Lab 5 - Deploying a container in to Docker

So now we have a Python model and we know how to run a Docker container.

What we have not seen is how to move from nothing to a container. So lets start by doing that.

Step 1

1. Create a new folder in your development area called cats
   (you will find out why shortly - trust me...)
2. In that folder create a new file called Dockerfile (no extention, just Dockerfile.)
3. Copy the following in to that Dockerfile

```
# Python on Build
FROM python:3-onbuild

# tell the port number the container should expose
EXPOSE 5000

# run the command
CMD ["python", "./app.py"]
```

4. Save the Dockerfile.

Note that we are doing a few things here.

1. We are pulling from python:3-onbuild this is an image which is always inline with the latest 3 image. But is also does something interesting. It is expects to find a file called requirements.txt We will add this is a moment
2. We are exposing port 5000 - This is the port our container will run in
3. we are telling the container to run a python script called app.py

Step 2

We need some code to run on our container. Let's populate app.py

5. Create a new file called app.py. Add the following:

```
from flask import Flask, render_template
import random

app = Flask(__name__)
```

Lab 5 - Deploying a container in to Docker

```
# list of cat images
images = [
    "https://firebasestorage.googleapis.com/v0/b/docker-
curriculum.appspot.com/o/catnip%2F0.gif?alt=media&token=0fff4b31-b3d8-44fb-be39-
723f040e57fb",
    "https://firebasestorage.googleapis.com/v0/b/docker-
curriculum.appspot.com/o/catnip%2F1.gif?alt=media&token=2328c855-572f-4a10-af8c-
23a6e1db574c",
    "https://firebasestorage.googleapis.com/v0/b/docker-
curriculum.appspot.com/o/catnip%2F10.gif?alt=media&token=647fd422-c8d1-4879-af3e-
fea695da79b2",
    "https://firebasestorage.googleapis.com/v0/b/docker-
curriculum.appspot.com/o/catnip%2F11.gif?alt=media&token=900cce1f-55c0-4e02-80c6-
ee587d1e9b6e",
    "https://firebasestorage.googleapis.com/v0/b/docker-
curriculum.appspot.com/o/catnip%2F2.gif?alt=media&token=8a108bd4-8dfc-4dbc-9b8c-
0db0e626f65b",
    "https://firebasestorage.googleapis.com/v0/b/docker-
curriculum.appspot.com/o/catnip%2F3.gif?alt=media&token=4e270d85-0be3-4048-99bd-
696ece8070ea",
    "https://firebasestorage.googleapis.com/v0/b/docker-
curriculum.appspot.com/o/catnip%2F4.gif?alt=media&token=e7daf297-e615-4dfc-aa19-
bee959204774",
    "https://firebasestorage.googleapis.com/v0/b/docker-
curriculum.appspot.com/o/catnip%2F5.gif?alt=media&token=a8e472e6-94da-45f9-aab8-
d51ec499e5ed",
    "https://firebasestorage.googleapis.com/v0/b/docker-
curriculum.appspot.com/o/catnip%2F7.gif?alt=media&token=9e449089-9f94-4002-a92a-
3e44c6bd18a9",
    "https://firebasestorage.googleapis.com/v0/b/docker-
curriculum.appspot.com/o/catnip%2F8.gif?alt=media&token=80a48714-7aaa-45fa-a36b-
a7653dc3292b",
    "https://firebasestorage.googleapis.com/v0/b/docker-
curriculum.appspot.com/o/catnip%2F9.gif?alt=media&token=a57a1c71-a8af-4170-8fee-
bfe11809f0b3",
]

@app.route('/')
def index():
    url = random.choice(images)
    return render_template('index.html', url=url)

if __name__ == "__main__":
    app.run(host="0.0.0.0")
```

I won't explain what this is doing in much detail here. We will look at it more when we containerise our model.

Step 3

We need a requirements.txt.

6. Create a file called requirements.txt

Lab 5 - Deploying a container in to Docker

7. add the following:

```
Flask==0.10.1
```

Step 4

8. Add a new folder called templates
9. Create a new file in there called index.html
10. Add the following code:

```
<html>
  <head>
    <style type="text/css">
      body {
        background: black;
        color: white;
      }
      div.container {
        max-width: 500px;
        margin: 100px auto;
        border: 20px solid white;
        padding: 10px;
        text-align: center;
      }
      h4 {
        text-transform: uppercase;
      }
    </style>
  </head>
  <body>
    <div class="container">
      <h4>Cat Gif of the day</h4>
      <img src="{{url}}" />
      <p><small>Courtesy: <a href="http://www.buzzfeed.com/copyranter/the-best-
cat-gif-post-in-the-history-of-cat-gifs">Buzzfeed</a></small></p>
    </div>
  </body>
</html>
```

What we have now is a Flask operating as a webserver and index.html acting as the front of our application.

Step 5

Let turn this is to a working application.

We need to change the location of our command line to the root of our docker folder.

Lab 5 - Deploying a container in to Docker

For me that looks like this

```
G:

cd G:\GitHubProjects\workshop-ModelManagement\MachineLearningFromModelToProduction\Docker\ManualImageCreationCats
```

11. Do the same for where you created your folder.

Now we are on the root we can build the container.

12. Run the following

```
docker build -t cats .
```

Meow Our cats are now building. You should see that the base image has been pulled and PIP has installed all the libs we need.

Now we need to run the image and see what it does.

13. Run the following command in the command prompt

```
docker run -p 5000:5000 cats
```

That should now be running our website. You can navigate to the following to see the outcome

14. Open a web browser and go to the following address.

```
localhost:5000
```

Dance kitties, dance!

15. You can now add web developer to your CV.

We have created a Python website and containerised it. From there we have deployed it to Docker. Docker has created a port funnel on 5000 from our localhost right through. That was pretty cool.

Let's stop that from running

16. Run the following command in the command prompt

```
docker stop cats
```

Lab 5 - Deploying a container in to Docker

I imagine you are thinking this lab is broken. He got us to do something that does not work. Well that is because an image can run multiple times. If you do not specify a name for your instance it will generate you one.

17. Run docker ps you can see all the images running.

```
docker ps
```

You will see that the image of cats running will have been run with a generated name. In my example that name is "serene_hoover" :P

18. Run the following command in the command prompt

```
docker stop serene_hoover
```

That will kill that running version. To add a name we can control we will need to add another parameter.

19. Run the following command in the command prompt

```
docker run --name kittens -p 5000:5000 cats
```

Ok this is now running under a named image

20. Run the following command in the command prompt

```
docker stop kittens
```

Lab completed