AWS Deployment Assignment – Flask & Express Application

Name : Harshil Bhardwaj

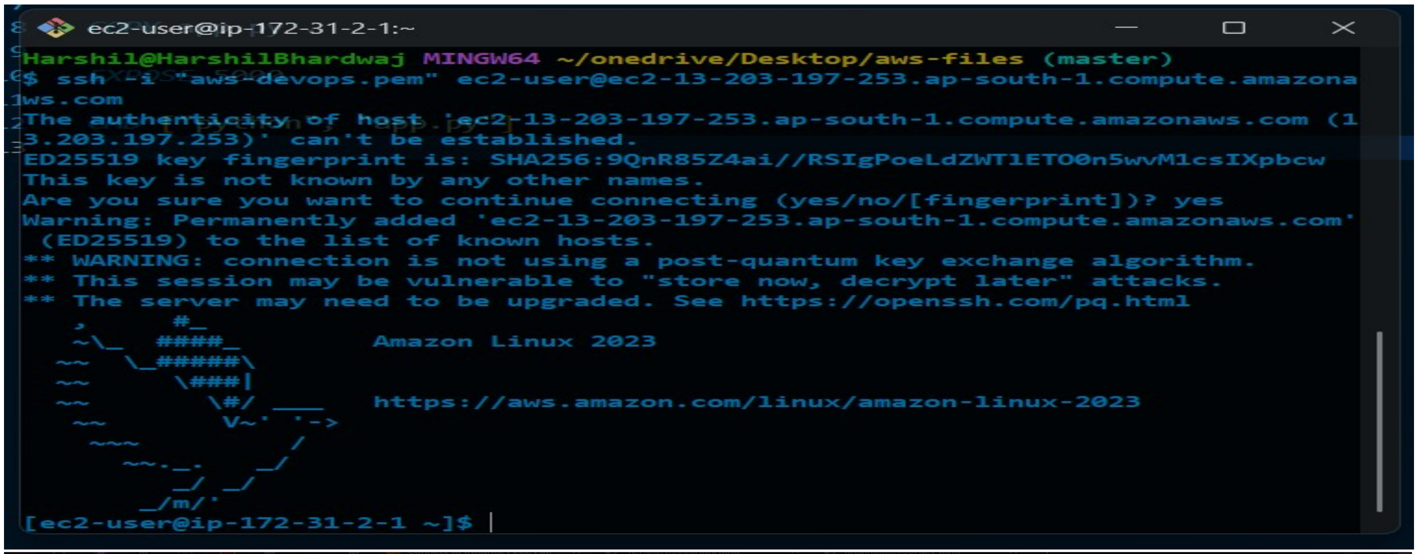
TASK – 1: Deploy Flask Backend & Express Frontend on a Single EC2 Instance

Objective:
Deploy both Flask backend and Express frontend on a single Amazon EC2 instance.

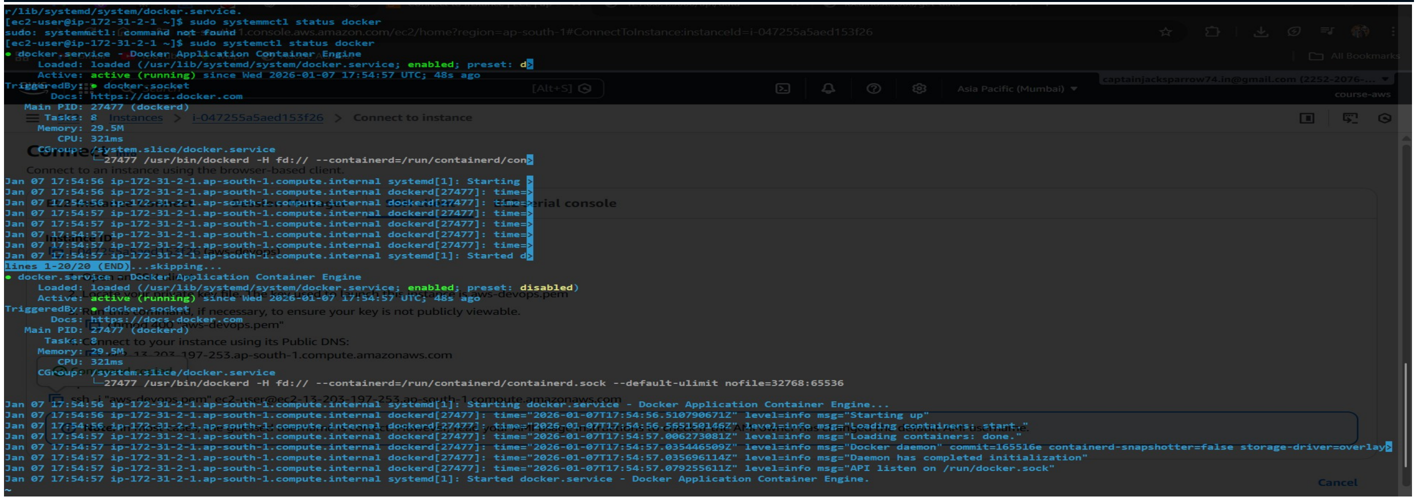
Steps Performed:
- Launched Amazon EC2 instance (Amazon Linux)
- Installed Python, Flask, Node.js, and Express
- Ran Flask backend on port 5000
- Ran Express frontend on port 3000
- Configured Security Group to allow ports 3000 and 5000

Result:
Both frontend and backend were accessible via EC2 public IP.

