Deployment

02476 Machine Learning Operations

Nicki Skafte Detlefsen,

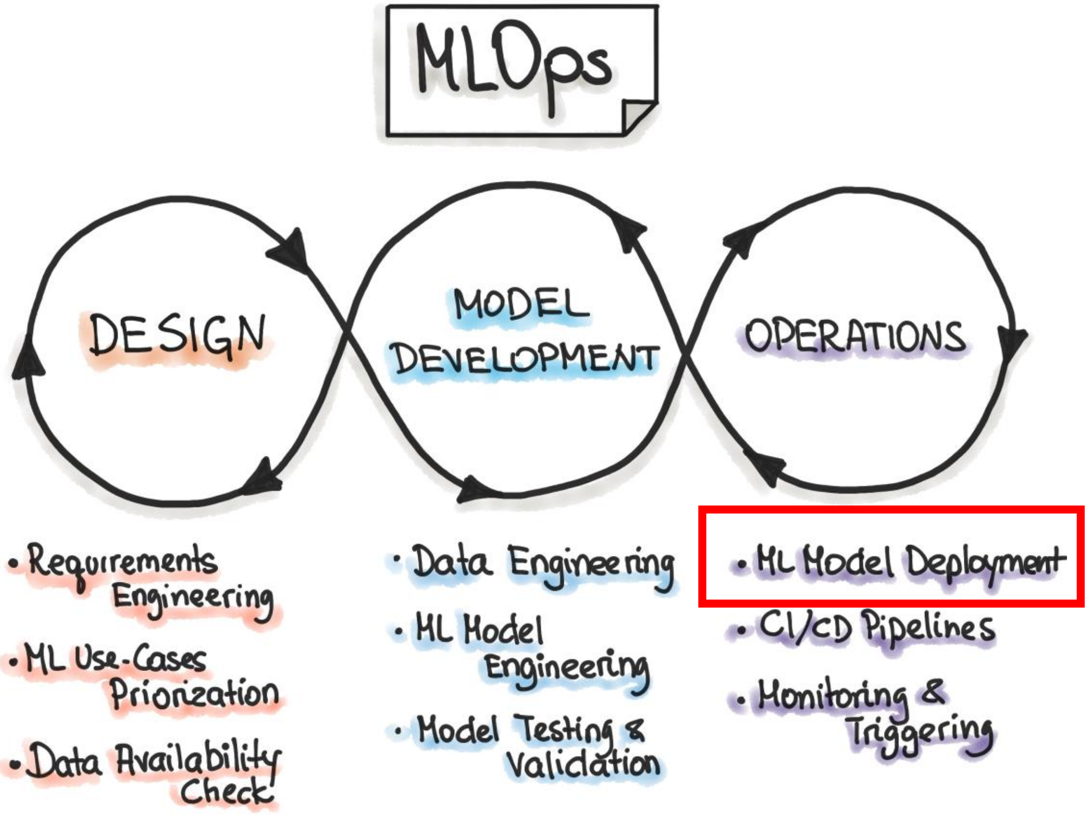
Postdoc

DTU Compute

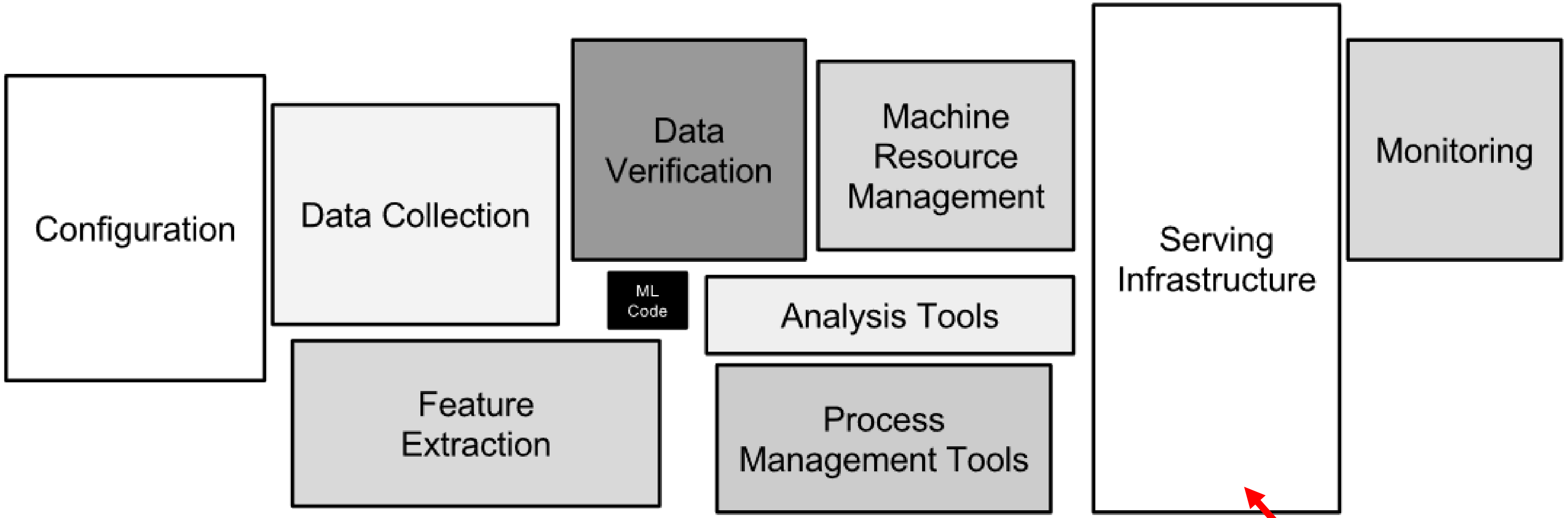
Loosely based on https://www.youtube.com/watch?v=2awmrMRf0dA

Freeing the model

- Model deployment is part of the operations in MLOps
- In a nutshell: make the model available to others


