# Model Deployment

with Python and Docker

Greg Johnson (GoGuardian)

# Please Run

---

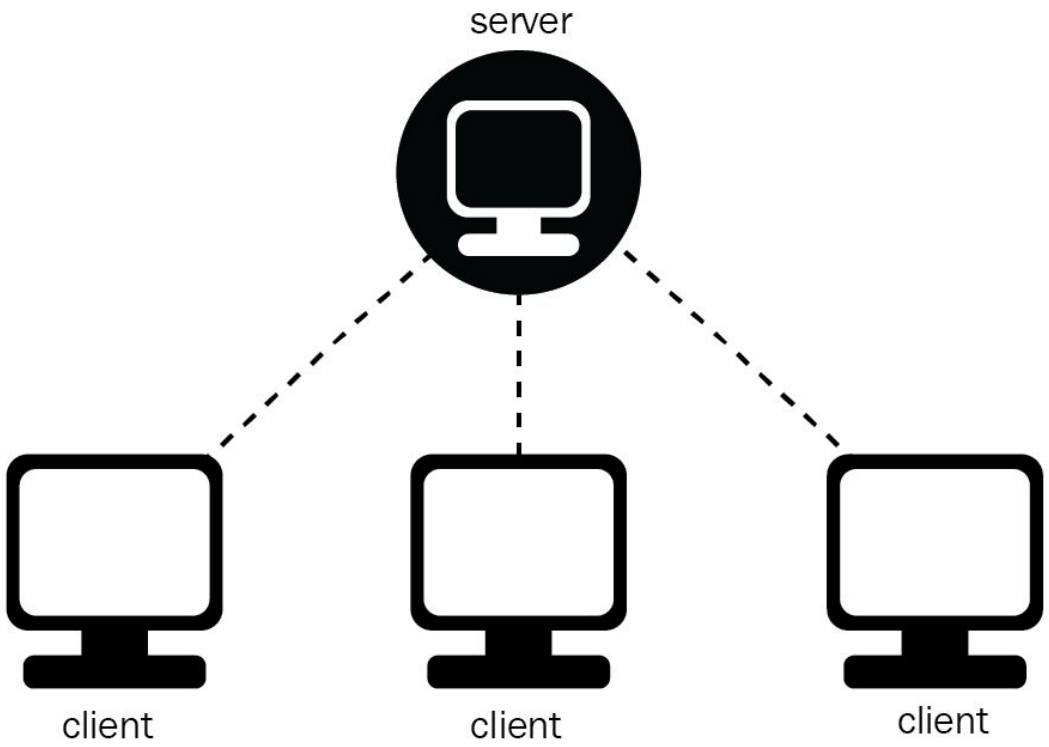
```
docker build -t model_server deployWithDocker/docker/
```

