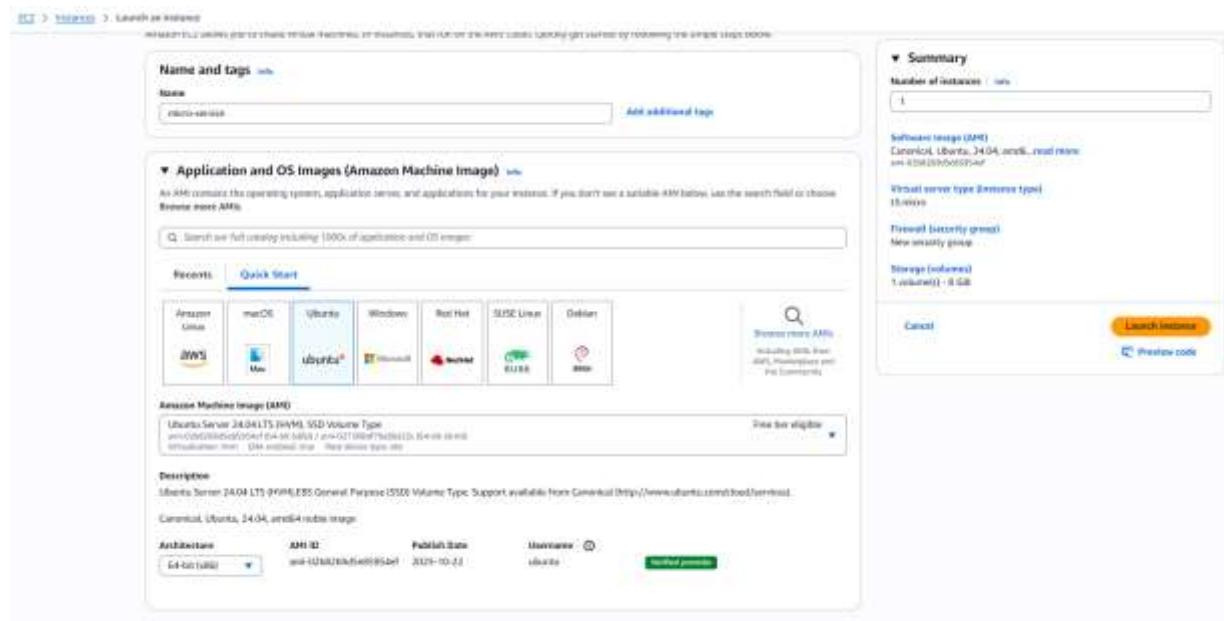


AWS Assignment

1.



Create Aws EC2 instance

```
ubuntu@ip-172-31-8-32:~$  
177.0.0.1 - [04/Jan/2020 09:11:53] "GET /Devops-practice/Flask/ HTTP/1.0" 200 -  
177.0.0.1 - [04/Jan/2020 09:11:58] "GET /Devops-practice/Flask/Frontend/ HTTP/1.0" 200 -  
177.0.0.1 - [04/Jan/2020 09:11:05] "GET /Devops-practice/Flask/Frontend/templates/ HTTP/1.0" 200 -  
177.0.0.1 - [04/Jan/2020 09:11:11] "GET /Devops-practice/Flask/Frontend/app.py HTTP/1.0" 200 -  
Read from remote host ec2-13-225-90-96.ap-south-1.compute.amazonaws.com: Connection reset by peer.  
Connection to ec2-13-225-90-96.ap-south-1.compute.amazonaws.com closed.  
ClientLoop: send disconnect! Connection reset by peer.  
ubuntu@ip-172-31-8-32:~/Desktop$  
I 1s  
AES-MicroService.rkt aes_ec2.pem http://127.0.0.1:5555/microservice.rkt  
  
KINOWA84:~/Desktop$ KINOWA84 ./microservice/AWS  
I ssh -l 'aws-ec2-pem' ubuntu@ec2-13-109-209-206.ap-south-1.compute.amazonaws.com  
The authenticity of host 'ec2-13-109-209-206.ap-south-1.compute.amazonaws.com (<109.209.206>)' can't be established.  
ECDSA key fingerprint is SHA256:09zD0ghH2L3LTT77q9MhMmMrVnxxJ5nQw+IQ333.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added 'ec2-13-109-209-206.ap-south-1.compute.amazonaws.com' (ED25519) to the list of known hosts.  
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 6.14.0-1013-mus n86_64)  
  
Documentation: https://help.ubuntu.com  
Management: https://landscape.canonical.com  
Support: https://ubuntu.com/pre  
  
System information as of Fri Jan 9 14:45:05 UTC 2020  
  
System Load: 0.61 Temperature: +27.1 °C  
Usage of /: 25.8% of 8.71GB Processes: 215  
Memory usage: 23% users logged in: 0  
Swap usage: 0% IPv4 address for ens3: 172.31.8.32  
  
Expanded Security Maintenance for Applications is not enabled.  
0 updates can be applied immediately.  
  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run sudo pro status  
  
The list of available updates is more than a week old.  
To check for new updates run: sudo apt update
```

