Deployment

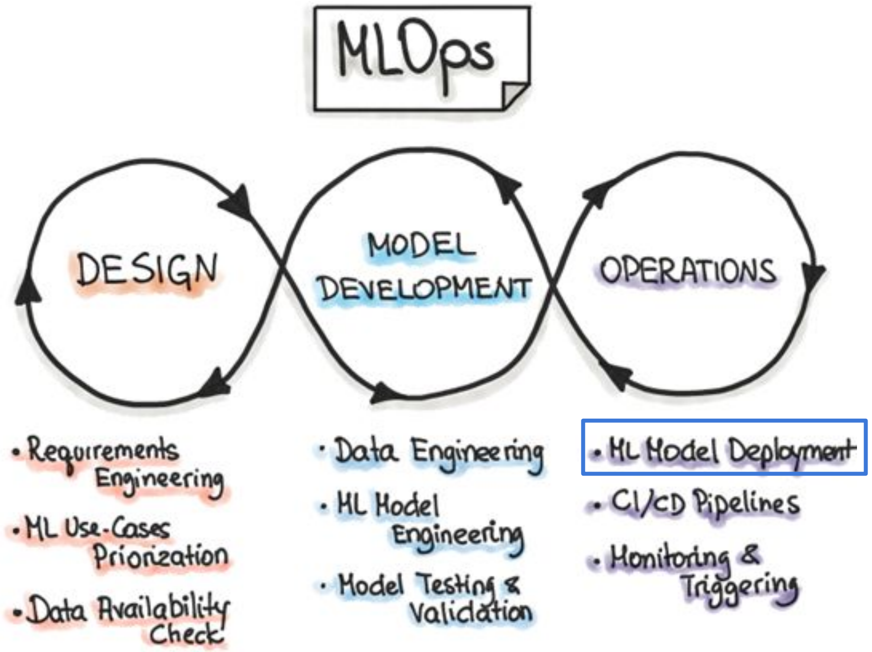
02476 Machine Learning Operations

Nicki Skafte Detlefsen

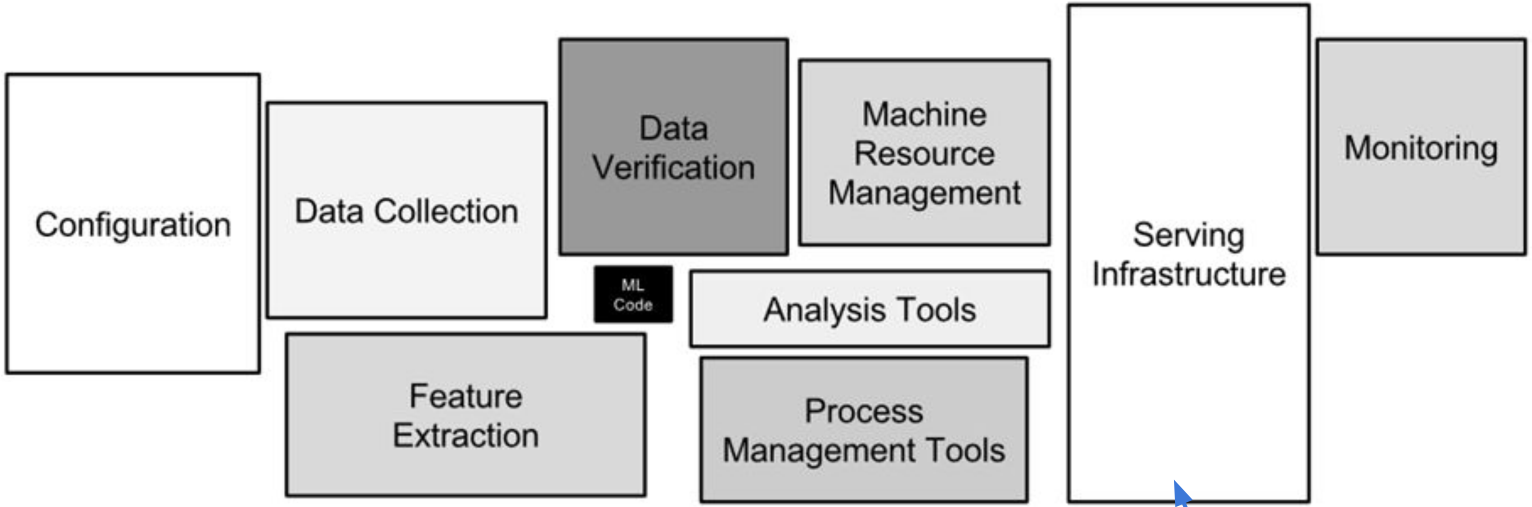
Freeing the model

In a nutshell:

