Lab 8: Serverless

In this lab, we'll deploy the dogs vs cats model we trained in the previous lab.

Download the model from here:

https://github.com/fenago/large-datasets/releases/download/dogs-cats-model/dogs_cats_10_0.687.h5

Question 1

Now convert this model from Keras to TF-Lite format.

What's the size of the converted model?

Question 2

To be able to use this model, we need to know the index of the input and the index of the output.

What's the output index for this model?

Preparing the image

You'll need some code for downloading and resizing images. You can use this code:

```python
from io import BytesIO
from urllib import request

from PIL import Image
```

```
def download_image(url):
    with request.urlopen(url) as resp:
        buffer = resp.read()
    stream = BytesIO(buffer)
    img = Image.open(stream)
    return img


def prepare_image(img, target_size):
    if img.mode != 'RGB':
        img = img.convert('RGB')
    img = img.resize(target_size, Image.NEAREST)
    return img
```

For that, you'll need to have pillow installed:

```bash
pip install pillow
```

Let's download and resize this image:

https://upload.wikimedia.org/wikipedia/commons/9/9a/Pug_600.jpg

Based on [the solution of the previous lab](https://github.com/alexeygrigorev/mlbookcamp-code/blob/master/course-zoomcamp/08-deep-learning/CNN_solution.ipynb),
what should be the target size for the image?

Question 3

Now we need to turn the image into an numpy array and pre-process it.

> Tip: Check the previous lab. What was the pre-processing
> we did there?

After the pre-processing, what's the value in the first pixel, the R channel?

Question 4

Now let's apply this model to this image. What's the output of the model?

Prepepare the lambda code

Now you need to copy all the code into a separate python file. You will need to use this file for the next two questions.

Tip: you can test this file locally with `ipython` or Jupyter Notebook by importing the file and invoking the function from this file.

Docker

For the next two questions, we'll use a Docker image that I already prepared. This is the Dockerfile that I used for creating the image:

```docker
FROM public.ecr.aws/lambda/python:3.8
COPY cats-dogs-v2.tflite .
```

And pushed it to [`agrigorev/zoomcamp-cats-dogs-lambda:v2`](https://hub.docker.com/r/fenago/zoomcamp-cats-dogs-lambda/tags).

> Note: The image already contains a model and it's not the same model
> as the one we used for questions 1-4.

Question 5

Now let's extend this docker image, install all the required libraries and add the code for lambda.

You don't need to include the model in the image. It's already included. The name of the file with the model is `cats-dogs-v2.tflite` and it's in the current workdir in the image (see the Dockerfile above for the reference).

What's the image id of the base image?

In the build logs (on Linux), you'll see a log like that:

```
$ docker some-command-for-building
Sending build context to Docker daemon  2.048kB
Step 1/N : FROM agrigorev/zoomcamp-cats-dogs-lambda:v2
 ---> XXXXXXXXXXXX
Step 2/N : ....
```

You need to get this `XXXXXXXXXXXX`.

On MacOS and Windows, the logs for `docker build` are different. To get the image id there, you can use `docker image ls -a`.

Question 6

Now run the container locally.

Score this image: https://upload.wikimedia.org/wikipedia/commons/1/18/Vombatus_ursinus_-Maria_Island_National_Park.jpg

What's the output from the model?

It's possible that your answers won't match exactly. If it's the case, select the closest one.

Publishing it to AWS

Now you can deploy your model to AWS!

* Publish your image to ECR
* Create a lambda function in AWS, use the ECR image
* Give it more RAM and increase the timeout
* Test it
* Expose the lambda function using API Gateway

Publishing to Docker hub

This is just for reference, this is how I published an image to Docker hub:

```bash
docker build -t cats-dogs-lambda .
docker tag cats-dogs-lambda:latest agrigorev/zoomcamp-cats-dogs-lambda:v2
docker push fenago/zoomcamp-cats-dogs-lambda:v2
```