# What are we doing here?

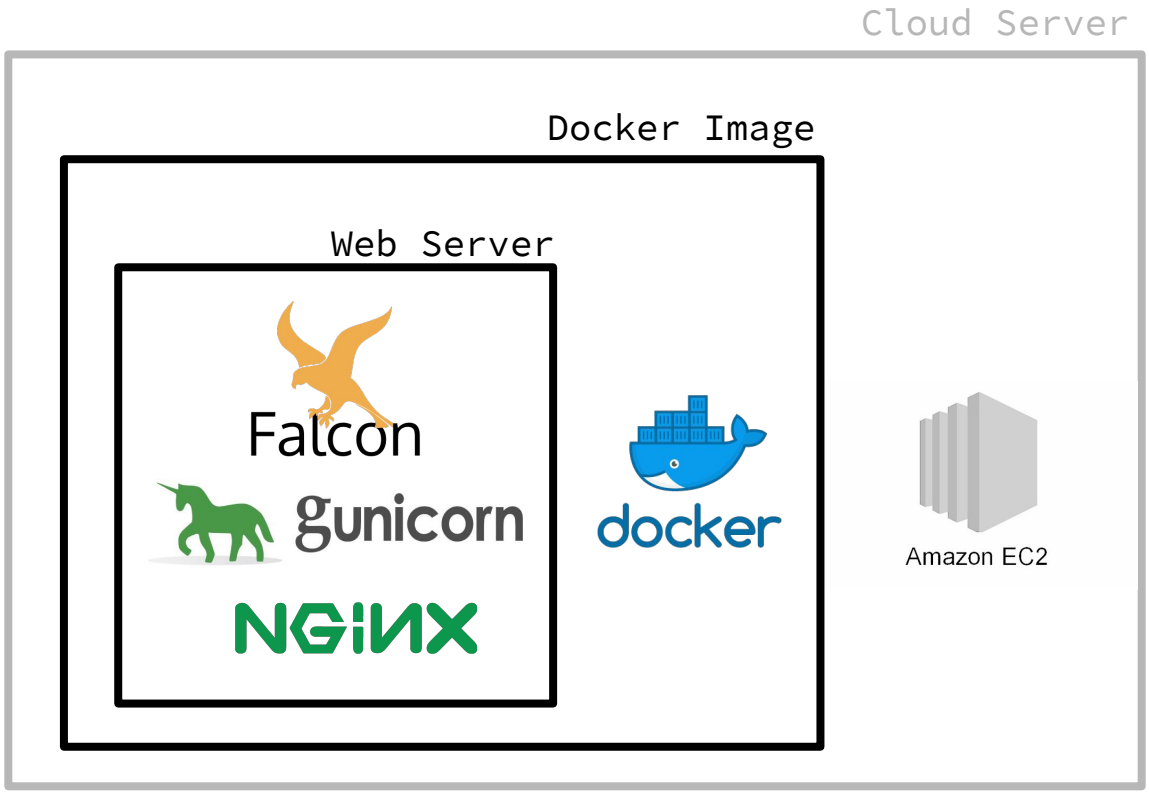
---

- I created this rad machine learning model in Python.
- How do I actually put it to work?

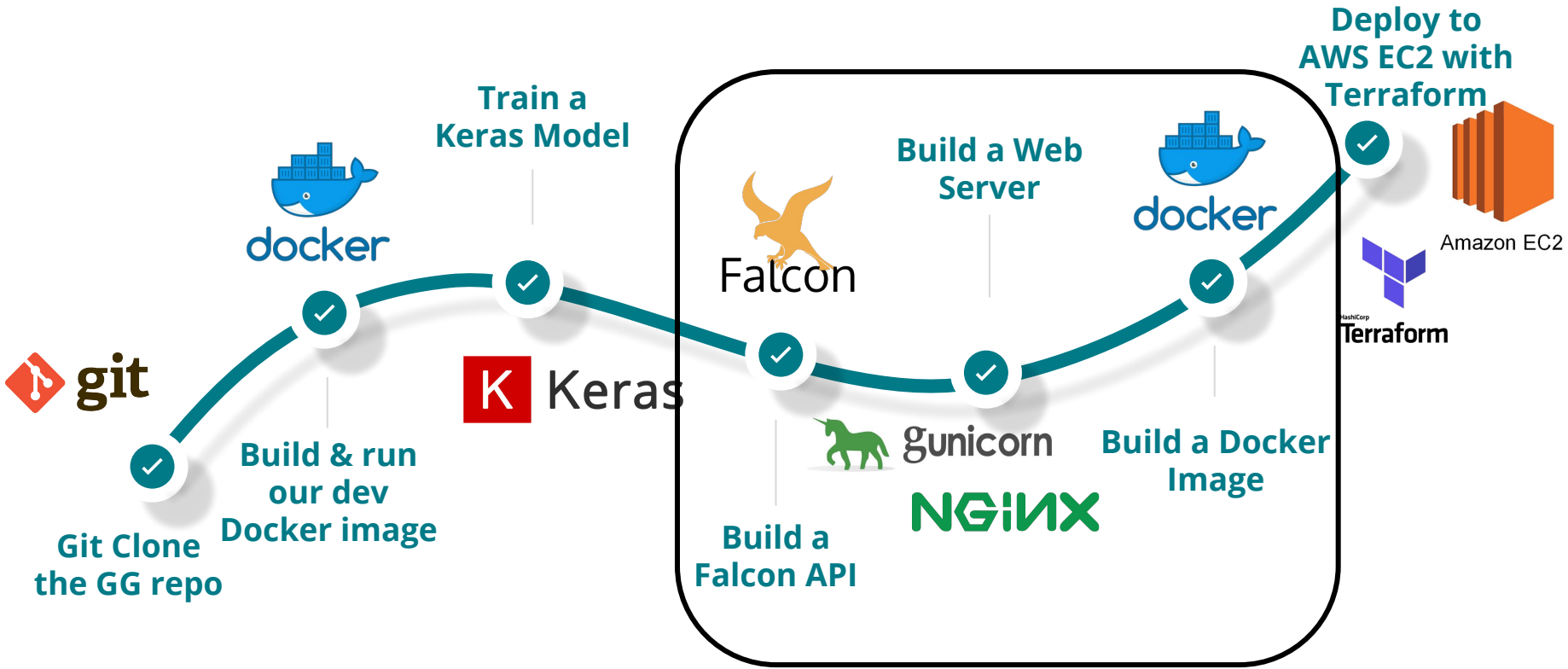
# How will our model communicate?