Connet our ec2 instance from local using Bash/ssh

Move backend-sourcecode to ec2 similarly frontend

Install docker in in ec2

```

$ docker@ip-172-31-8-32:~$ docker
Usage: docker [OPTIONS] COMMAND [Args...]
      --help     Print this message or the help of one command.
      --version  Print the Docker version.
      -c        Create a container.
      -d        Detach from a running container.
      -f        Force removal of containers.
      -l        List logs of a container.
      -p        List port mappings or a specific mapping for the container.
      -r        Remove a container.
      -v        Inspect a container.
      -w        Remove all containers.
      -a        Save one or more images to a tar archive (streamed to STDOUT by default).
      -c        Start one or more stopped containers.
      -C        Create a live stream of container(s) resource usage statistics.
      -e        Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE.
      -t        Display the running processes of a container.
      -u        Increase all processes within one or more containers.
      -v        Update configuration of one or more containers.
      -w        Block until one or more containers stop, then print their exit codes.

Global Options:
  --config string   Location of client config files (default: "/home/ubuntu/.docker")
  -c --context string Name of the context to use to connect to the daemon (overrides DOCKER_HOST env var and default context set with "docker context use").
  -D --debug        Enable debug mode.
  -H --host string  Docker socket to connect to.
  -L --log-level string Set the logging level ("debug", "info", "warn", "error", "fatal") (default: "info")
  -l --log-opt string Log opt passed to the log driver.
  --tls            Trust certs signed only by this CA (default: "/home/ubuntu/.docker/ca.pem")
  --tlscacert string Path to TLS certificate file (default: "/home/ubuntu/.docker/ca.pem")
  --tlscert string Path to TLS certificate file (default: "/home/ubuntu/.docker/cert.pem")
  --tlscipher string TLS cipher to use for TLS (default: "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256")
  --tlsverify      Use TLS and verify the remote.
  -V --version     Print version information and quit.

Run 'docker COMMAND --help' for more information on a command.

For more help see https://docs.docker.com/guide/
abouthttps://172-31-8-32-/backend/runs/docker-build .
DEPRECATION: The https://builder is deprecated and will be removed in a future release.
  Test all the builds component to build images with BuildKit:
    https://docs.docker.com/guide/buildkit/

```

Sending build context to Docker daemon 18.01B
Step 0/7 : FROM python:3-alpine
Already exists
Step 1/7 : ADD . /
Step 2/7 : RUN pip install flask
Step 3/7 : ADD . /
Step 4/7 : RUN pip install gunicorn
Step 5/7 : EXPOSE 8000
Step 6/7 : CMD ["gunicorn", "-p", "5000", "app:app"]
Step 7/7 : LABEL \$Dockerfile
Successfully built cfc01126a768
Successfully tagged backend:latest
http://172.31.8.32-/backend/runs/docker-build/images
REPOSITORY TAG IMAGE ID CREATED SIZE
backend latest cfc01126a768 4 weeks ago 42MB
backend latest c7210213b0fa 3 weeks ago 110MB
python 3.6-alpine 34492f0135f 3 weeks ago 47.4MB
node 10-alpine ebb3bb1bb04 3 weeks ago 115MB
http://172.31.8.32-/backend/runs/docker-run -d \