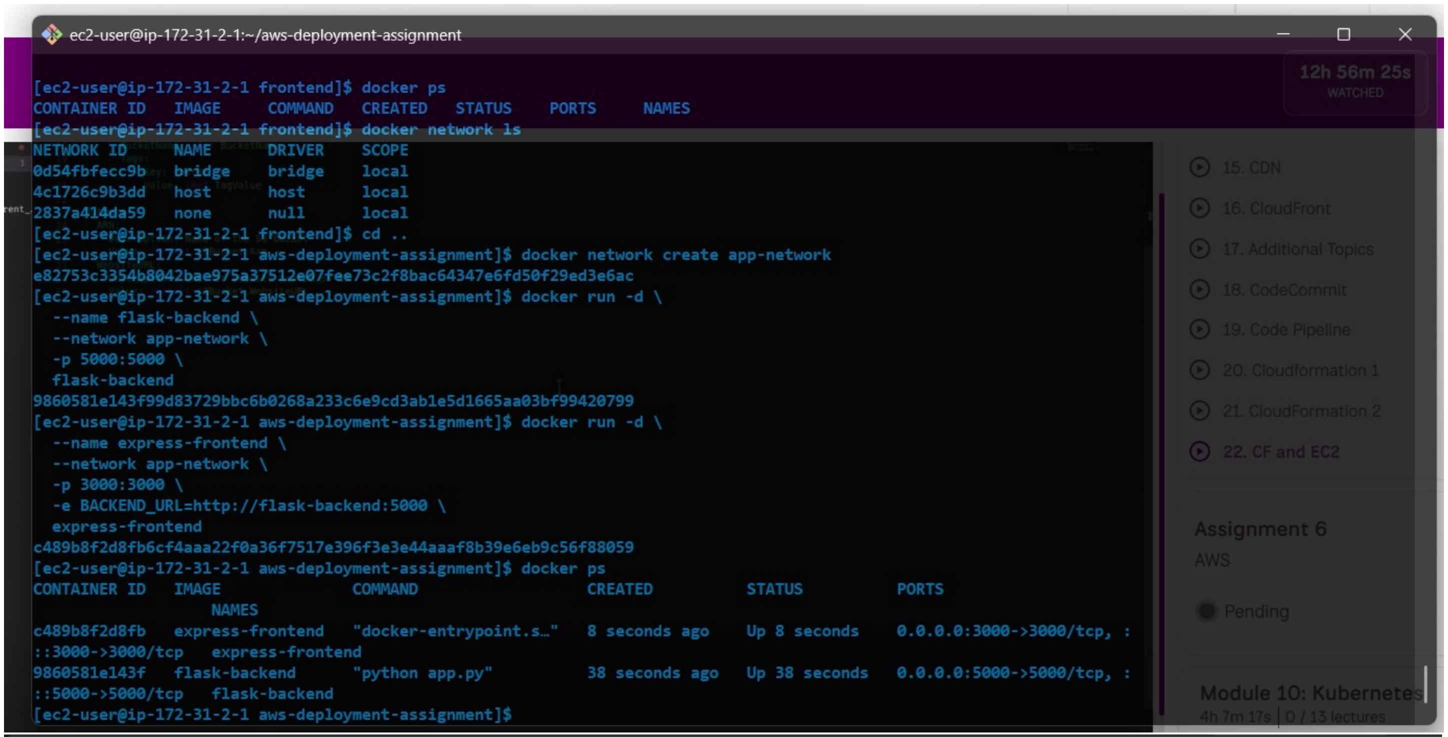
Screenshot:



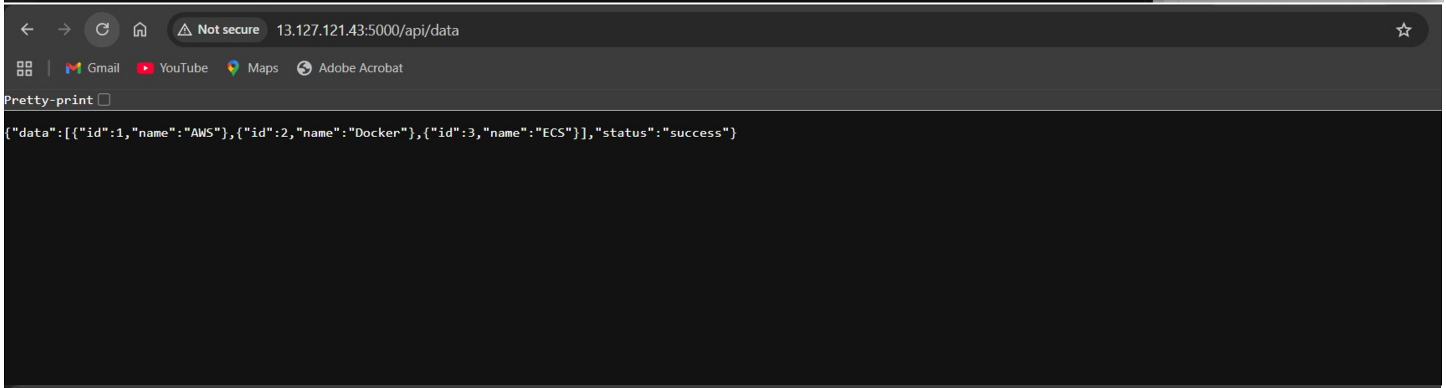
```
ec2-user@ip-172-31-2-1:~
Harshil@HarshilBhardwaj MINGW64 ~/onedrive/Desktop/aws-files (master)
$ ssh -i "aws-devops.pem" ec2-user@ec2-13-203-197-253.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-13-203-197-253.ap-south-1.compute.amazonaws.com (13.203.197.253)' can't be established.
ED25519 key fingerprint is: SHA256:9QnR85Z4ai//RSIgPoeLdZWT1ETO0n5wvM1csIXpbcw
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-13-203-197-253.ap-south-1.compute.amazonaws.com' (ED25519) to the list of known hosts.
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-2-1 ~]$ |
```