# What are we going to build?



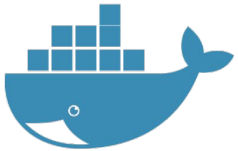
# What will we do today?



# What do I need?

---

**Docker** Community Edition

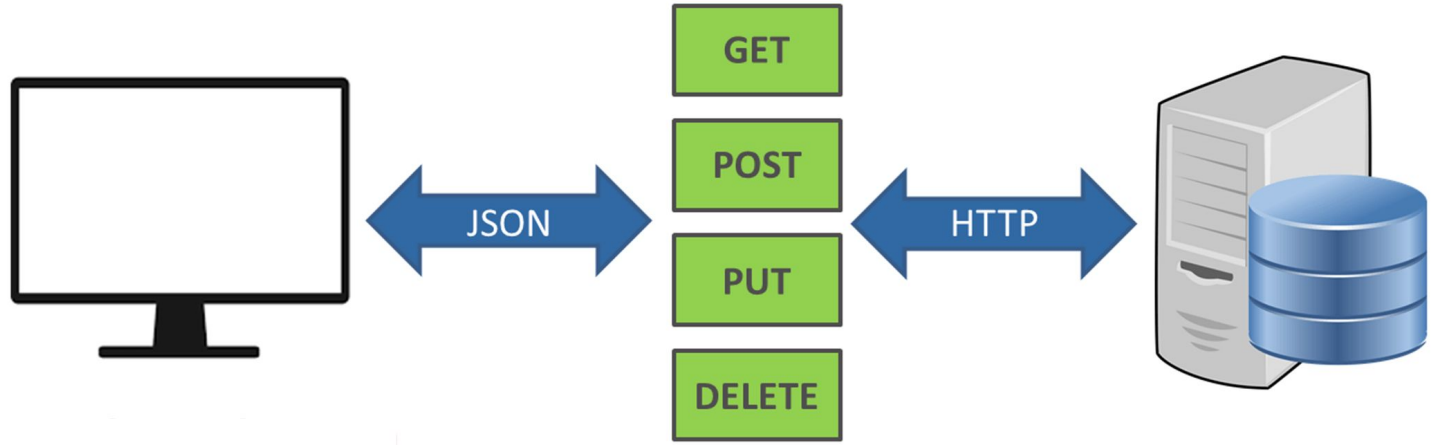


Install on:

- **Mac**: https://docs.docker.com/docker-for-mac/install/
- **Windows**: https://docs.docker.com/docker-for-windows/install/
- **Ubuntu**: https://docs.docker.com/install/linux/docker-ce/ubuntu/