MLOps
DESIGN
MODEL DEVELOPMENT
OPERATIONS
• Requirements Engineering
• ML Use-Cases Priorization
• Data Availability Check
• Data Engineering
• ML Model Engineering
• Model Testing & Validation
• ML Model Deployment
• CI/CD Pipelines
• Monitoring & Triggering

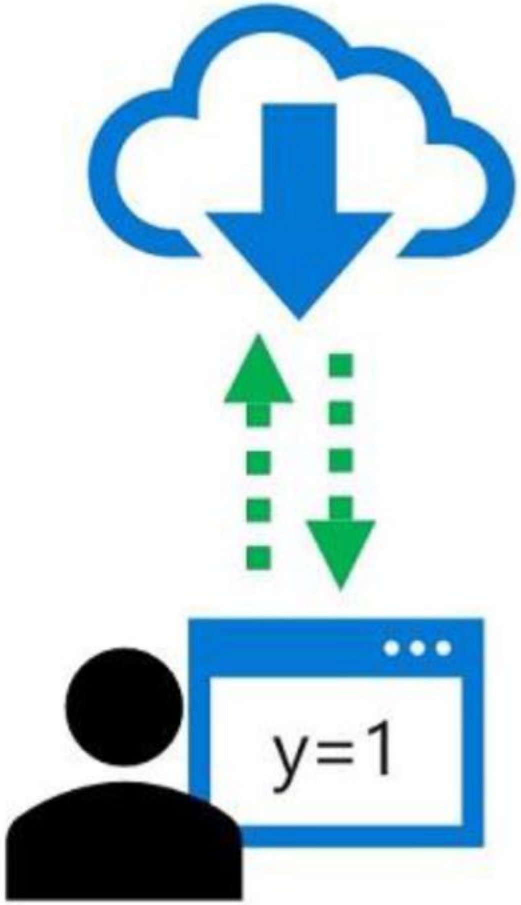
Remember this?



