Shenasa

Tensorflow Serving

Mohadese Yousefi

Contents

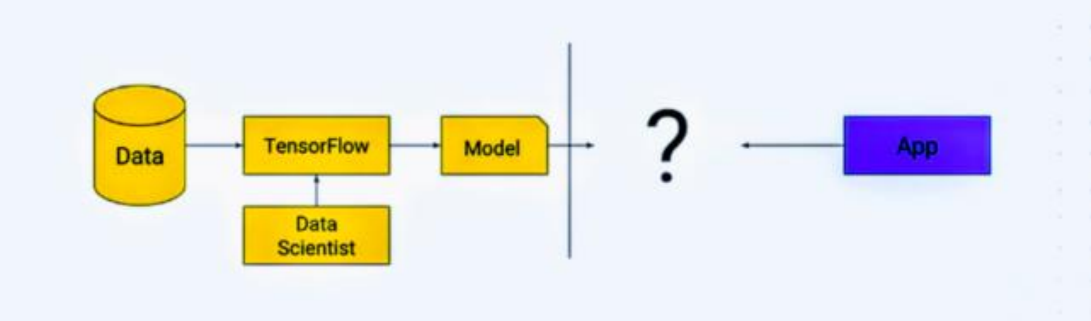
1 What is serving?

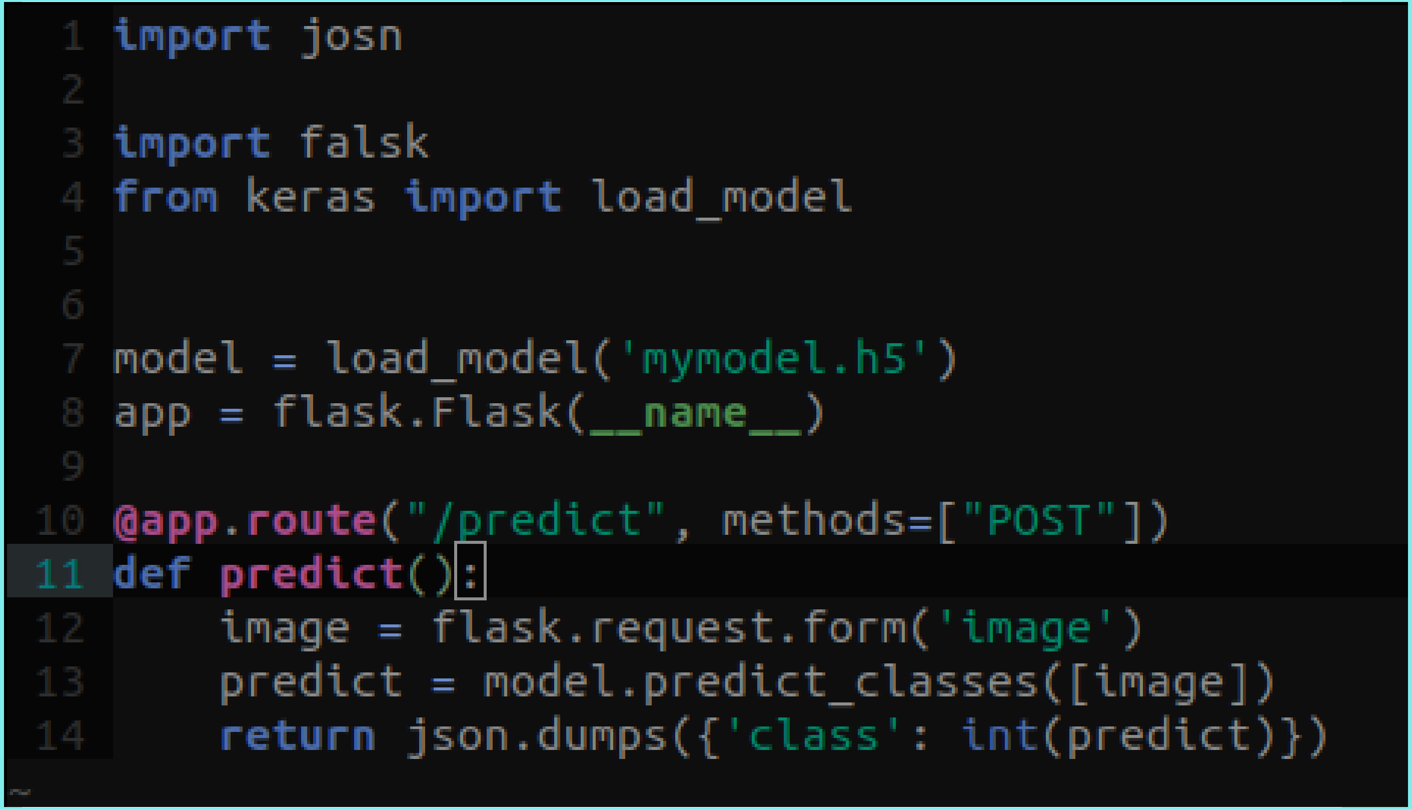
2 Export the model