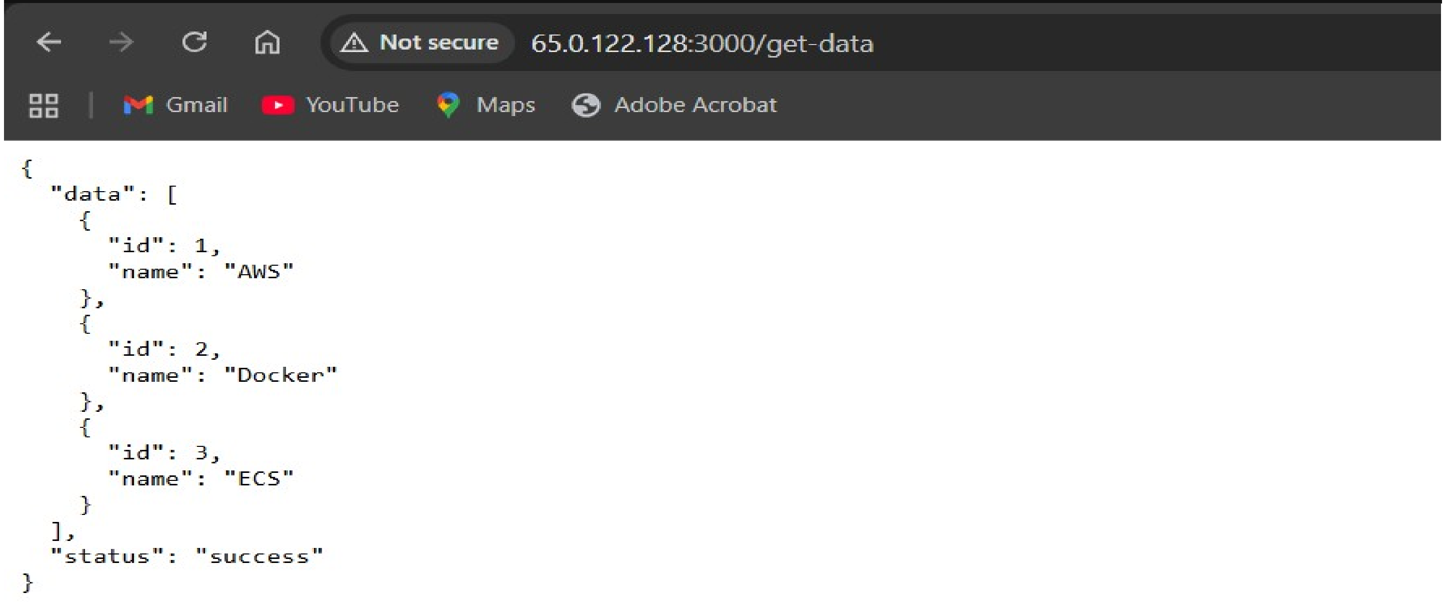
```
r/lib/systemd/system/docker.service.
[ec2-user@ip-172-31-2-1 ~]$ sudo systemmctl status docker
sudo: systemmctl: command not found
[ec2-user@ip-172-31-2-1 ~]$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: d>
     Active: active (running) since Wed 2026-01-07 17:54:57 UTC; 48s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 27477 (dockerd)
      Tasks: 8
     Memory: 29.5M
        CPU: 321ms
     CGroup: /system.slice/docker.service
             └─27477 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/con>

Jan 07 17:54:56 ip-172-31-2-1.ap-south-1.compute.internal systemd[1]: Starting >
Jan 07 17:54:56 ip-172-31-2-1.ap-south-1.compute.internal dockerd[27477]: time=>
Jan 07 17:54:56 ip-172-31-2-1.ap-south-1.compute.internal dockerd[27477]: time=>
Jan 07 17:54:57 ip-172-31-2-1.ap-south-1.compute.internal dockerd[27477]: time=>
Jan 07 17:54:57 ip-172-31-2-1.ap-south-1.compute.internal dockerd[27477]: time=>
Jan 07 17:54:57 ip-172-31-2-1.ap-south-1.compute.internal dockerd[27477]: time=>
Jan 07 17:54:57 ip-172-31-2-1.ap-south-1.compute.internal dockerd[27477]: time=>
Jan 07 17:54:57 ip-172-31-2-1.ap-south-1.compute.internal systemd[1]: Started d>
lines 1-20/20 (END)...skipping...
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: disabled)
     Active: active (running) since Wed 2026-01-07 17:54:57 UTC; 48s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 27477 (dockerd)
      Tasks: 8
     Memory: 29.5M
        CPU: 321ms
     CGroup: /system.slice/docker.service
             └─27477 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536

Jan 07 17:54:56 ip-172-31-2-1.ap-south-1.compute.internal systemd[1]: Starting docker.service - Docker Application Container Engine...
Jan 07 17:54:56 ip-172-31-2-1.ap-south-1.compute.internal dockerd[27477]: time="2026-01-07T17:54:56.510790671Z" level=info msg="Starting up"
Jan 07 17:54:56 ip-172-31-2-1.ap-south-1.compute.internal dockerd[27477]: time="2026-01-07T17:54:56.565150146Z" level=info msg="Loading containers: start."
Jan 07 17:54:57 ip-172-31-2-1.ap-south-1.compute.internal dockerd[27477]: time="2026-01-07T17:54:57.006273081Z" level=info msg="Loading containers: done."
Jan 07 17:54:57 ip-172-31-2-1.ap-south-1.compute.internal dockerd[27477]: time="2026-01-07T17:54:57.035446509Z" level=info msg="Docker daemon" commit=165516e containerd-snapshotter=false storage-driver=overlay>
Jan 07 17:54:57 ip-172-31-2-1.ap-south-1.compute.internal dockerd[27477]: time="2026-01-07T17:54:57.035696114Z" level=info msg="Daemon has completed initialization"
Jan 07 17:54:57 ip-172-31-2-1.ap-south-1.compute.internal dockerd[27477]: time="2026-01-07T17:54:57.079255611Z" level=info msg="API listen on /run/docker.sock"
Jan 07 17:54:57 ip-172-31-2-1.ap-south-1.compute.internal systemd[1]: Started docker.service - Docker Application Container Engine.
~
```



```
[ec2-user@ip-172-31-2-1 frontend]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[ec2-user@ip-172-31-2-1 frontend]$ docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
0d54fbfecc9b   bridge    bridge    local
4c1726c9b3dd   host      host      local
2837a414da59   none      null      local
[ec2-user@ip-172-31-2-1 frontend]$ cd ..
[ec2-user@ip-172-31-2-1 aws-deployment-assignment]$ docker network create app-network
e82753c3354b8042bae975a37512e07fee73c2f8bac64347e6fd50f29ed3e6ac
[ec2-user@ip-172-31-2-1 aws-deployment-assignment]$ docker run -d \
  --name flask-backend \
  --network app-network \
  -p 5000:5000 \
  flask-backend
9860581e143f99d83729bbc6b0268a233c6e9cd3ab1e5d1665aa03bf99420799
[ec2-user@ip-172-31-2-1 aws-deployment-assignment]$ docker run -d \
  --name express-frontend \
  --network app-network \
  -p 3000:3000 \
  -e BACKEND_URL=http://flask-backend:5000 \
  express-frontend
c489b8f2d8fb6cf4aaa22f0a36f7517e396f3e3e44aaaf8b39e6eb9c56f88059
[ec2-user@ip-172-31-2-1 aws-deployment-assignment]$ docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS          PORTS
                    NAMES
c489b8f2d8fb   express-frontend   "docker-entrypoint.s…"   8 seconds ago    Up 8 seconds    0.0.0.0:3000->3000/tcp, :
::3000->3000/tcp   express-frontend
9860581e143f   flask-backend      "python app.py"          38 seconds ago   Up 38 seconds   0.0.0.0:5000->5000/tcp, :
::5000->5000/tcp   flask-backend
[ec2-user@ip-172-31-2-1 aws-deployment-assignment]$
```



```
{"data":[{"id":1,"name":"AWS"},{"id":2,"name":"Docker"},{"id":3,"name":"ECS"}],"status":"success"}
```



```
{
  "data": [
    {
      "id": 1,
      "name": "AWS"
    },
    {
      "id": 2,
      "name": "Docker"
    },
    {
      "id": 3,
      "name": "ECS"
    }
  ],
  "status": "success"
}
```

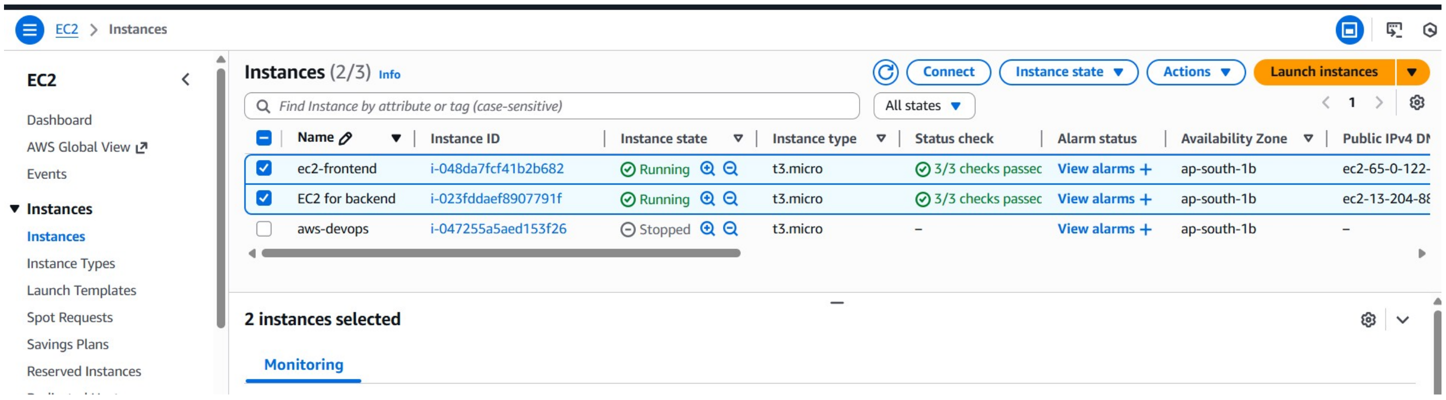
TASK – 2: Deploy Flask Backend & Express Frontend on Separate EC2 Instances

Objective:
Deploy backend and frontend on different EC2 instances.