This is big for a reason

What do we want to deploy

In ML, *inferencing* refer to the use of a trained model to predict labels for new data on which the model has not been trained



Many levels of deployment (within machine learning)

1. Github reposatory + link to model weights
    - Easy to ”deploy”
    - Pain in the *** to use
2. Deploy on local computer/cluster
    - Fairly easy getting up and running, just requires people can access from outside
    - Can be fairly easy to use
    - Does not scale at all
3. Deploy to cloud service
    - Can be a pain to setup
    - Easy to use and scales to ∞ (and beyond!)

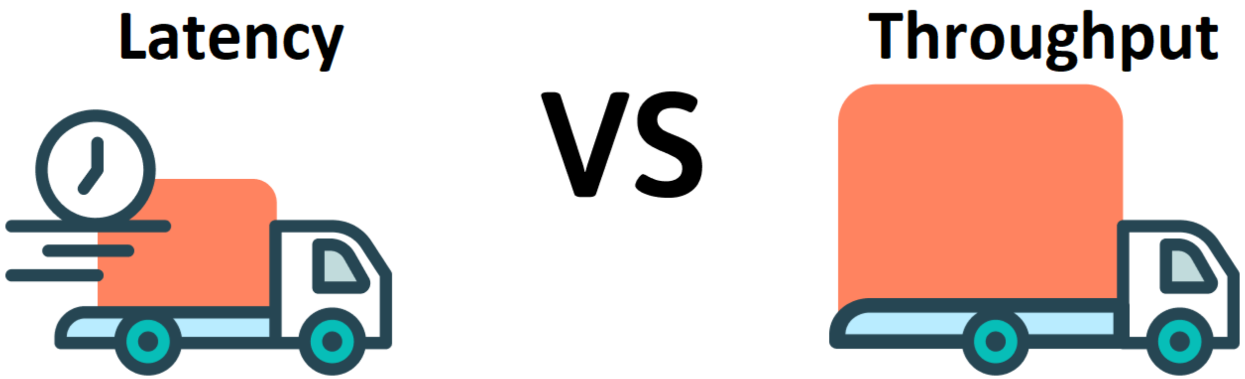
Production requirements

1. Portability

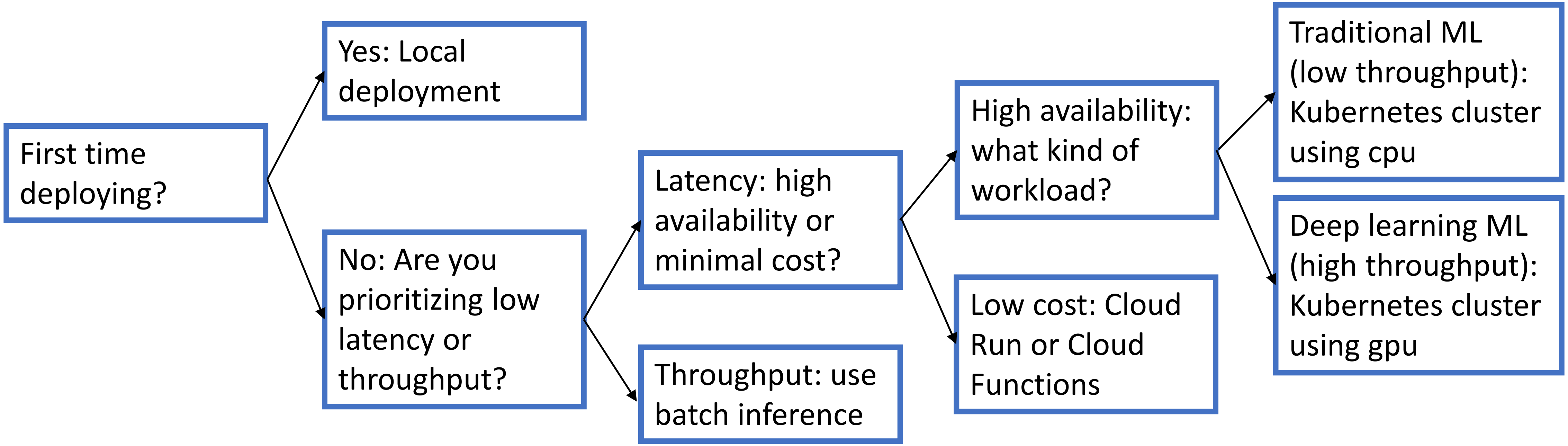
   Models should be exportable to wide variety of enviroments, from C++ servers to mobile

2. Performance

   We want to optimize common patterns in neural networks to improve inference latency and throughput



Choosing the right service



First time deploying?