Make model available to invoke easily and continuously



Easy to get started, hard to get right



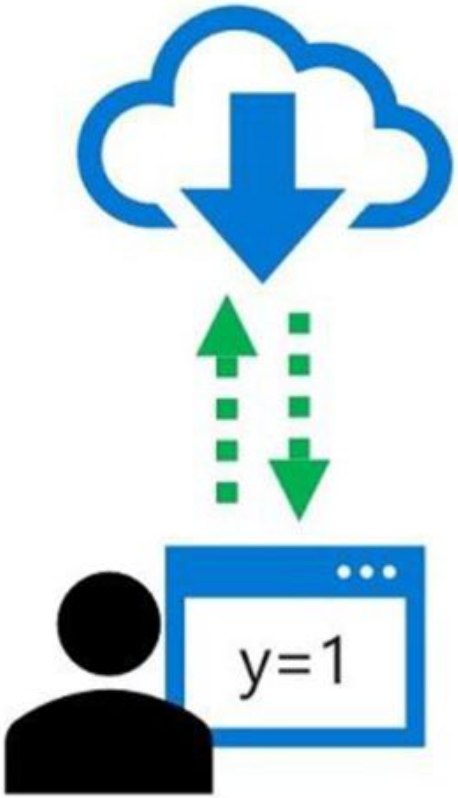
This is big for a reason

Server infrastructure depends on your use case

What to 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.

Around 80% of compute spend in the cloud on machine learning is spend on inference.



Production requirements

1. Portability

   Models should be exportable to wide variety of environments, from c++ servers to mobile

2. Performance

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

Latency

VS

Throughput

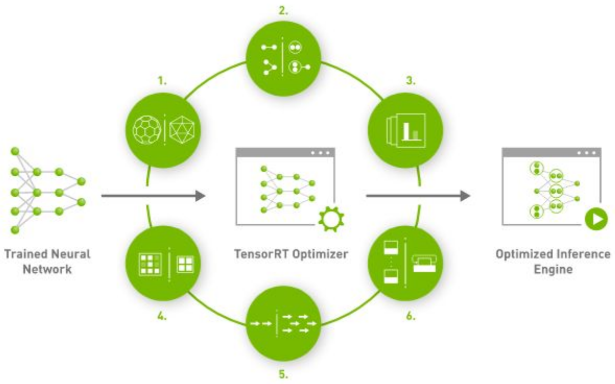
Before you deploy

Start by optimizing your model!

- Pruning
- Quantization
- Compile to low language
- Device optimizations*

to increase throughput, reduce memory and reduce energy consumption

Assume we have such model



*TensorRT

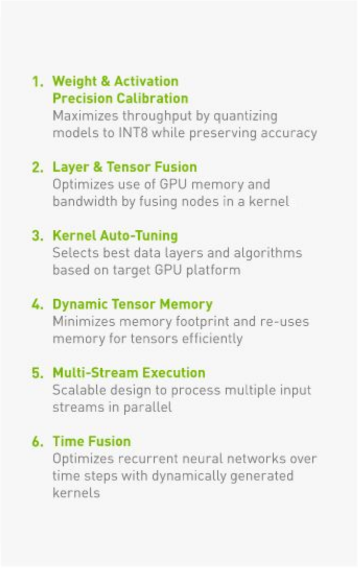
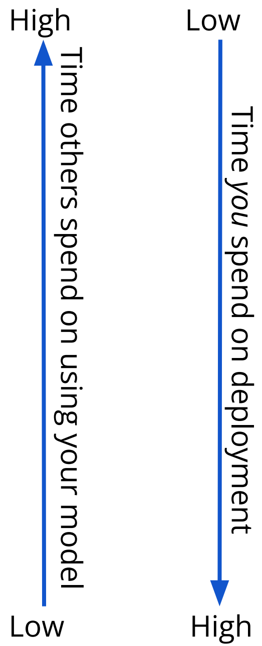


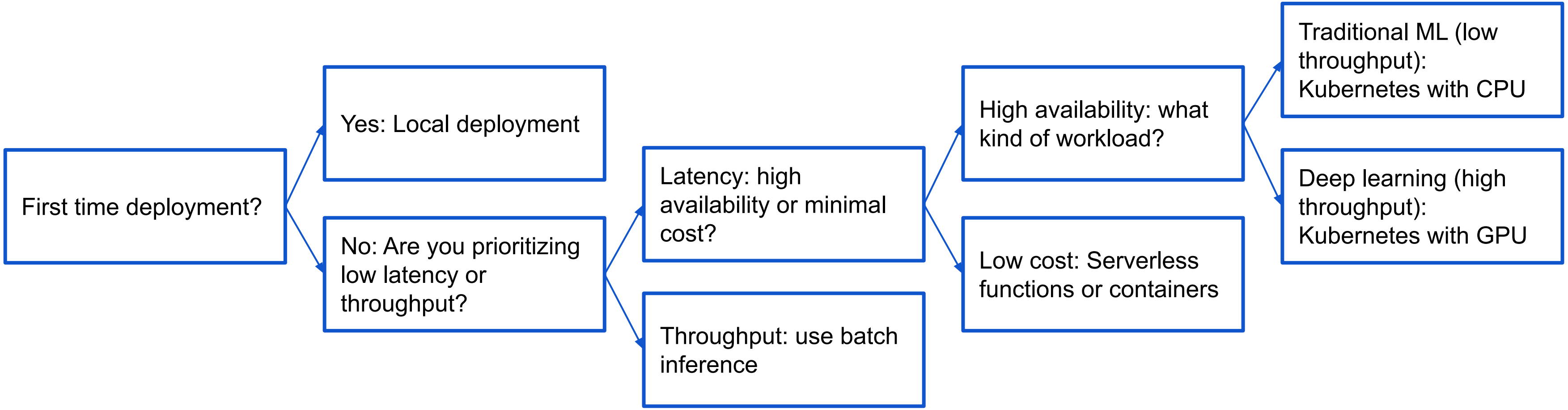
Image credit: https://developer.nvidia.com/tensorrt