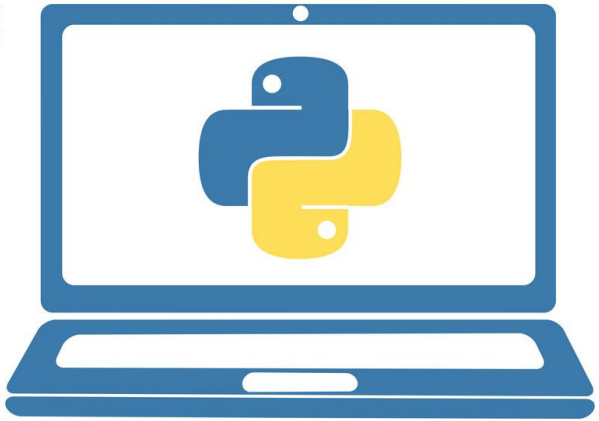
# APIs

# REST APIs




```
GET library.com/book
```

# Why Python?



# Python Framework Overload

# Building an API with 

---

1. Define prediction resource
2. Define the post method on the resource
3. Assign an endpoint to the resource

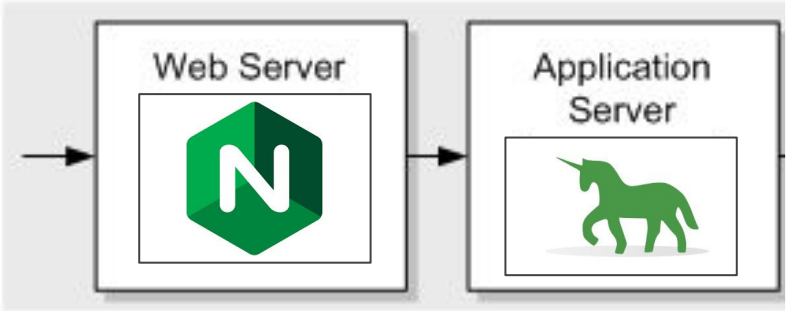
POST */predict*

# Our API:

___ cat deployWithDocker/docker/model_server.py

```
class PredictResource():

    def __init__(self, model):
        self.model = model

    def on_post(self, req, resp):
        image = req.get_param('image')
        data = preprocess(image)
        predicted_data = self.model.predict_classes(data)[0]

        output = {'prediction': str(predicted_data)}
        resp.status = falcon.HTTP_200
        resp.body = json.dumps(output)

mnistModel = load_model('model.h5')
predictResource = PredictResource(mnistModel)

api = falcon.API()
api.add_route('/predict', predictResource)
```

# Three-Tier Architecture



# Nginx

---

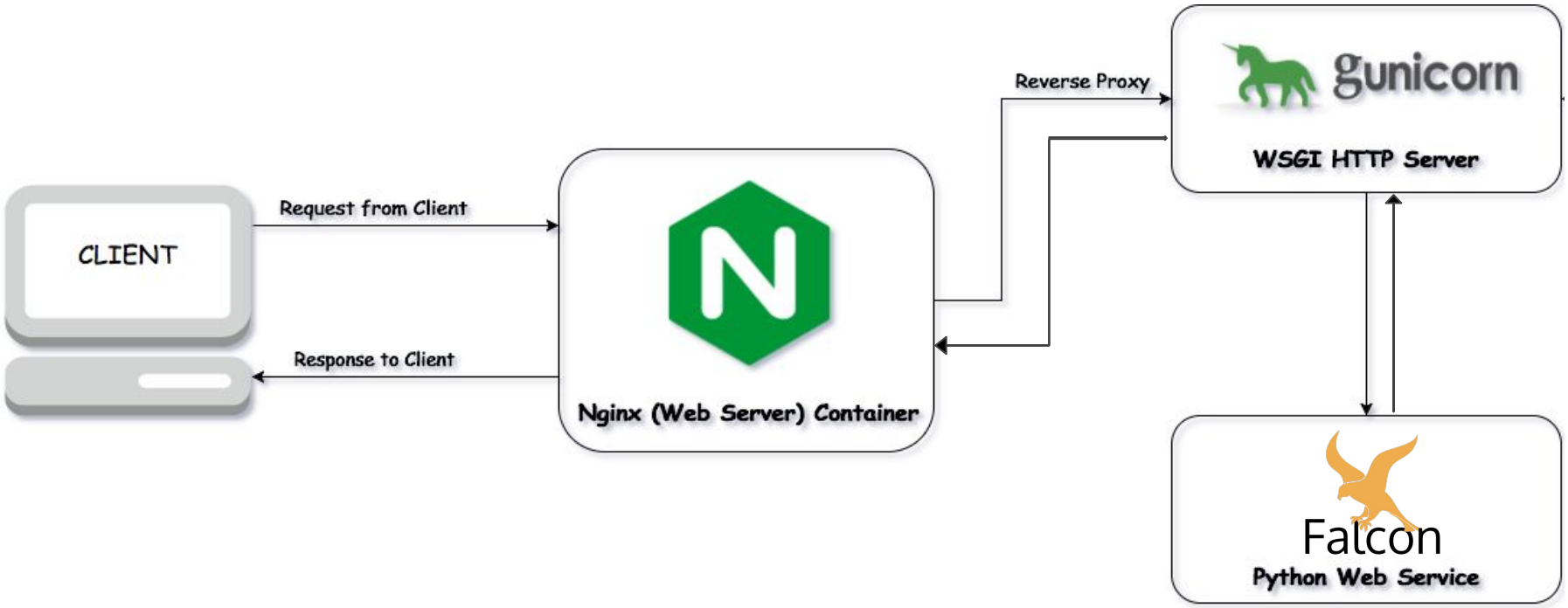
- Address Apache limitations
- Deals with messy real-world requests
- **Awesome** combo with a pre-forking server
- Serves static files
- Protects against DoS attacks

