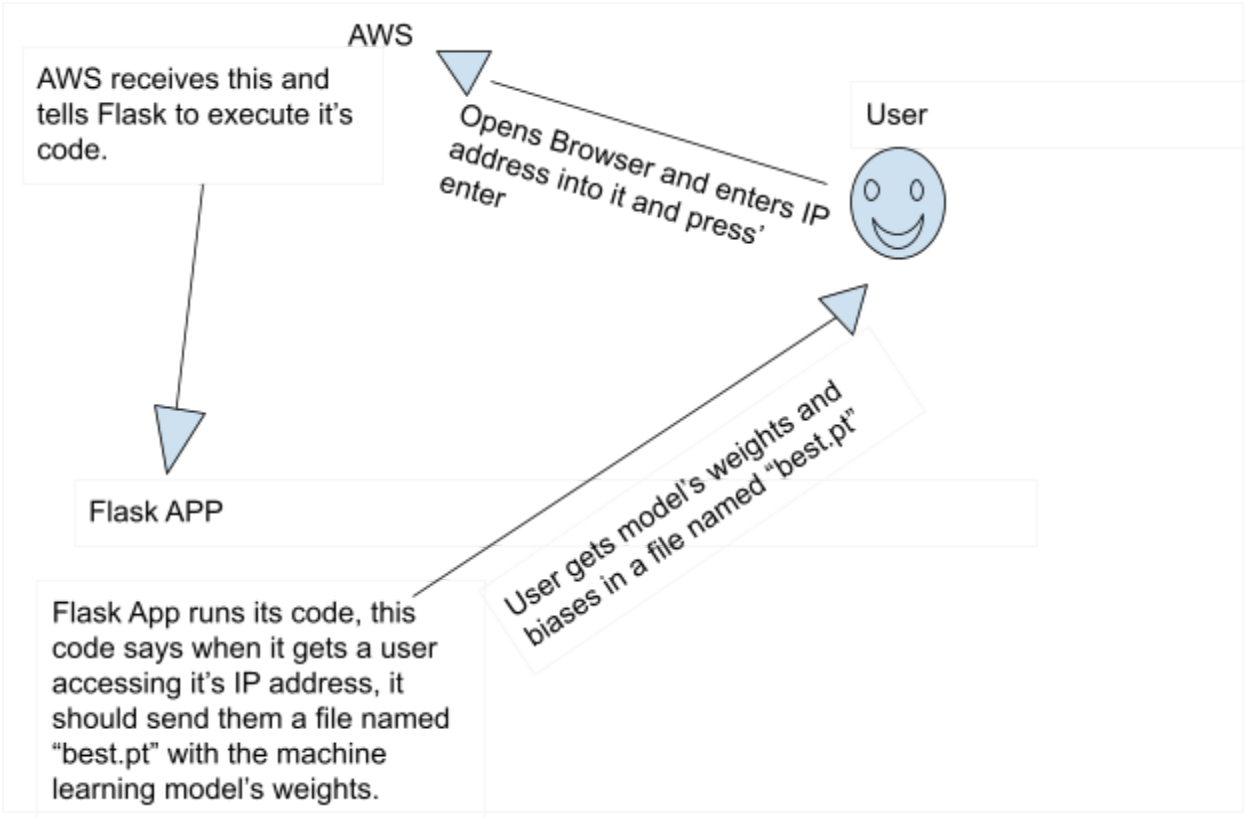
Capstone Project

The final project is an API that delivers the weights and biases of a neural network ready to be loaded into a model. The weights and biases are attuned to object detection and classification of playing cards.

The API can be accessed by pasting the IP address `100.25.85.14` into a browser. This will initiate the Flask App and return the weights and biases of the model.

Here is the sketch with the deployment method outlined:



The decision to simply give the user the weights and biases streamlines the deployment cycle. It also minimizes costs and computation. The cost of running inference is off-loaded to the end user. They have to load the model with the weights and biases then execute the code. The only costs are hosting the API, which are minimal, and data transfer, which is minimal as well. This will cost less than $50 a month. The only remaining cost that is uncapped is retraining and refinement. The model can be updated at anytime and the API can reflect that. As, the API can easily get a new model to output to users. The retraining costs could be a few hundred to a couple thousand depending on the amount of data and methods involved. This should not be a major issue as the model has high performance and doesn't need to be refined.

Code to take "best.pt" and put it into a model and get to predictions

```
import numpy as np
import pandas as pd
import torch
import os
import cv2
from matplotlib import pyplot as plt
%matplotlib inline

# 1.

# open a browser and paste 100.25.85.14 , this will download the weights of the neural network in a file named
# best.pt. Save that file somewhere on your computer.

# 2.
# Type the following code and execute it. The path = parameter should have the path where best.pt is saved.

model = torch.hub.load("ultralytics/yolov5", "custom", path="path/to/where/best.pt is on your system", force_reload=True)

# 3.

# Define a path to an image with a card in it. For reference, the images were 416 by 416 pixels.
# Something like this.
img = os.path.join("yolov5", "data", "playing_card_dataset_object_detection", "test", "images", "199424464_jpg.rf.ab6dec831e4c29a3d9f220f82674eb15.jpg")

# 4.

# Pass the image into the model for classification
results = model(img)

# 5.

# Outputting "results" will give you information about that prediction.

results

# 6.

# To get the desired image with bounding boxes and classification,
# run code similar to the below,
plt.imshow(np.squeeze(results.render()))
```

```
plt.show()

# Congratulations, you have classified an image.
```