Yes: Local deployment

No: Are you prioritizing low latency or throughput?

Latency: high availability or minimal cost?

Throughput: use batch inference

High availability: what kind of workload?

Low cost: Cloud Run or Cloud Functions

Traditional ML (low throughput): Kubernetes cluster using cpu

Deep learning ML (high throughput): Kubernetes cluster using gpu

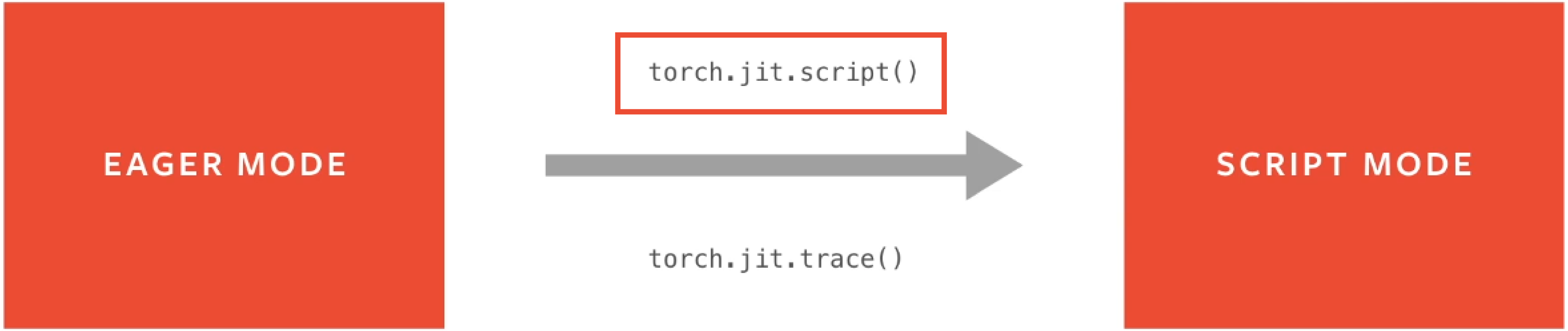
What are the challenges with Pytorch in production

- Pytorch is a dynamic framework (uses a dynamic graph)
  - This is not great in production as we need to know sizes etc. for compilation and optimization

- Why not use a static framework (Tensorflow 1.x, Caffe2 etc.)?
  - Do you really want to port all your work?

- What can we do to solve this?

Convert to script mode!