Steps Performed:
- Launched two EC2 instances
- Backend EC2 ran Flask on port 5000
- Frontend EC2 ran Express on port 3000
- Configured Security Groups for secure communication

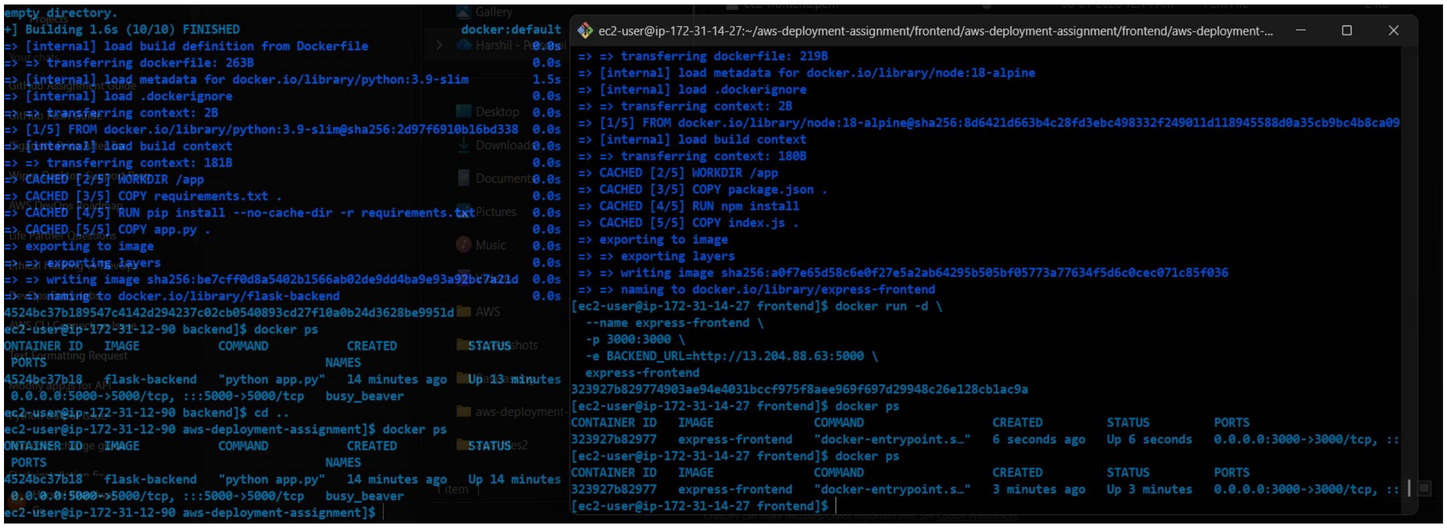
Result:
Frontend successfully fetched data from backend EC2.

Screenshot:
Frontend EC2 fetching data from Backend EC2



Instances (2/3) Info

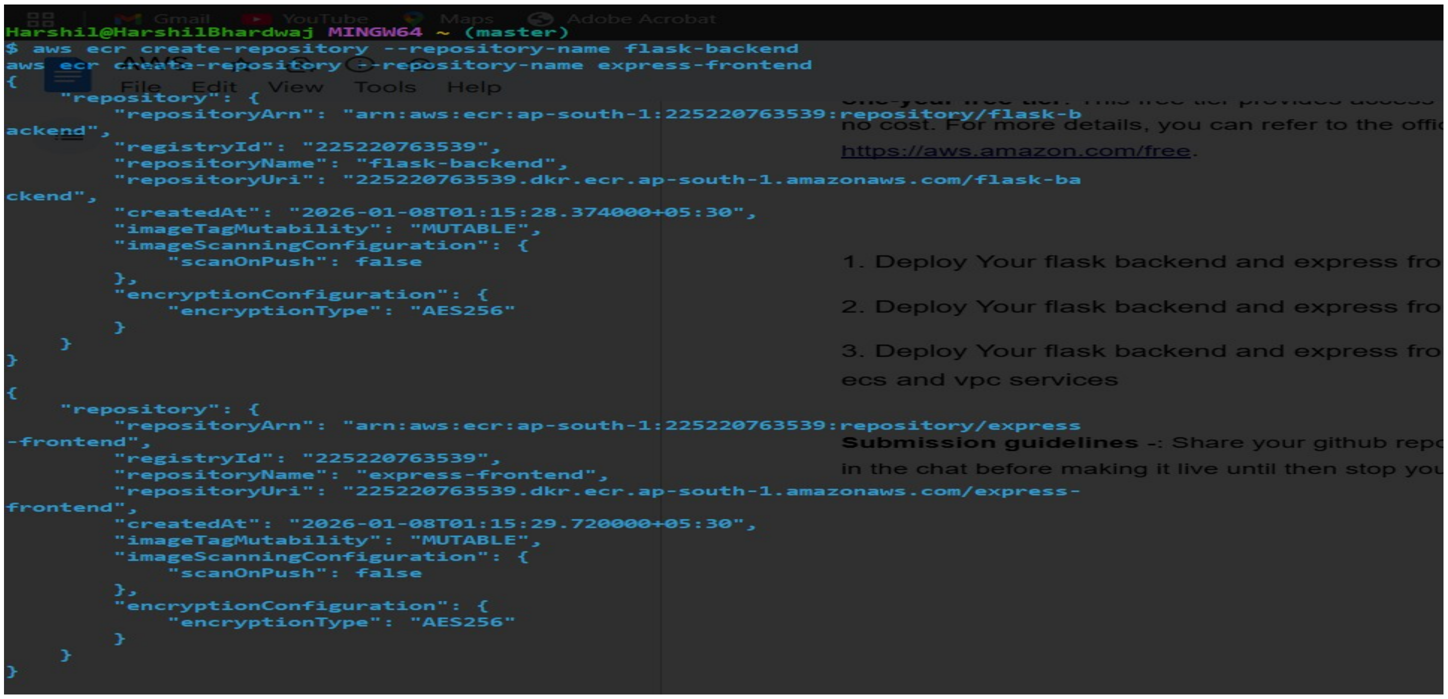
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DI
ec2-frontend	i-048da7fcf41b2b682	Running	t3.micro	3/3 checks passec	View alarms +	ap-south-1b	ec2-65-0-122-
EC2 for backend	i-023fddaef8907791f	Running	t3.micro	3/3 checks passec	View alarms +	ap-south-1b	ec2-13-204-88
aws-devops	i-047255a5aed153f26	Stopped	t3.micro	–	View alarms +	ap-south-1b	–



```
empty directory.
+] Building 1.6s (10/10) FINISHED                                    docker:default
=> [internal] load build definition from Dockerfile                           0.0s
=> => transferring dockerfile: 263B                                           0.0s
=> [internal] load metadata for docker.io/library/python:3.9-slim             1.5s
=> [internal] load .dockerignore                                              0.0s
=> => transferring context: 2B                                                0.0s
=> [1/5] FROM docker.io/library/python:3.9-slim@sha256:2d97f6910b16bd338      0.0s
=> [internal] load build context                                              0.0s
=> => transferring context: 181B                                              0.0s
=> CACHED [2/5] WORKDIR /app                                                  0.0s
=> CACHED [3/5] COPY requirements.txt .                                       0.0s
=> CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt            0.0s
=> CACHED [5/5] COPY app.py .                                                 0.0s
=> exporting to image                                                         0.0s
=> => exporting layers                                                        0.0s
=> => writing image sha256:be7cff0d8a5402b1566ab02de9dd4ba9e93a92bc7a21d      0.0s
=> => naming to docker.io/library/flask-backend                               0.0s
4524bc37b189547c4142d294237c02cb0540893cd27f10a0b24d3628be9951d
ec2-user@ip-172-31-12-90 backend]$ docker ps
ONTAINER ID   IMAGE           COMMAND           CREATED          STATUS
 PORTS                                      NAMES
4524bc37b18   flask-backend   "python app.py"   14 minutes ago   Up 13 minutes
 0.0.0.0:5000->5000/tcp, :::5000->5000/tcp   busy_beaver
ec2-user@ip-172-31-12-90 backend]$ cd ..
ec2-user@ip-172-31-12-90 aws-deployment-assignment]$ docker ps
ONTAINER ID   IMAGE           COMMAND           CREATED          STATUS
 PORTS                                      NAMES
4524bc37b18   flask-backend   "python app.py"   14 minutes ago   Up 14 minutes
 0.0.0.0:5000->5000/tcp, :::5000->5000/tcp   busy_beaver
ec2-user@ip-172-31-12-90 aws-deployment-assignment]$
```