3 Docker and gRPC

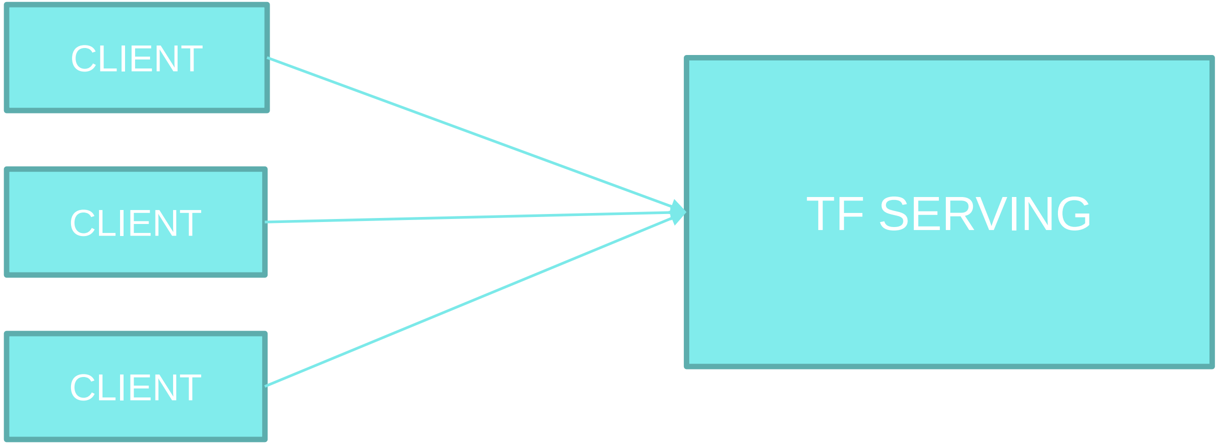
Serving

In machine learning topic, you have to deploy your model for client or customer for prediction or classify their own data.