# Gunicorn

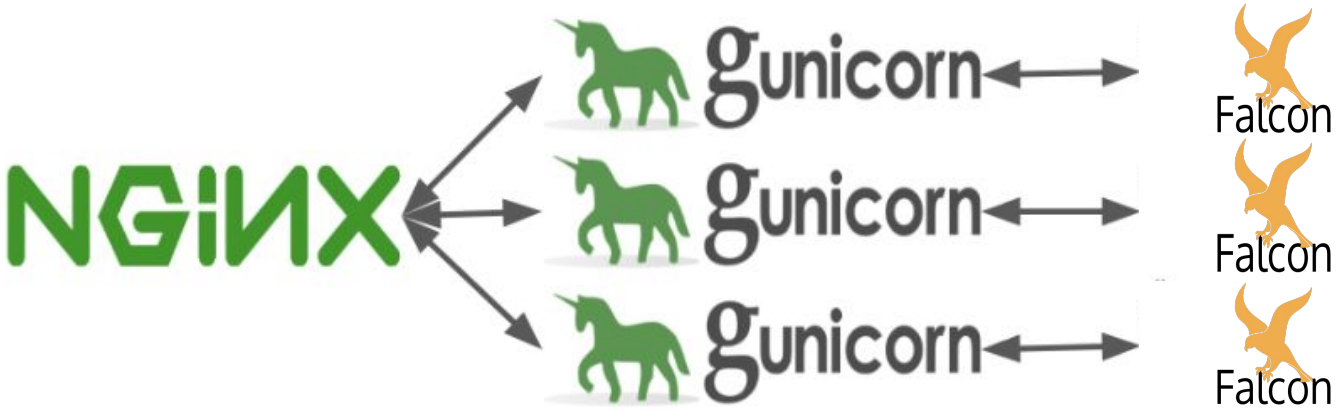
---

- Simple request processing
- Calls our API code
- WSGI translator
- Pre-fork server

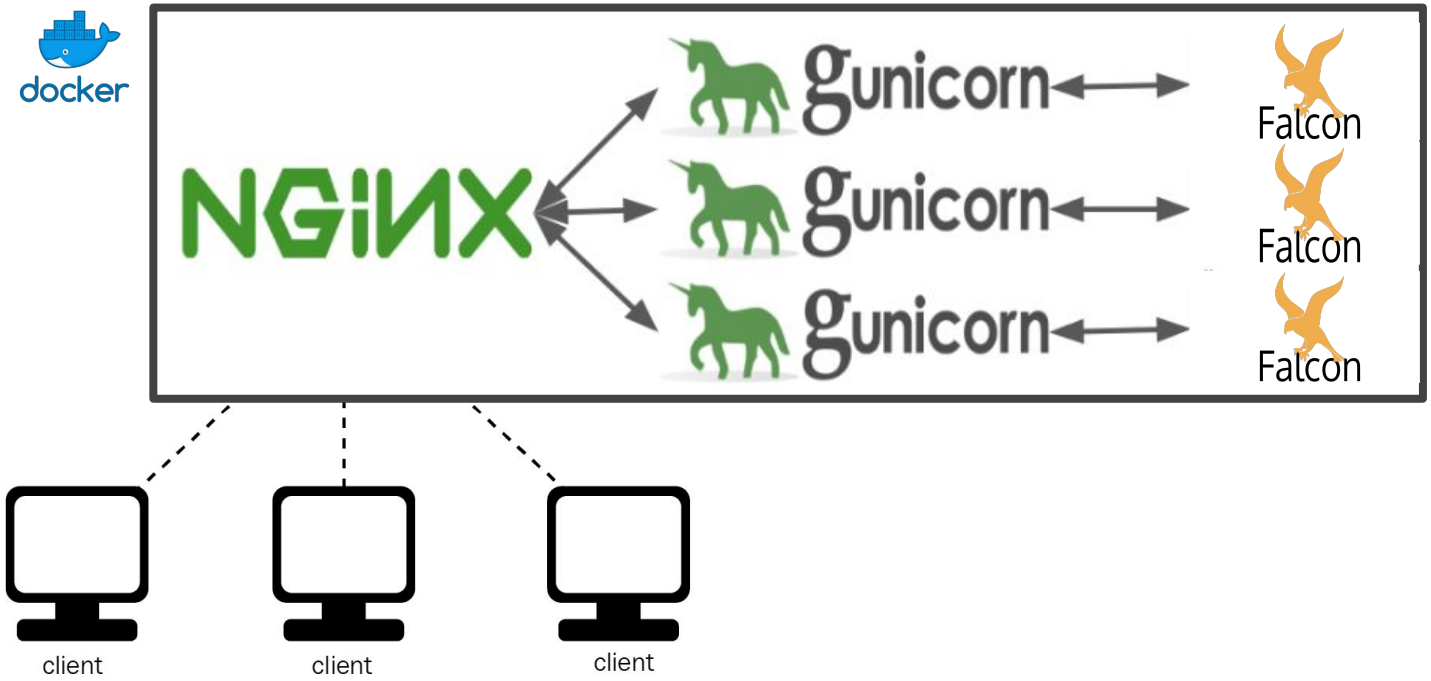
# Building a Web Server with Gunicorn & NGINX