Build docker image for frontend and backend both

```

$ docker@ip-172-31-8-32:~$ docker
---> Using cache
--> 8a44c7b930f9
tag: v1 : <none>:1
<none>:1 <none>
--> 7a303131a3
tag: v1 : entrypoint ["python"]
<none>:1 <none>
--> 7a303131a3
tag: v1 : CMD ["python", "app.py"]
<none>:1 <none>
--> cf2d5267e8
<none>:1 <none> cfc01126a768
<none>:1 <none> tagged backend:latest
http://172.31.8.32-/backend/runs/docker-build/images
REPOSITORY          TAG      IMAGE ID      CREATED          SIZE
backend            latest   cfc01126a768   4 weeks ago    42MB
backend            latest   c7210213b0fa   3 weeks ago    110MB
python             3.6-alpine  34492f0135f   3 weeks ago    47.4MB
node              10-alpine  ebb3bb1bb04   3 weeks ago    115MB
http://172.31.8.32-/backend/runs/docker-run -d \
--network app-network
--name frontend
--name backend
backend:latest
user: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock. Read "http://docs.docker.com/networking/docker-run-d/#check-socket-permissions" for more information.
http://172.31.8.32-/backend/runs/docker-run -d --name backend --network app-network --p 8000:8000 backend:latest
http://172.31.8.32-/backend/runs/docker-run -d --name frontend --network app-network --p 8000:8000 frontend:latest
user: Error response from daemon: Conflict. The container name "backend" is already in use by container "1e360a7298a354f1a847870f7837b3c1081a7b882a16374483179". You have to remove (or rename) a container to be able to reuse that name.

# Docker run --help For more information
http://172.31.8.32-/backend/runs/docker-run -d --name backend --network app-network --p 8000:8000 backend:latest
http://172.31.8.32-/backend/runs/docker-run -d --name frontend --network app-network --p 8000:8000 frontend:latest
user: Error response from daemon: Conflict. The container name "backend" is already in use by container "1e360a7298a354f1a847870f7837b3c1081a7b882a16374483179". You have to remove (or rename) a container to be able to reuse that name.

# Docker run --help For more information
http://172.31.8.32-/backend/runs/docker-run -d --name backend --network app-network --p 8000:8000 backend:latest
http://172.31.8.32-/backend/runs/docker-run -d --name frontend --network app-network --p 8000:8000 frontend:latest
http://172.31.8.32-/backend/runs/docker-run -d --name frontend
--name frontend
--network app-network
--p 8000:8000
frontend:latest
1e360a7298a354f1a847870f7837b3c1081a7b882a16374483179
http://172.31.8.32-/backend/runs/docker-run -d --name backend
backend:latest
1e360a7298a354f1a847870f7837b3c1081a7b882a16374483179
CONTAINER ID  IMAGE           COMMAND      CREATED          STATUS          PORTS          NAMES
1e360a7298a354f1a847870f7837b3c1081a7b882a16374483179  frontend:latest
1e360a7298a354f1a847870f7837b3c1081a7b882a16374483179  frontend:latest
1e360a7298a354f1a847870f7837b3c1081a7b882a16374483179  backend:latest

```

Both container run in same environment

Success: Inbound security group rules successfully modified on security group sg-00565eabaad638cb7 (Launch-wizard-2)

[Details](#)