ec2-user@ip-172-31-14-27:~/aws-deployment-assignment/frontend/aws-deployment-assignment/frontend/aws-deployment-...

```
=> => transferring dockerfile: 219B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca09
=> [internal] load build context
=> => transferring context: 180B
=> CACHED [2/5] WORKDIR /app
=> CACHED [3/5] COPY package.json .
=> CACHED [4/5] RUN npm install
=> CACHED [5/5] COPY index.js .
=> exporting to image
=> => exporting layers
=> => writing image sha256:a0f7e65d58c6e0f27e5a2ab64295b505bf05773a77634f5d6c0cec071c85f036
=> => naming to docker.io/library/express-frontend
[ec2-user@ip-172-31-14-27 frontend]$ docker run -d \
  --name express-frontend \
  -p 3000:3000 \
  -e BACKEND_URL=http://13.204.88.63:5000 \
  express-frontend
323927b829774903ae94e4031bccf975f8aee969f697d29948c26e128cb1ac9a
[ec2-user@ip-172-31-14-27 frontend]$ docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS           PORTS
323927b82977   express-frontend   "docker-entrypoint.s…"   6 seconds ago    Up 6 seconds     0.0.0.0:3000->3000/tcp, ::
[ec2-user@ip-172-31-14-27 frontend]$ docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS           PORTS
323927b82977   express-frontend   "docker-entrypoint.s…"   3 minutes ago    Up 3 minutes     0.0.0.0:3000->3000/tcp, ::
[ec2-user@ip-172-31-14-27 frontend]$
```



```
Harshil@HarshilBhardwaj MINGW64 ~ (master)
$ aws ecr create-repository --repository-name flask-backend
aws ecr create-repository --repository-name express-frontend
{
    "repository": {
        "repositoryArn": "arn:aws:ecr:ap-south-1:225220763539:repository/flask-b
ackend",
        "registryId": "225220763539",
        "repositoryName": "flask-backend",
        "repositoryUri": "225220763539.dkr.ecr.ap-south-1.amazonaws.com/flask-ba
ckend",
        "createdAt": "2026-01-08T01:15:28.374000+05:30",
        "imageTagMutability": "MUTABLE",
        "imageScanningConfiguration": {
            "scanOnPush": false
        },
        "encryptionConfiguration": {
            "encryptionType": "AES256"
        }
    }
}

{
    "repository": {
        "repositoryArn": "arn:aws:ecr:ap-south-1:225220763539:repository/express
-frontend",
        "registryId": "225220763539",
        "repositoryName": "express-frontend",
        "repositoryUri": "225220763539.dkr.ecr.ap-south-1.amazonaws.com/express-
frontend",
        "createdAt": "2026-01-08T01:15:29.720000+05:30",
        "imageTagMutability": "MUTABLE",
        "imageScanningConfiguration": {
            "scanOnPush": false
        },
        "encryptionConfiguration": {
            "encryptionType": "AES256"
        }
    }
}
```

TASK – 3: Deploy Flask & Express using Docker, ECR, ECS & ALB

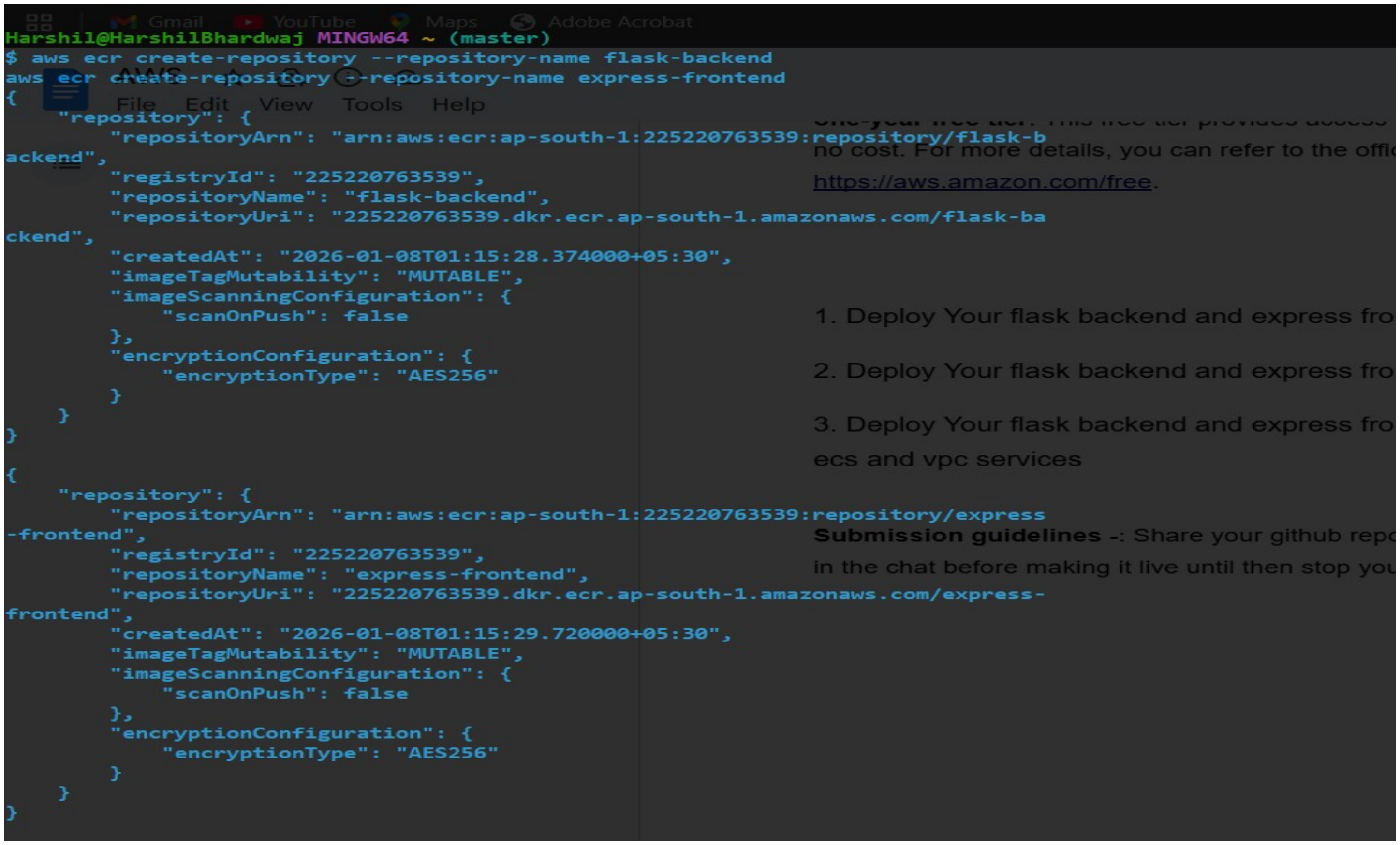
Objective:
Deploy containerized applications using AWS services.