# Multiprocessing in our Web Server



# Building a Docker Image



# Our Dockerfile

```
---
```

```
cat deployWithDocker/docker/Dockerfile
```

```
FROM python:3.7.1

RUN apt-get update && apt-get install -y nginx

RUN pip install tensorflow keras falcon falcon-multipart gunicorn pillow

WORKDIR /model_server/

COPY . .

RUN cp ./nginx.conf /etc/nginx/conf.d/

CMD service nginx restart && \
    gunicorn -b 0.0.0.0:5000 model_server:api
```

# Deploy!

---

```
docker run \
    --rm -d --name ms_container -p 8081:8081 \
    model_server

docker logs ms_container
```

# Test!

---

```
docker run \
    --rm -d --name ms_container -p 8081:8081 \
    model_server

docker logs ms_container

---

curl localhost:8081/health

curl -F “image=@images/img_1.jpg” localhost:8081/predict
```

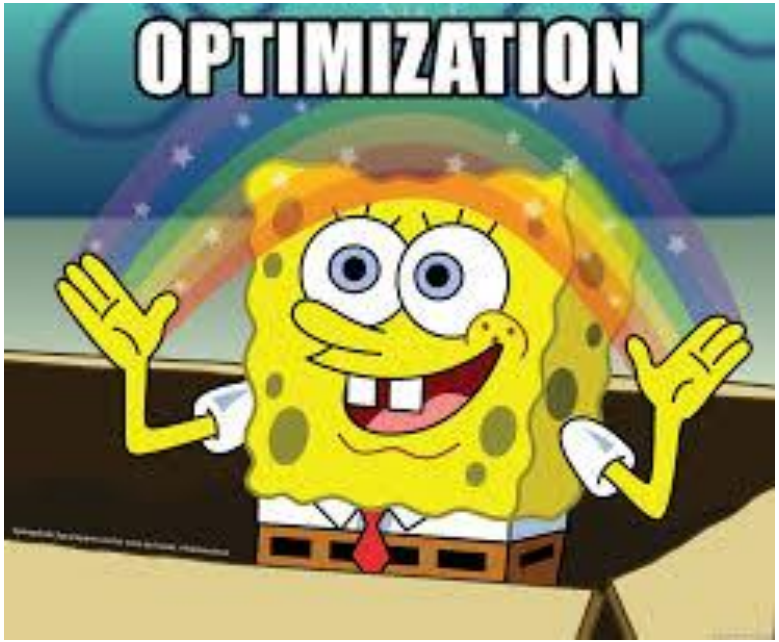
# Deploy to the Cloud!

# But what about real life?

---

Reality is more complicated..

- Model compression
- Parallelization, batching
- EC2 Instance Types
- GPU or AWS Elastic Inference
- Deployment with AWS ECS
- More sophisticated serving frameworks
- Client-side deployment



# Questions?