sg-00565eabaad638cb7 - launch-wizard-2

Action

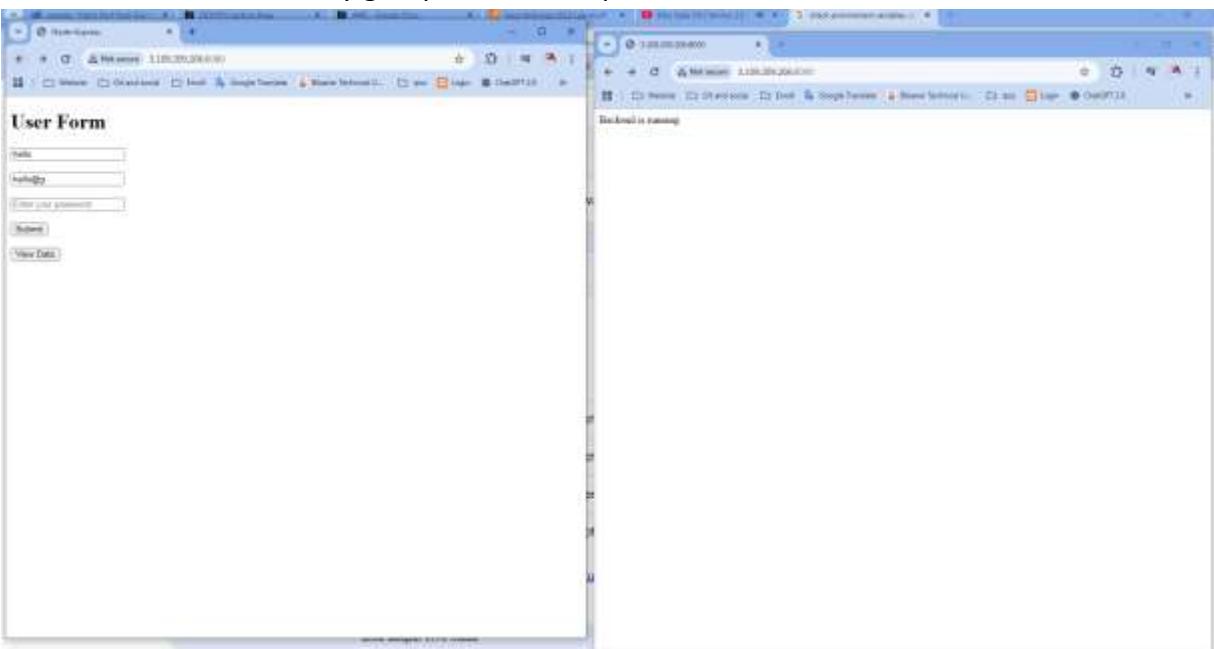
Security group name: Launch-wizard-2	Security group ID: sg-00565eabaad638cb7	Description: Launch-wizard-2 created 2024-01-09T14:55:50Z	VPC ID: vpc-0c18d43ea35e5ef2
Owner: 38274996244	Inbound rules count: 4 Permission entries	Outbound rules count: 1 Permission entry	

Inbound rules Outbound rules Sharing VPC associations Tags

Inbound rules (4)

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
-	sg-05c3ca025602aa76	IPv4	SSH	TCP	22	0.0.0.0/0	-
-	sg-03f53a1fbf2bc2fa	IPv4	Custom TCP	TCP	8000	0.0.0.0/0	-
-	sg-0d921f0007596f21	IPv4	Custom TCP	TCP	8000	0.0.0.0/0	-
-	sg-06c3da0ec5346694	IPv4	HTTP	TCP	80	0.0.0.0/0	-

Add Inbound rule to security group to receive request from outside/fromlocal



Both containers run successfully but on same public IP

2.

