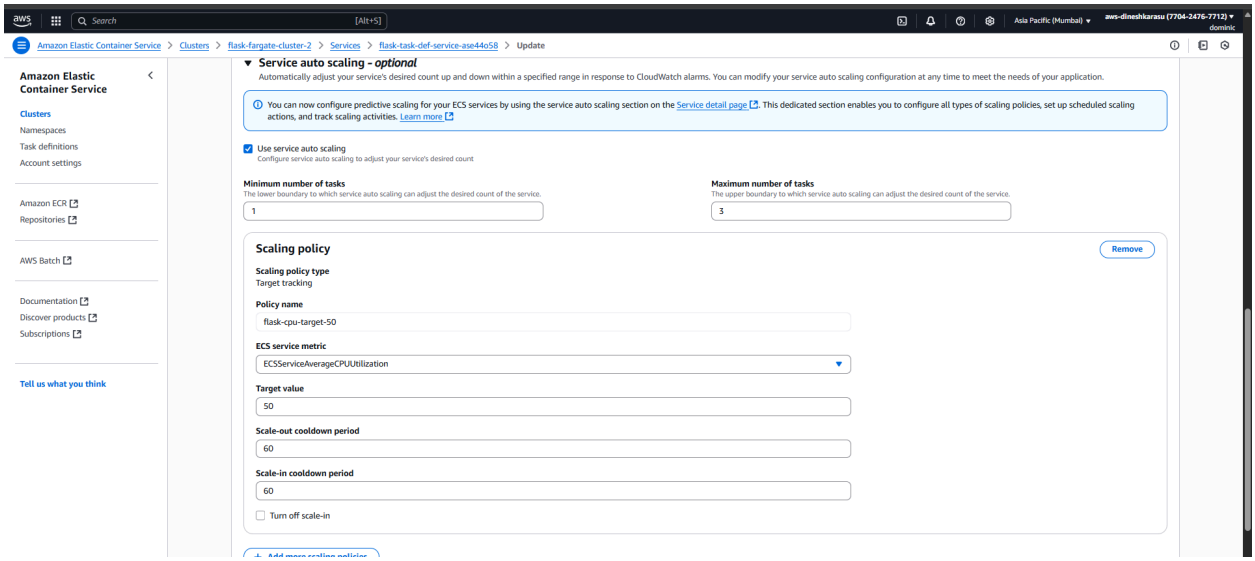
```
[ec2-user@ip-172-31-6-178 flask-app]$ docker run -d -p 5000:5000 flask-app
7185f9d72be24e4396125ed243c9eec51ba833c81285a22dd3642fc9dd30dec8
[ec2-user@ip-172-31-6-178 flask-app]$ docker ps
```

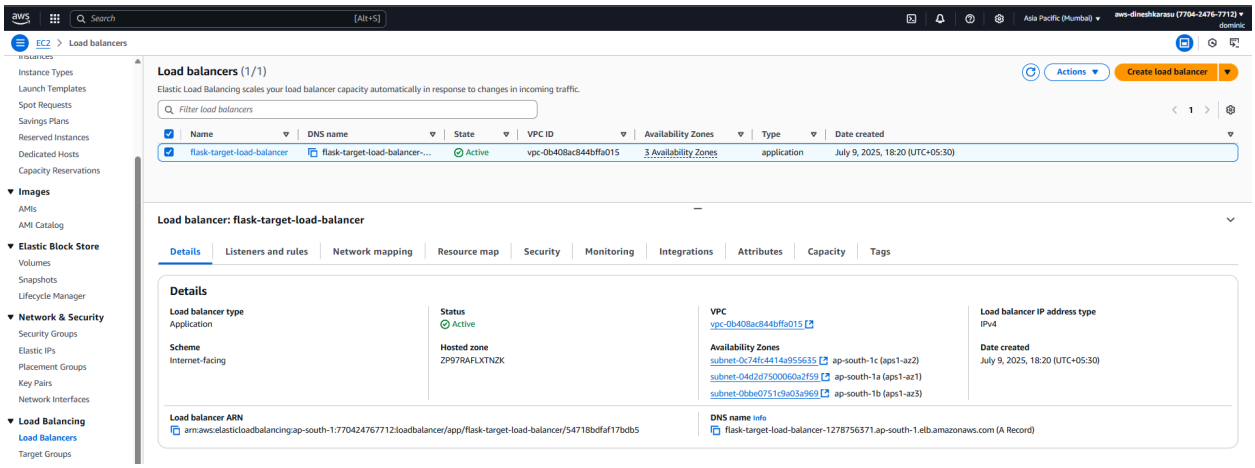
```
CONTAINER ID   IMAGE      COMMAND          CREATED          STATUS          PORTS
NAMES
7185f9d72be2   flask-app   "python app.py"   11 seconds ago   Up 10 seconds
0.0.0.0:5000->5000/tcp, :::5000->5000/tcp   elated_germain
[ec2-user@ip-172-31-6-178 flask-app]$ aws ecr create-repository --repository-name
flask-app-repo --region ap-south-1
```

```
[ec2-user@ip-172-31-6-178 flask-app]$ aws ecr create-repository --repository-name
flask-app-repo --region ap-south-1
{
    "repository": {
        "repositoryArn": "arn:aws:ecr:ap-south-1:770424767712:repository/flask-app-repo",
        "registryId": "770424767712",
        "repositoryName": "flask-app-repo",
        "repositoryUri": "770424767712.dkr.ecr.ap-south-1.amazonaws.com/flask-app-repo",
        "createdAt": "2025-07-09T12:04:46.553000+00:00",
        "imageTagMutability": "MUTABLE",
        "imageScanningConfiguration": {
            "scanOnPush": false
        },
        "encryptionConfiguration": {
            "encryptionType": "AES256"
        }
    }
}
[ec2-user@ip-172-31-6-178 flask-app]$ aws ecr get-login-password --region ap-south-1 | docker
login --username AWS --password-stdin 770424767712.dkr.ecr.ap-south-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[ec2-user@ip-172-31-6-178 flask-app]$ docker tag flask-app
770424767712.dkr.ecr.ap-south-1.amazonaws.com/flask-app-repo
[ec2-user@ip-172-31-6-178 flask-app]$ docker push
770424767712.dkr.ecr.ap-south-1.amazonaws.com/flask-app-repo
Using default tag: latest
```