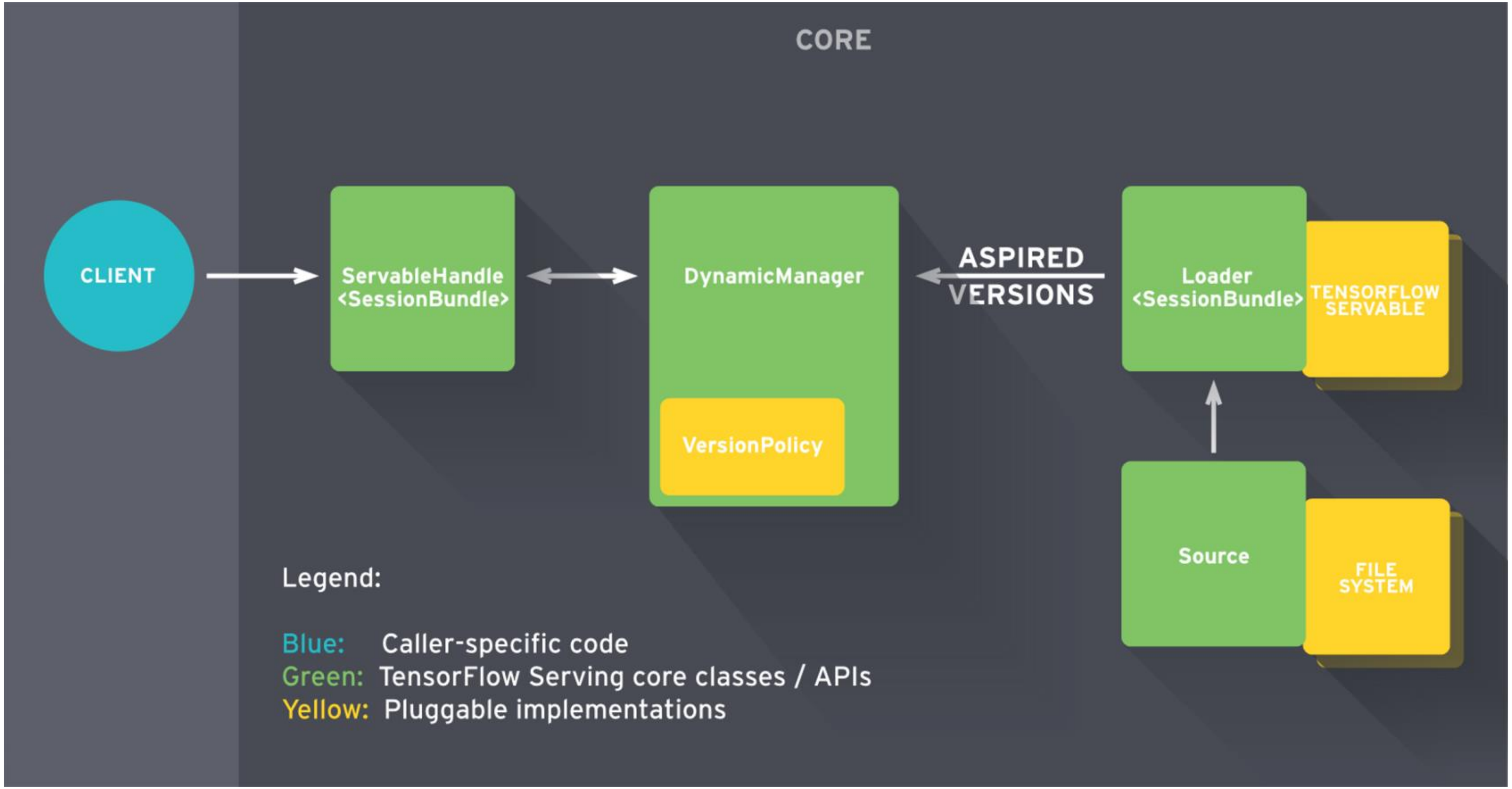
```
import josn

import falsk
from keras import load_model


model = load_model('mymodel.h5')
app = flask.Flask(__name__)

@app.route("/predict", methods=["POST"])
def predict():
    image = flask.request.form('image')
    predict = model.predict_classes([image])
    return json.dumps({'class': int(predict)})
```


CLIENT
CLIENT
CLIENT
TF SERVING

TF serving

- C++ library
- Online, low latency
- Multiple model (A/B testing)
- Multiple version of the same model
- Maintaing the life cycle
- Google opensource library

Servable Architecture



Legend:

Blue: Caller-specific code
Green: TensorFlow Serving core classes / APIs
Yellow: Pluggable implementations

servable

The central abstraction in TensorFlow Serving. Servables are the underlying objects that clients use to perform computation (for example, a lookup or inference)

loader

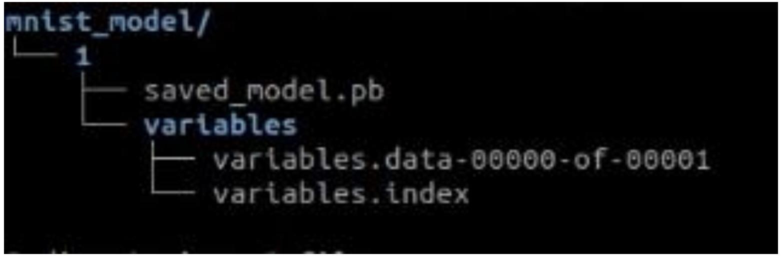
Manage a servable's life cycle. The Loader API enables common infrastructure independent from specific learning algorithms, data or product use-cases involved. Specifically, Loaders standardize the APIs for loading and unloading a servable.

manager

Managers listen to Sources and track all versions. The Manager tries to fulfill Sources' requests, but may refuse to load an aspired version if, say, required resources aren't available. Managers may also postpone an "unload".