Steps Performed:
- Created Docker images for backend and frontend
- Pushed images to Amazon ECR
- Created ECS Cluster (Fargate)
- Configured Application Load Balancer with path-based routing

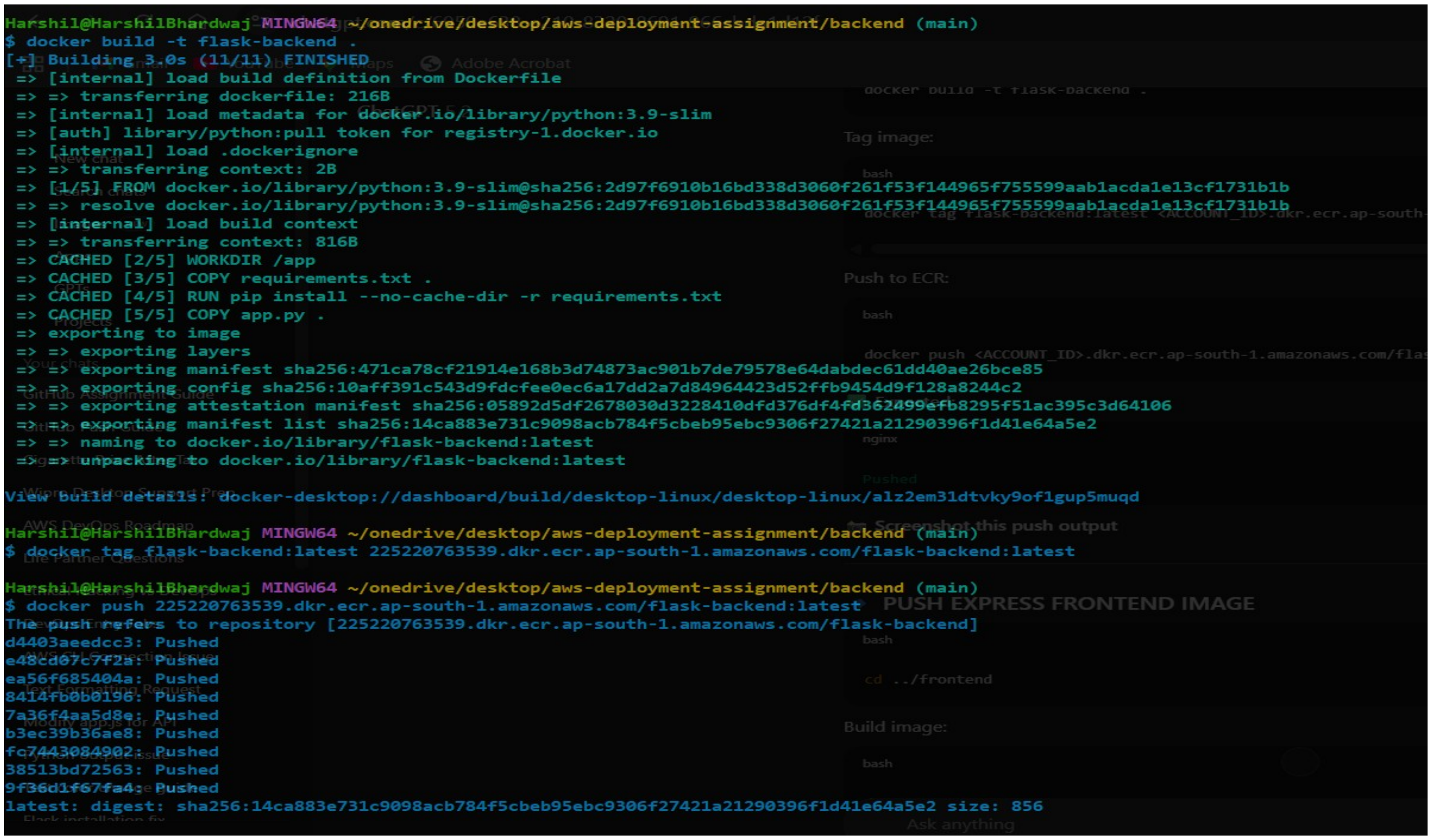
Result:
Application accessible via ALB DNS URL.

Screenshots:

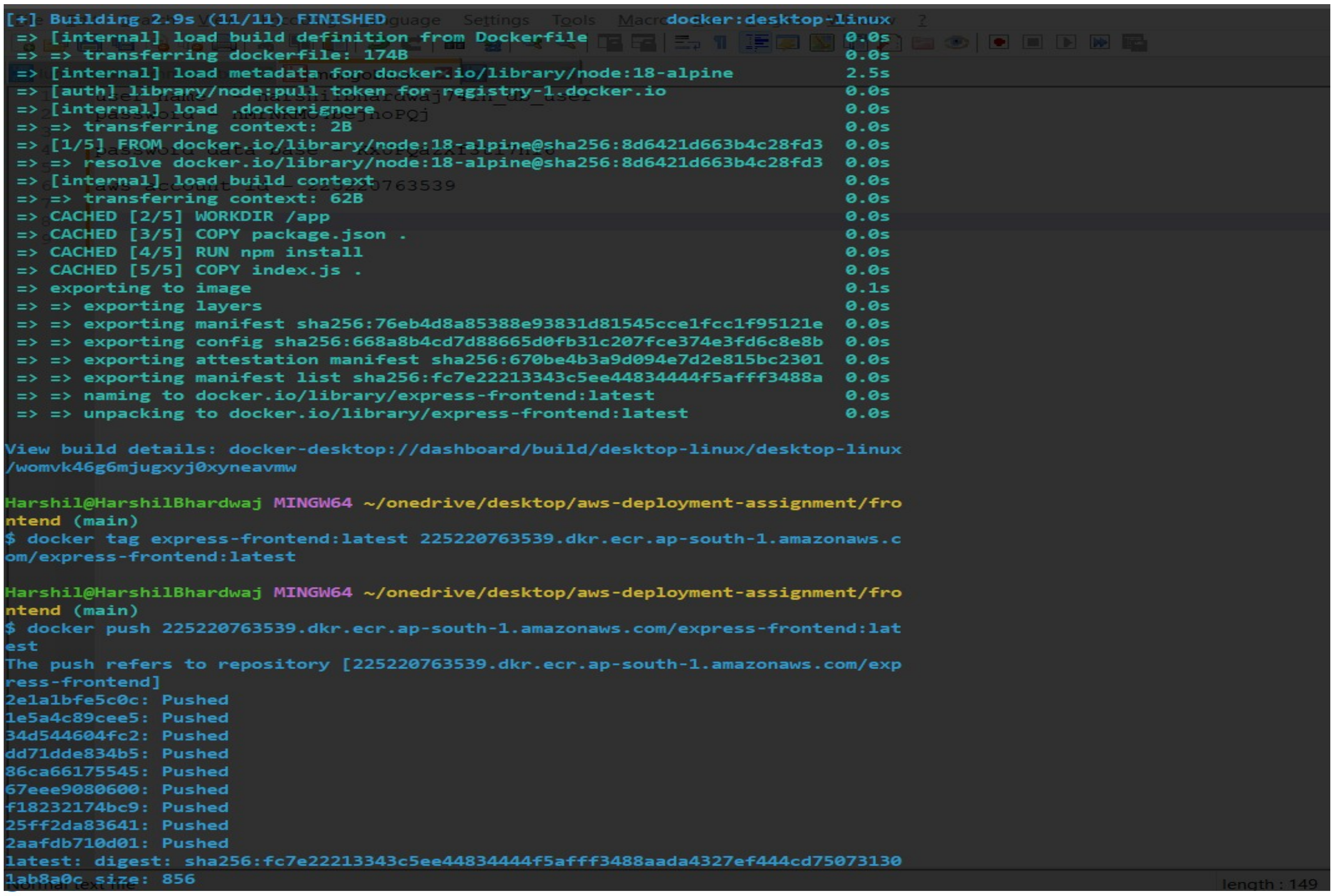
ECS services, target groups, and ALB output



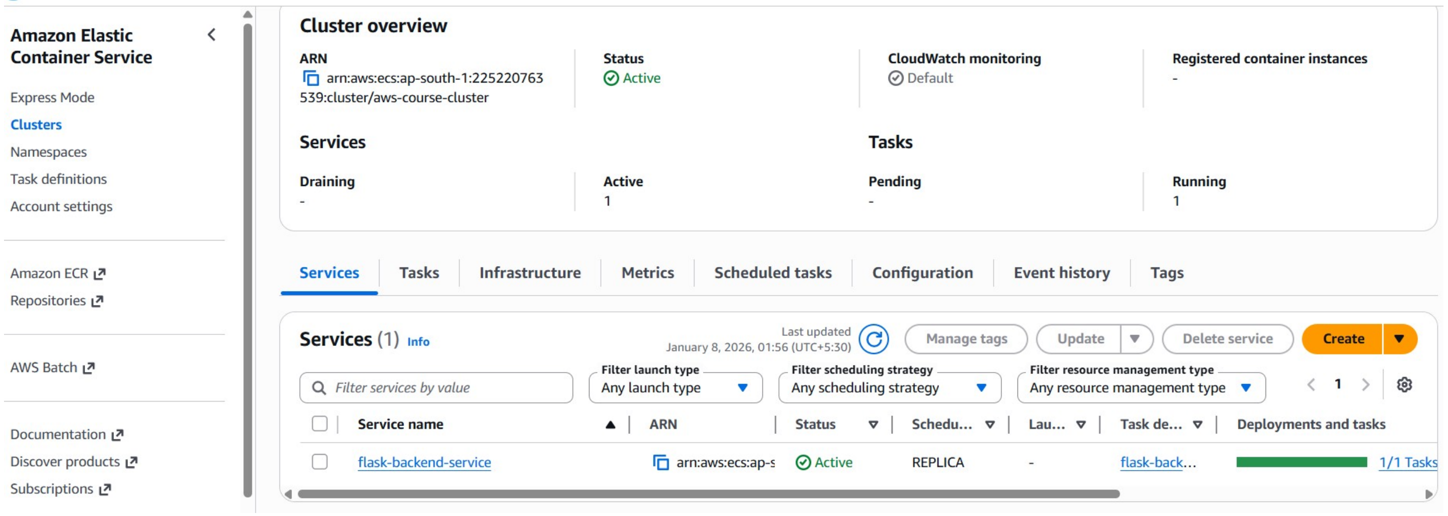
```
Harshil@HarshilBhardwaj MINGW64 ~ (master)
$ aws ecr create-repository --repository-name flask-backend
aws ecr create-repository --repository-name express-frontend
{
    "repository": {
        "repositoryArn": "arn:aws:ecr:ap-south-1:225220763539:repository/flask-b
ackend",
        "registryId": "225220763539",
        "repositoryName": "flask-backend",
        "repositoryUri": "225220763539.dkr.ecr.ap-south-1.amazonaws.com/flask-ba
ckend",
        "createdAt": "2026-01-08T01:15:28.374000+05:30",
        "imageTagMutability": "MUTABLE",
        "imageScanningConfiguration": {
            "scanOnPush": false
        },
        "encryptionConfiguration": {
            "encryptionType": "AES256"
        }
    }
}

{
    "repository": {
        "repositoryArn": "arn:aws:ecr:ap-south-1:225220763539:repository/express
-frontend",
        "registryId": "225220763539",
        "repositoryName": "express-frontend",
        "repositoryUri": "225220763539.dkr.ecr.ap-south-1.amazonaws.com/express-
frontend",
        "createdAt": "2026-01-08T01:15:29.720000+05:30",
        "imageTagMutability": "MUTABLE",
        "imageScanningConfiguration": {
            "scanOnPush": false
        },
        "encryptionConfiguration": {
            "encryptionType": "AES256"
        }
    }
}
```



```
Harshil@HarshilBhardwaj MINGW64 ~/onedrive/desktop/aws-deployment-assignment/backend (main)
$ docker build -t flask-backend .
[+] Building 3.0s (11/11) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 216B
 => [internal] load metadata for docker.io/library/python:3.9-slim
 => [auth] library/python:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/5] FROM docker.io/library/python:3.9-slim@sha256:2d97f6910b16bd338d3060f261f53f144965f755599aab1acda1e13cf1731b1b
 => => resolve docker.io/library/python:3.9-slim@sha256:2d97f6910b16bd338d3060f261f53f144965f755599aab1acda1e13cf1731b1b
 => [internal] load build context
 => => transferring context: 816B
 => CACHED [2/5] WORKDIR /app
 => CACHED [3/5] COPY requirements.txt .
 => CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt
 => CACHED [5/5] COPY app.py .
 => exporting to image
 => => exporting layers
 => => exporting manifest sha256:471ca78cf21914e168b3d74873ac901b7de79578e64dabdec61dd40ae26bce85
 => => exporting config sha256:10aff391c543d9fdcfee0ec6a17dd2a7d84964423d52ffb9454d9f128a8244c2
 => => exporting attestation manifest sha256:05892d5df2678030d3228410dfd376df4fd362499efb8295f51ac395c3d64106
 => => exporting manifest list sha256:14ca883e731c9098acb784f5cbeb95ebc9306f27421a21290396f1d41e64a5e2
 => => naming to docker.io/library/flask-backend:latest
 => => unpacking to docker.io/library/flask-backend:latest

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/alz2em31dtvky9of1gup5muqd

Harshil@HarshilBhardwaj MINGW64 ~/onedrive/desktop/aws-deployment-assignment/backend (main)
$ docker tag flask-backend:latest 225220763539.dkr.ecr.ap-south-1.amazonaws.com/flask-backend:latest

Harshil@HarshilBhardwaj MINGW64 ~/onedrive/desktop/aws-deployment-assignment/backend (main)
$ docker push 225220763539.dkr.ecr.ap-south-1.amazonaws.com/flask-backend:latest
The push refers to repository [225220763539.dkr.ecr.ap-south-1.amazonaws.com/flask-backend]
d4403aeedcc3: Pushed
e48cd67c7f2a: Pushed
ea56f685404a: Pushed
8414fb0b0196: Pushed
7a36f4aa5d8e: Pushed
b3ec39b36ae8: Pushed
fc7443084902: Pushed
38513bd72563: Pushed
9f36d1f67fa4: Pushed
latest: digest: sha256:14ca883e731c9098acb784f5cbeb95ebc9306f27421a21290396f1d41e64a5e2 size: 856
```



```
[+] Building 2.9s (11/11) FINISHED                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                              0.0s
 => => transferring dockerfile: 174B                                              0.0s
 => [internal] load metadata for docker.io/library/node:18-alpine                 2.5s
 => [auth] library/node:pull token for registry-1.docker.io                       0.0s
 => [internal] load .dockerignore                                                 0.0s
 => => transferring context: 2B                                                   0.0s
 => [1/5] FROM docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3         0.0s
 => => resolve docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3         0.0s
 => [internal] load build context                                                 0.0s
 => => transferring context: 62B                                                  0.0s
 => CACHED [2/5] WORKDIR /app                                                     0.0s
 => CACHED [3/5] COPY package.json .                                              0.0s
 => CACHED [4/5] RUN npm install                                                  0.0s
 => CACHED [5/5] COPY index.js .                                                  0.0s
 => exporting to image                                                            0.1s
 => => exporting layers                                                           0.0s
 => => exporting manifest sha256:76eb4d8a85388e93831d81545cce1fcc1f95121e         0.0s
 => => exporting config sha256:668a8b4cd7d88665d0fb31c207fce374e3fd6c8e8b         0.0s
 => => exporting attestation manifest sha256:670be4b3a9d094e7d2e815bc2301         0.0s
 => => exporting manifest list sha256:fc7e22213343c5ee44834444f5afff3488a         0.0s
 => => naming to docker.io/library/express-frontend:latest                        0.0s
 => => unpacking to docker.io/library/express-frontend:latest                     0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux
/womvk46g6mjugxyj0xyneavmw

Harshil@HarshilBhardwaj MINGW64 ~/onedrive/desktop/aws-deployment-assignment/fro
ntend (main)
$ docker tag express-frontend:latest 225220763539.dkr.ecr.ap-south-1.amazonaws.c
om/express-frontend:latest

Harshil@HarshilBhardwaj MINGW64 ~/onedrive/desktop/aws-deployment-assignment/fro
ntend (main)
$ docker push 225220763539.dkr.ecr.ap-south-1.amazonaws.com/express-frontend:lat
est
The push refers to repository [225220763539.dkr.ecr.ap-south-1.amazonaws.com/exp
ress-frontend]
2e1a1bfe5c0c: Pushed
1e5a4c89cee5: Pushed
34d544604fc2: Pushed
dd71dde834b5: Pushed
86ca66175545: Pushed
67eee9080600: Pushed
f18232174bc9: Pushed
25ff2da83641: Pushed
2aafdb710d01: Pushed
latest: digest: sha256:fc7e22213343c5ee44834444f5afff3488aada4327ef444cd75073130
1ab8a0c size: 856
```