Amazon Elastic Container Service
Clusters
Namespaces
Task definitions
Account settings
Amazon ECR
Repositories
AWS Batch
Documentation
Discover products
Subscriptions
Tell us what you think
flask-fargate-cluster-2
Update cluster
Delete cluster
Cluster overview
ARN
arn:aws:ecs:ap-south-1:770424767712:cluster/flask-fargate-cluster-2
Status
Active
CloudWatch monitoring
Container Insights with enhanced observability
View in CloudWatch
Registered container instances
Services
Draining
Active
Tasks
Pending
Running
Services
Tasks
Infrastructure
Metrics
Scheduled tasks
Configuration
Tags
Services (1) Info
Manage tags
Update
Delete service
Create
Filter services by value
Filter launch type
Any launch type
Filter service type
Any service type
Service name
ARN
Status
Service type
Created at
Deployments and tasks
Last deployment
Task definition
flask-task-def-service-ase44o58
Active
REPLICA
21 hours ago
1/1 Tasks running
Completed | View
flask-task-def:1


Amazon Elastic Container Service
Clusters
Namespaces
Task definitions
Account settings
Amazon ECR
Repositories
AWS Batch
Documentation
Discover products
Subscriptions
Tell us what you think
Service auto scaling - optional
Automatically adjust your service's desired count up and down within a specified range in response to CloudWatch alarms. You can modify your service auto scaling configuration at any time to meet the needs of your application.
You can now configure predictive scaling for your ECS services by using the service auto scaling section on the Service detail page. This dedicated section enables you to configure all types of scaling policies, set up scheduled scaling actions, and track scaling activities. Learn more
Use service auto scaling
Configure service auto scaling to adjust your service's desired count
Minimum number of tasks
The lower boundary to which service auto scaling can adjust the desired count of the service.
1
Maximum number of tasks
The upper boundary to which service auto scaling can adjust the desired count of the service.
3
Scaling policy
Remove
Scaling policy type
Target tracking
Policy name
flask-cpu-target-50
ECS service metric
ECSServiceAverageCPUUtilization
Target value
50
Scale-out cooldown period
60
Scale-in cooldown period
60
Turn off scale-in


EC2
Load balancers
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Load balancers (1/1)
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.
Actions
Create load balancer
Filter load balancers
Name
DNS name
State
VPC ID
Availability Zones
Type
Date created
flask-target-load-balancer
flask-target-load-balancer-...
Active
vpc-0b408ac844bffa015
3 Availability Zones
application
July 9, 2025, 18:20 (UTC+05:30)
Load balancer: flask-target-load-balancer
Details
Listeners and rules
Network mapping
Resource map
Security
Monitoring
Integrations
Attributes
Capacity
Tags
Details
Load balancer type
Application
Status
Active
VPC
vpc-0b408ac844bffa015
Load balancer IP address type
IPv4
Scheme
Internet-facing
Hosted zone
ZP97RAFLXTNZK
Availability Zones
subnet-0c74fc4414a955635 ap-south-1c (aps1-az2)
subnet-04d2d7500060a2f59 ap-south-1a (aps1-az1)
subnet-0bbe0751c9a03a969 ap-south-1b (aps1-az3)
Date created
July 9, 2025, 18:20 (UTC+05:30)
Load balancer ARN
arn:aws:elasticloadbalancing:ap-south-1:770424767712:loadbalancer/app/flask-target-load-balancer/54718bdfaf17bdb5
DNS name Info
flask-target-load-balancer-1278756371.ap-south-1.elb.amazonaws.com (A Record)