EAGER MODE

`torch.jit.script()`

`torch.jit.trace()`

SCRIPT MODE

For prototyping, training, experimenting

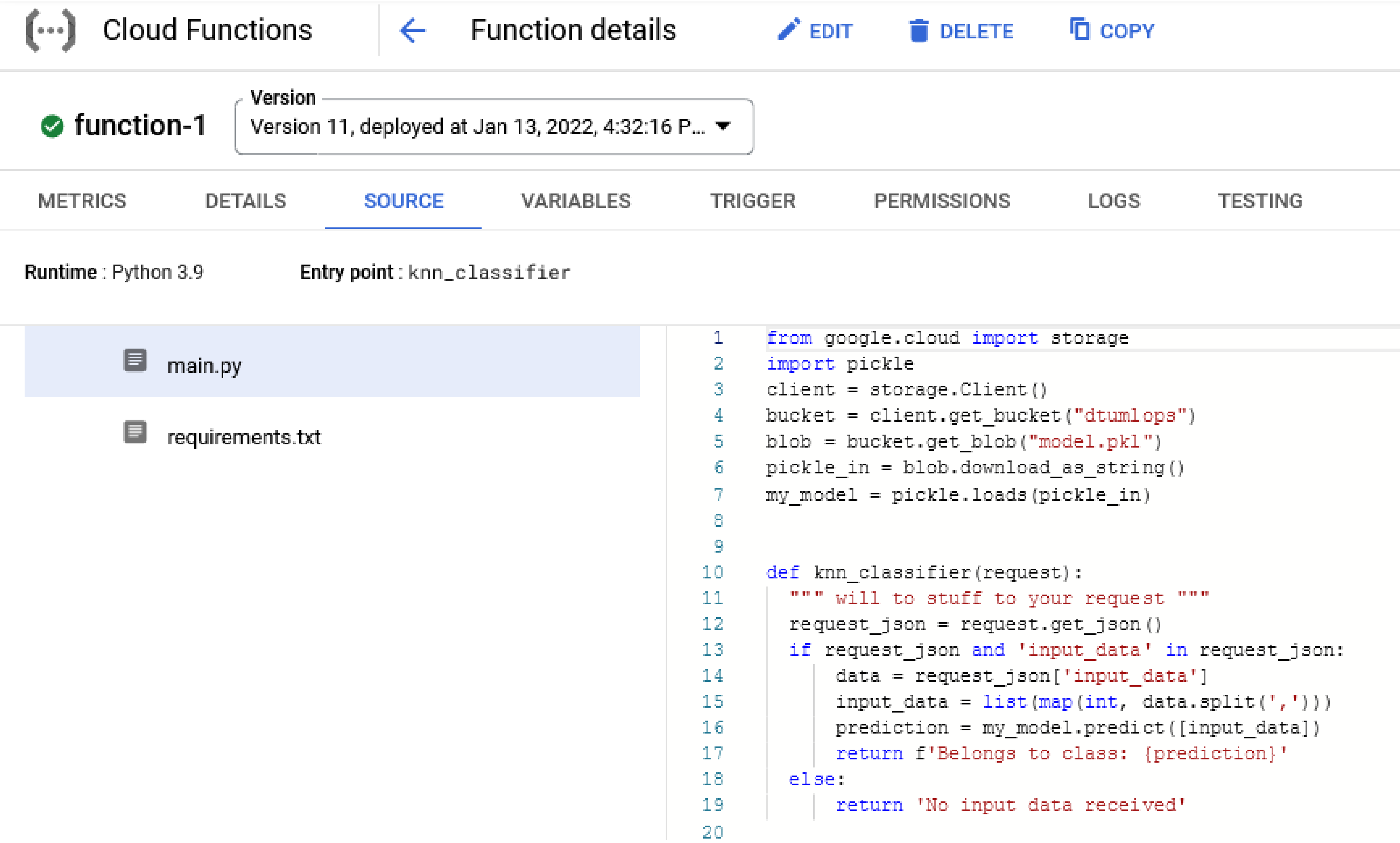
For production deployment

Serilization

- torch.jit.script serialize the model, but what does it mean?
- Serilization essentially encodes all modules methods, submodules, parameters, and attributes into a byte stream
- This makes the encoded model independent of python!
- This is basically just ”pickling” and “unpickling”.

Cloud functions

Simple one script files for deployment



function-1

Version: Version 11, deployed at Jan 13, 2022, 4:32:16 P...