Export Model

```
import tensorflow as tf


tf.keras.models.save_model(
    model,
    filepath,
    signatures=None,
)
```



```
mnist_model/
└── 1
    ├── saved_model.pb
    └── variables
        ├── variables.data-00000-of-00001
        └── variables.index
```

Run the code.

Docker

Docker is a platform for developers and sysadmins to **build, run, and share** applications with containers. The use of containers to deploy applications is called *containerization*. Containers are not new, but their use for easily deploying applications is.

- **Flexible**
- **Lightweight**
- **Portable**
- **Loosely coupled**:
- **Scalable**
- **Secure**

Installation

https://docs.docker.com/engine/install/

Essentials docker commands

docker images : lists all images in a docker hosts.

docker ps: list containers.

docker run <image_name>: run a container. You need to give at least a docker image as a parameter

docker pull <image_name>: pull an image from docker registry.

docker build –t <define the name of image> . : build a docker image from a Dockerfile

Pull tensorflow serving docker

doker images are available for cpu and gpu hardware

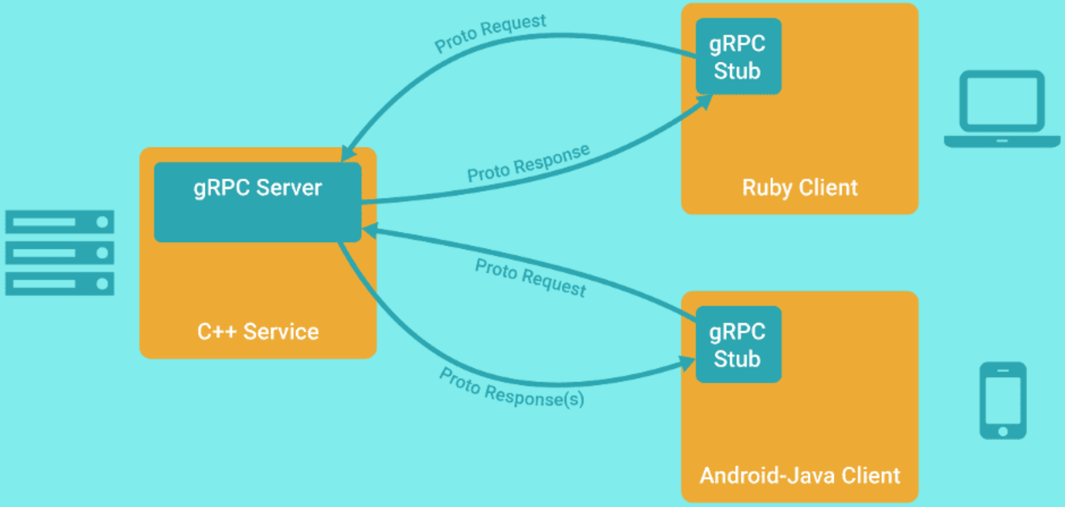
```
docker pull tensorflow/serving
```

```
docker run -p 8501:8501 \
           -p 8500:8500 \
           --mount type=bind,\
           source=/path/to/my_model/,\
           target=/models/my_model \
           -e MODEL_NAME=my_model \
           -t tensorflow/serving
```

```
docker run          …..
                              tensorflow/serving:latest-gpu
```

grpc

gRPC is a modern open source high performance RPC framework that can run in any environment. It can efficiently connect services in and across data centers with pluggable support for load balancing, tracing, health checking and authentication. It is also applicable in last mile of distributed computing to connect devices, mobile applications and browsers to backend services.



Inferences via gRPC

- Inference data needs to be converted to the Profobuf format.
- Request types have designed types. e.g float, int
- Payloads need to be converted base64
- Connect to the server via gRPC stubs

gRPC vs REST

- REST is easy to implement and to debug
- RPC is more network efficient, smaller payloads.
- RPC can provide much faster inferences!