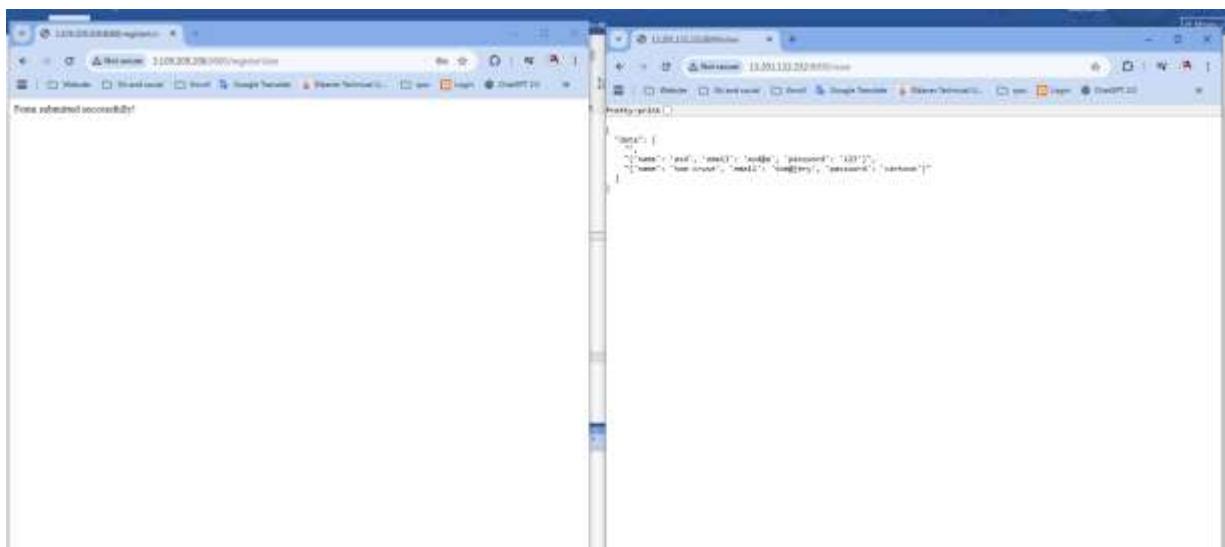
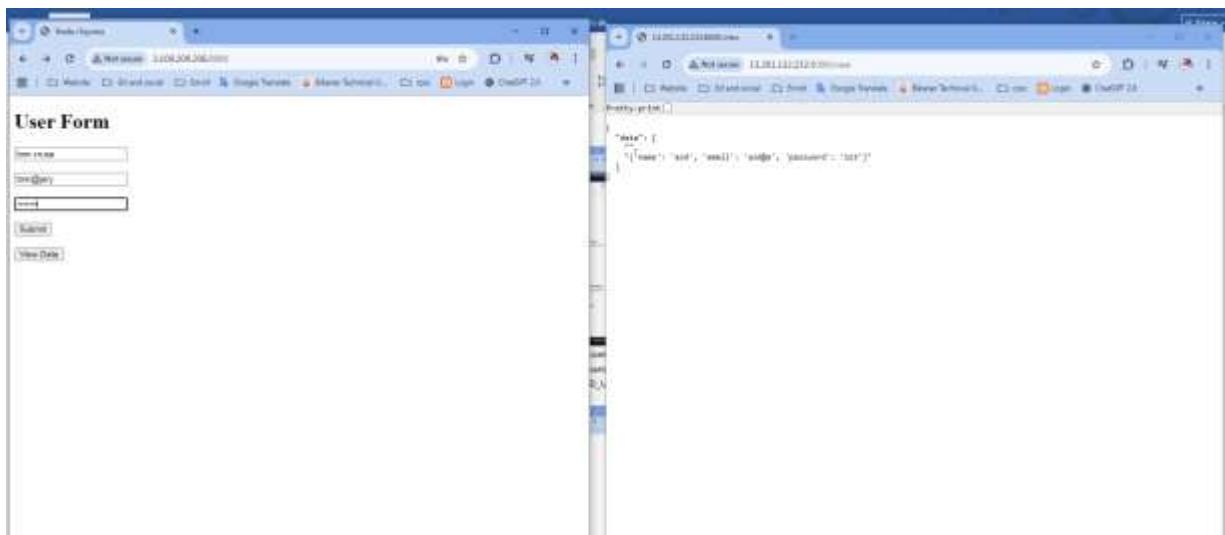
The left screenshot shows the 'Instance summary for i-095ba1ea5d75f79b (Frontend)'. It includes details like Instance ID (i-095ba1ea5d75f79b), Public IPv4 address (139.180.209.300), Private IPv4 address (172.31.18.82), and Auto Scaling Group (aws-ec2-112-81-82-as-with-1249424546). The right screenshot shows the 'Instance summary for i-088d294c79c3ff2cc (Backend)'. It includes details like Instance ID (i-088d294c79c3ff2cc), Public IPv4 address (139.180.211.191), Private IPv4 address (172.31.18.306), and Auto Scaling Group (aws-ec2-112-81-82-as-with-1249424546).