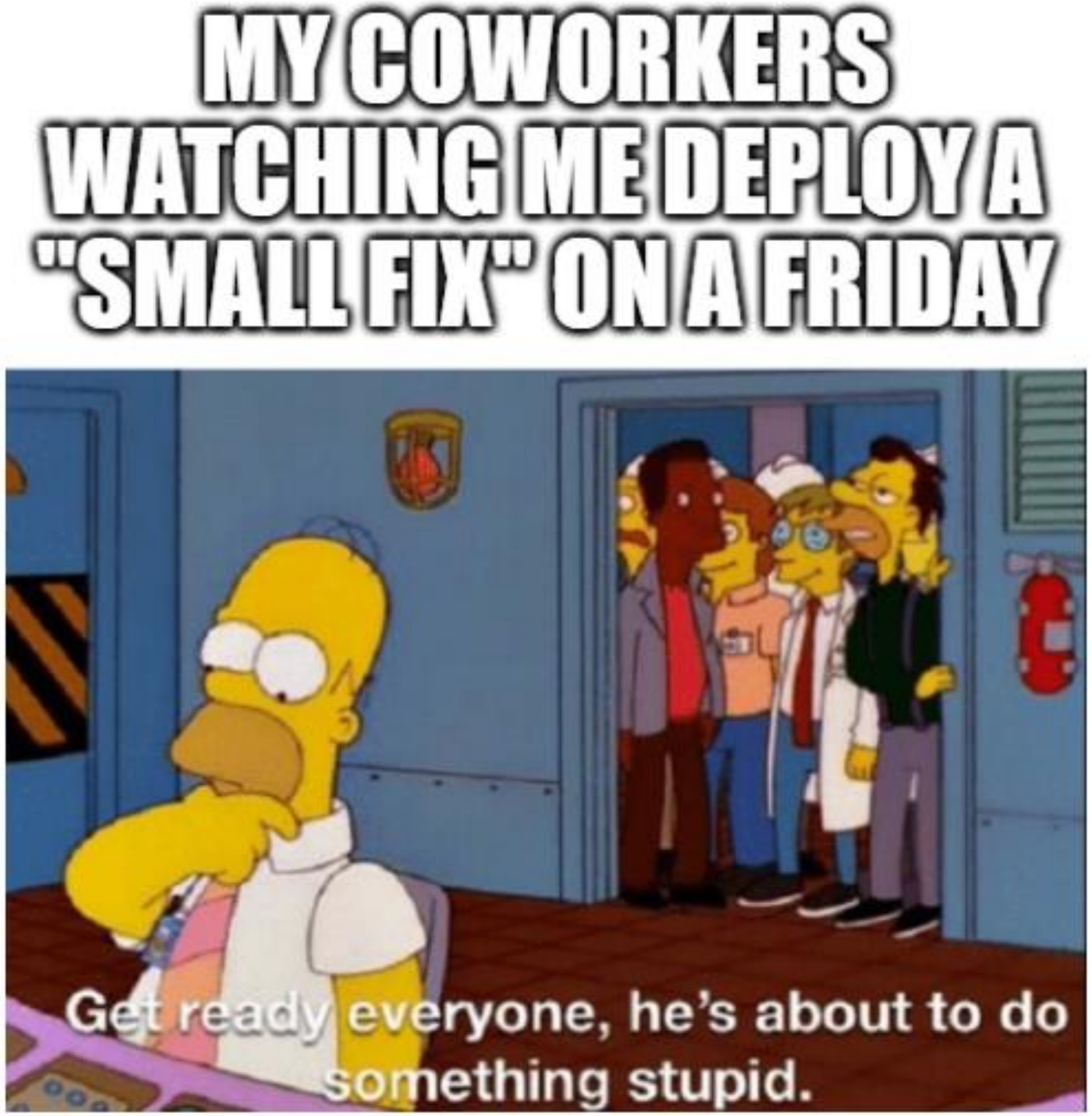
Runtime : Python 3.9 **Entry point** : knn_classifier

```python
from google.cloud import storage
import pickle
client = storage.Client()
bucket = client.get_bucket("dtumlops")
blob = bucket.get_blob("model.pkl")
pickle_in = blob.download_as_string()
my_model = pickle.loads(pickle_in)


def knn_classifier(request):
  """ will to stuff to your request """
  request_json = request.get_json()
  if request_json and 'input_data' in request_json:
      data = request_json['input_data']
      input_data = list(map(int, data.split(',')))
      prediction = my_model.predict([input_data])
      return f'Belongs to class: {prediction}'
  else:
      return 'No input data received'

```

Meme of the day


MY COWORKERS WATCHING ME DEPLOY A "SMALL FIX" ON A FRIDAY
Get ready everyone, he's about to do something stupid.