# Lab 06 - Deploying your model

The last lab was a bit of a tangent, but it was to show some of the quirks of working with Docker.

Now that we have a good grasp on the basics, let's do the same for our model.

## Step 1

In the last set of labs, we wrapped our model up in a Flask script. Now we have an API we can ping our model and get a response. What we don't want to do now is have to run the model each time manually. Let's put it in a docker container.

Have a go at doing what you think you need to base on the last lab.

Remember the general steps are:

1. Create a dockerfile
2. Create the application code
3. Create a requirements.txt file
4. Build the image
5. Check that the model is running

If you do get stuck here is the answer:

# Lab 06 - Deploying your model

You cheat! I am just kidding. It is complicated.

Let's walk through the solution end-to-end.

## Step 1

1. Create the following files:

```
Dockerfile
DiabetesModelFlask.py
requirements.txt
```

2. In the Dockerfile we will want to add the following:

```
FROM python:3-onbuild
COPY . usr/src/app
CMD ["python", "DiabetesModelFlask.py"]
```

3. In DiabetesModelFlask.py we want to add the following:

```
# Product Service
from flask import Flask, request, jsonify
from flask_restful import Resource, Api
import _pickle as pickle
import numpy as np
from sklearn import datasets, linear_model

app = Flask(__name__)
api = Api(app)

PickleModelPath = './regression.pkl'

@app.route('/score')
def apicall():
    var1 = float(request.args.get('var1'))
    with open(PickleModelPath, 'rb') as k:
        PickleModel = pickle.load(k)
    Answer = PickleModel.predict(var1)
    return jsonify(Answer.tolist())


@app.route('/train')
def apicalled():
    # Load the diabetes dataset
    diabetes = datasets.load_diabetes()
    # Use only one feature
    diabetes_X = diabetes.data[:, np.newaxis, 2]
    # Split the data into training/testing sets
    diabetes_X_train = diabetes_X[:-20]
    diabetes_X_test = diabetes_X[-20:]
    # Split the targets into training/testing sets
    diabetes_y_train = diabetes.target[:-20]
    diabetes_y_test = diabetes.target[-20:]
```

## Lab 06 - Deploying your model

```
    # Create linear regression object
    regr = linear_model.LinearRegression()
    # Train the model using the training sets
    regr.fit(diabetes_X_train, diabetes_y_train)
    # Make predictions using the testing set
    diabetes_y_pred = regr.predict(diabetes_X_test)
    with open(PickleModelPath, 'wb') as f:
            pickle.dump(regr, f)
    return "Model hase been retrained. Run /score to score model"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5071, debug=True)
```

Now our requirements are slightly more difficult. We have a lot more modules which need to be added.

4. Add the following to requirements.txt

```
pandas==0.18.1
scikit-learn==0.19.1
numpy==1.15.4
scipy==0.18.1
Flask==0.12
flask_restful==0.3.5
```

### Step 2

5. Navigate to the location of your docker file.

For me that looks like this

```
G:

cd G:\GitHubProjects\workshop-
ModelManagement\MachineLearningFromModelToProduction\Docker\DiabetesProductionDock
er
```

6. Build our model in to an image

```
docker build -t productiondiabetes .
```

This will be build our image.

7. Run our image as a container

```
docker run -d --name productiondiabetes -p 5071:5071 productiondiabetes
```

8. Navigate to localhost:5071

   You will see an error. That is because part of our web service is missing. If you train the model first, you will be able to execute the model.

## Lab 06 - Deploying your model

9. Navigate to the following url

```
localhost:5071/train?var1=1
localhost:5071/score?var1=1
```

Press ctrl+c to end the session.

**Lab done.**