Two instances launched and run backend container after that run frontend container

and pass backend url as environment variable to frontend container

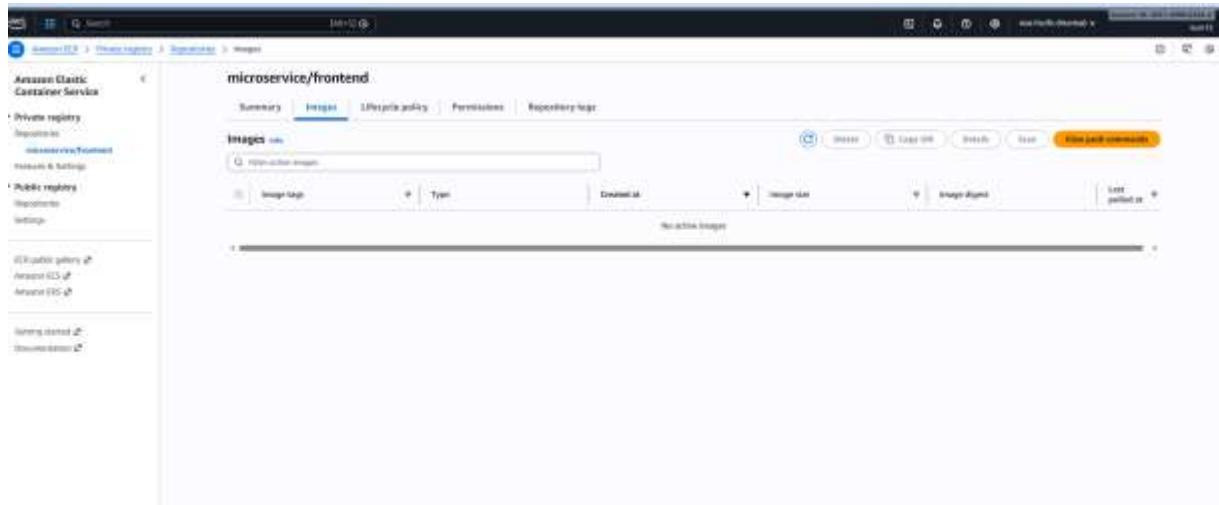
Ex: docker run -d -p 8080:8080 -e BACKEND_URL=http://13.201.132.232:8000 frontend:latest

The left screenshot shows a 'User Form' with fields for 'User your name', 'User your email', 'User your password', 'Submit', and 'View Test'. The right screenshot shows a terminal window with the command 'curl -X POST -H "Content-Type: application/json" -d '{"name": "John", "email": "john@example.com", "password": "123456"}' being typed.



Data successfully received and saved by backend.

3.



Create ECR reporisitory

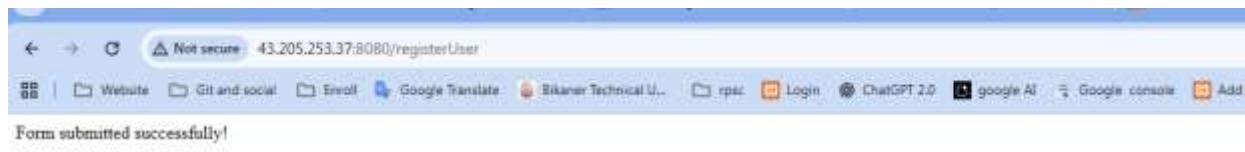
```
root@ip-10-10-10-10: ~ % cd microservice/ ; docker build -t microservice/frontend .
root@ip-10-10-10-10: ~ % docker push 382749986244.dkr.ecr.ap-south-1.amazonaws.com/microservice/frontend:latest
The push refers to repository [382749986244.dkr.ecr.ap-south-1.amazonaws.com/microservice/frontend]
25d53025e29: Pushed
c770cf104000: Pushed
c11fe4be148: Pushed
3be70093c259: Pushed
71b9a158f07: Pushed
c7e7bc519bc: Pushed [=====] 42.27MB/120.8MB
7bb2fcf5ef67: Pushed
```

Push frontend docker image to ECR & similarly backed

Create cluster for deploy service

both service created and both run succeaafuly

Frontend run fine



Form linked to backed properly