Cluster overview

ARN: arn:aws:ecs:ap-south-1:225220763539:cluster/aws-course-cluster
Status: Active
CloudWatch monitoring: Default
Registered container instances: -

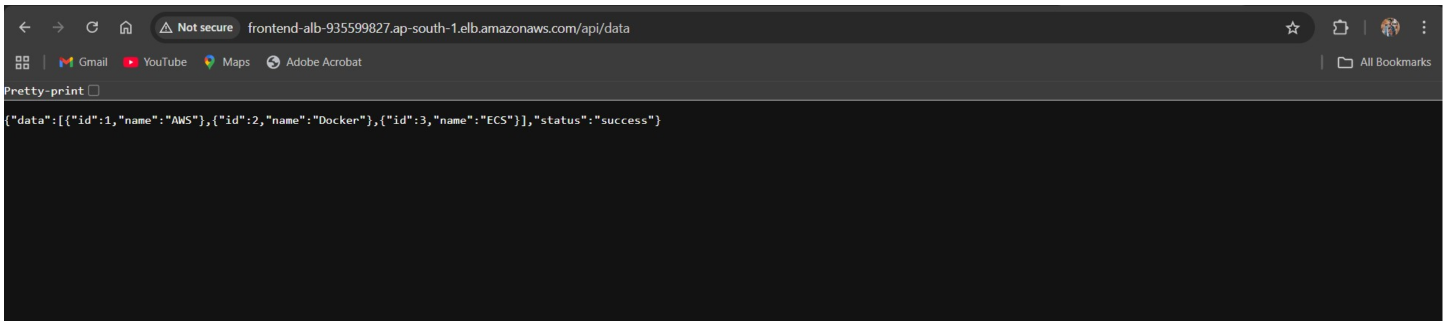
Services
Draining: -
Active: 1

Tasks
Pending: -
Running: 1

Services (1) Info — Last updated January 8, 2026, 01:56 (UTC+5:30)

Filter launch type: Any launch type
Filter scheduling strategy: Any scheduling strategy
Filter resource management type: Any resource management type

Service name	ARN	Status	Schedu...	Lau...	Task de...	Deployments and tasks
flask-backend-service	arn:aws:ecs:ap-s	Active	REPLICA	-	flask-back...	1/1 Tasks



Not secure frontend-alb-935599827.ap-south-1.elb.amazonaws.com/api/data

```
{"data":[{"id":1,"name":"AWS"},{"id":2,"name":"Docker"},{"id":3,"name":"ECS"}],"status":"success"}
```

Conclusion

All three deployment strategies were successfully implemented and verified using AWS services.

Thank you