# Deployment in docker

### Setup

## Setup aws client

```
pip install awscli
aws configure (y pones tus credenciales)
```

## Set environmental variables

On the following steps change the $variables with your own image name and region

```
img_name='dd-zero-shot-text-classification'
aws_region=us-west-2
aws_account_id=401913772240
```

## Create Dockerfile

Should look something like this.

Notes:

- Create the file without extension in something like VS Code.
- On the last part CMD – is handler_filename.function.

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

# Copy function code and models into our /var/task
COPY ./ ${LAMBDA_TASK_ROOT}/

# install our dependencies
RUN python3 -m pip install -r requirements.txt --target
${LAMBDA_TASK_ROOT}

# Set the CMD to your handler (could also be done as a parameter
override outside of the Dockerfile)
CMD [ "handler.handler" ]
```

For the case of python levenshtein is needed or gcc package is needed, image taken from aws public repository is linux-aws based so you need to use yum.

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

# Copy function code and models into our /var/task
COPY ./ ${LAMBDA_TASK_ROOT}/

# Use gcc and install dependencies
RUN rm -rf /var/cache/yum \
  && yum install -y python3-devel gcc gcc-c++ make \
  && python3 -m pip install -r requirements.txt --target
${LAMBDA_TASK_ROOT}

# Remove

# Set the CMD to your handler (could also be done as a parameter
override outside of the Dockerfile)
CMD [ "handler.runtask" ]
```

### Verify Folder structure

Should look something like this:

```
.
 Dockerfile
 README.md
 __init__.py
 handler.py
 requirements.txt
 serverless.yml
```

#### Deploy

### Build the image

```
docker build -t $img_name .
```

There is a point at the end you should put

### Test localy (optional)

Run the image:

```
docker run -p 8080:8080 $img_name
```

And in another terminal:

```
curl --request POST \
  --url <http://localhost:8080/2015-03-31/functions/function
/invocations> \
  --header 'Content-Type: application/json' \
  --data '{"body":"{\"text\":\"La UNAM es una universidad mexicana
que...\", \"labels\":[\"Insitución educativa\", \"Insitución
financiera\"]}"}'
```

### Create ecr repository

```
aws ecr create-repository --region $aws_region --repository-name
$img_name > /dev/null
```

### Login to aws ecr

```
aws ecr get-login-password --region $aws_region | docker login --
username AWS --password-stdin $aws_account_id.dkr.ecr.$aws_region.
amazonaws.com
```

### Tag the image

```
docker tag $img_name $aws_account_id.dkr.ecr.$aws_region.amazonaws.com
/$img_name
```

### Push the image

```
docker push $aws_account_id.dkr.ecr.$aws_region.amazonaws.com/$img_name
```

#### Deploy lambda

1. Copy the sha digest return by the push, put it in the serverless yml and deploy de image

```
serverless deploy
```

The serverless should look something like this

```
service: tu-service

provider:
  name: aws
  region: us-west-2
  memorySize: 5120
  timeout: 60
  stage: dev
  role: arn:aws:iam::401913772240:role/lambda-gateway-execution-role
  deploymentBucket:
    name: dive-serverless-deployments
    deploymentPrefix: tu-service

functions:
  classifier:
    image: 401913772240.dkr.ecr.us-west-2.amazonaws.com/dd-zero-shot-
text-classification@sha256:
df1d0cce9f4968738dfe35868abc091b7427eb998e59c4374568910448abb001 #
Cambiar esto por el nombre de tu imagen @ el sha que te regresa el push
```