Create an Ec2 instance to avoid architecture difference in docker and create a simple dockerfile

```
FROM public.ecr.aws/lambda/python:3.8

RUN yum install -y gcc-c++ python3-devel
RUN pip install torch torchvision
```

EC2 amazon linux machine has an slightly irritating issue of adding sudo to commands.

You have to install docker in the ec2 instance.

```
sudo yum update -y
sudo yum install -y docker
sudo service docker start (start the docker daemon)
sudo usermod -a -G docker ec2-user
docker run hello-world (verify installation)
```

Torch and torchvision have issues with macos so don't try to run the above dockerfile in macos system.
Related

```
sudo docker build -t shukhapriya .
```

Creates a docker image named shukhapriya run this command having the dockerfile directory

```
`docker images`
```

verify shukhapriya is present in output

```
docker run -p 9000:8080 hello-world and curl -XPOST
"http://localhost:9000/2015-03-31/functions/function/invocations" -d '{}'
```

to check whether lambda function working expectedly or not. Note the output would be in another terminal.

Run `sudo aws configure` and verify/add the aws credentials. Note that we are using sudo here and not simply aws configure which would create a problem, please check .

```
sudo chmod 666 /var/run/docker.sock
```

to prevent some docker error again mentioned in above link

```
sudo aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin 714116641665.dkr.ecr.us-east-1.amazonaws.com
```

to connect to ecr.

Create a ECR repo

```
aws ecr create-repository --repository-name ml-libraries --image-scanning-
configuration scanOnPush=true
```

```
docker tag shukhapriya 714116641665.dkr.ecr.us-east-1.amazonaws.com/ml-
libraries:latest
```

tag the image with the repo uri. You would be getting the repo uri in the 6th command output as "repositoryUri": "714116641665.dkr.ecr.us-east-1.amazonaws.com/ml-libraries". Add a new tag instead of latest such as ‘torch-cpu’ etc.

Simply push it now

```
docker push 714116641665.dkr.ecr.us-east-1.amazonaws.com/ml-
libraries:latest .
```

Pushing into an repo that has a similar image takes very little time (less than 3 seconds for me)

Some helpful commands:

1. docker images # Lists all images
2. docker ps # All running containers
3. docker ps -a # All containers including stopped
4. docker rmi images # Removes an image which does not have any container linked to it
5. docker container prune # Kills containers not being used
6. docker kill container-id
7. docker image prune # Kills ideal images
8. docker exec -it container-id bash # We go into a bash terminal inside the container
9. docker run image-id # Starts a container, use docker ps to get container id

You can do step 1, 8 and 9 to create a new container and adding to same old repo, ensure you change the name ‘latest’ to something else.

Some errors / tips faced:

- Use psycopg2-binary instead of psycopg while pip installing in dockerfile.
- Ensure you copy the files after pip installing inside the dockerfile to use the cache.
- AWS Lambda does not commit transactions with sqlalchemy so ensure you commit in code.
- Any models, images store in a directory, it causes errors otherwise COPY model/ ${LAMBDA_TASK_ROOT}/model/
- encoded_string = event['body'] the handler takes two parameters body and context. So even during testing of lambda function in cosole, the test should be something like {“body”: “image_base64_string”, “para2”:””, para3:””.....}
- Ensure the lambda function output is in json format so that the api will run successfully. Otherwise the lambda function when triggered will run correctly, inserting into DB but api shows error.

```
def handler(event, context): # your code here

    response = {
        "statusCode": 200,
        "headers": {
            "Content-Type": "text/plain"
        },
        "body": "Successful"
    }

    return response
```

You cannot change the lambda function once created so before careful and make any changes before!