Many levels of deployment

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



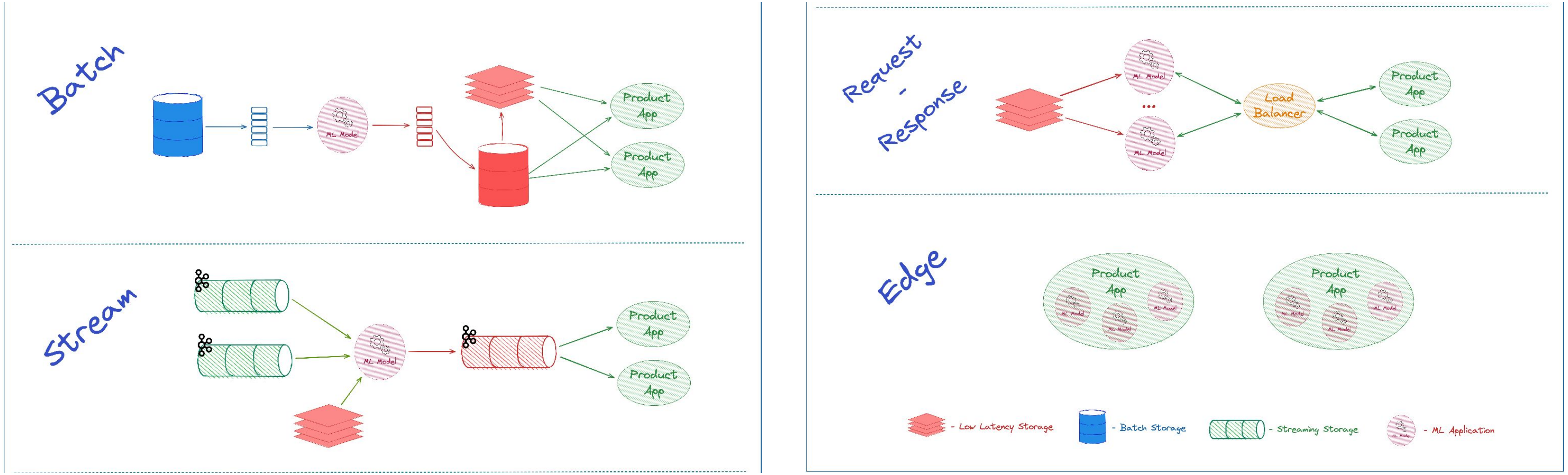
Choosing the right service



Different kind of deployments

Scheduled run

On demand run



Continues run

Embedded run

Image credit: https://swirlai.substack.com/p/sai-03-machine-learning-deployment

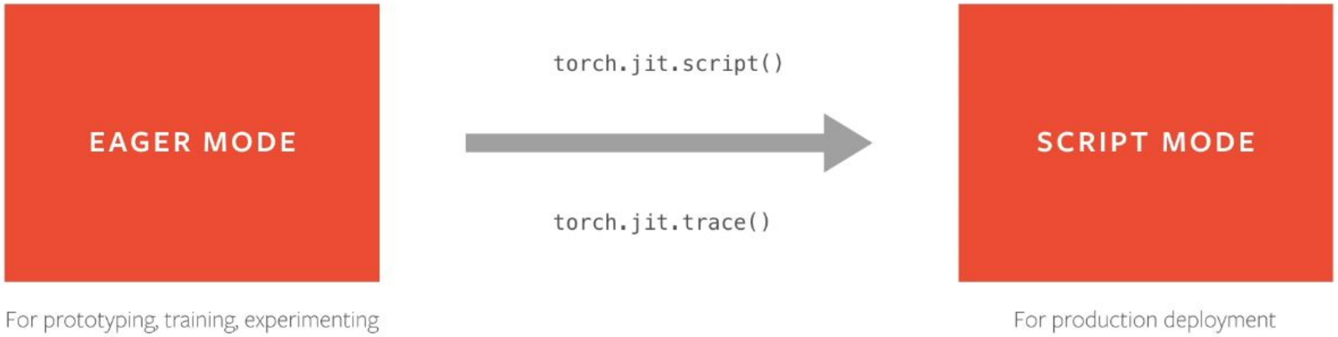
Deployment of Pytorch applications

Pytorch is a dynamic framework (uses a dynamic graph)

- Great for development
- In practise a lot of performance is lost to the JIT compiler

Image credit: http://pytorch.org/about/

Solution: convert to script mode



`torch.jit.script` serialize the model, but what does that mean?

- Serialization 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”.

Other options for Pytorch

TensorRT

ONNX

GLOW

Torchserve

You can also use FastAPI or Flask

API=Application Programming Interface

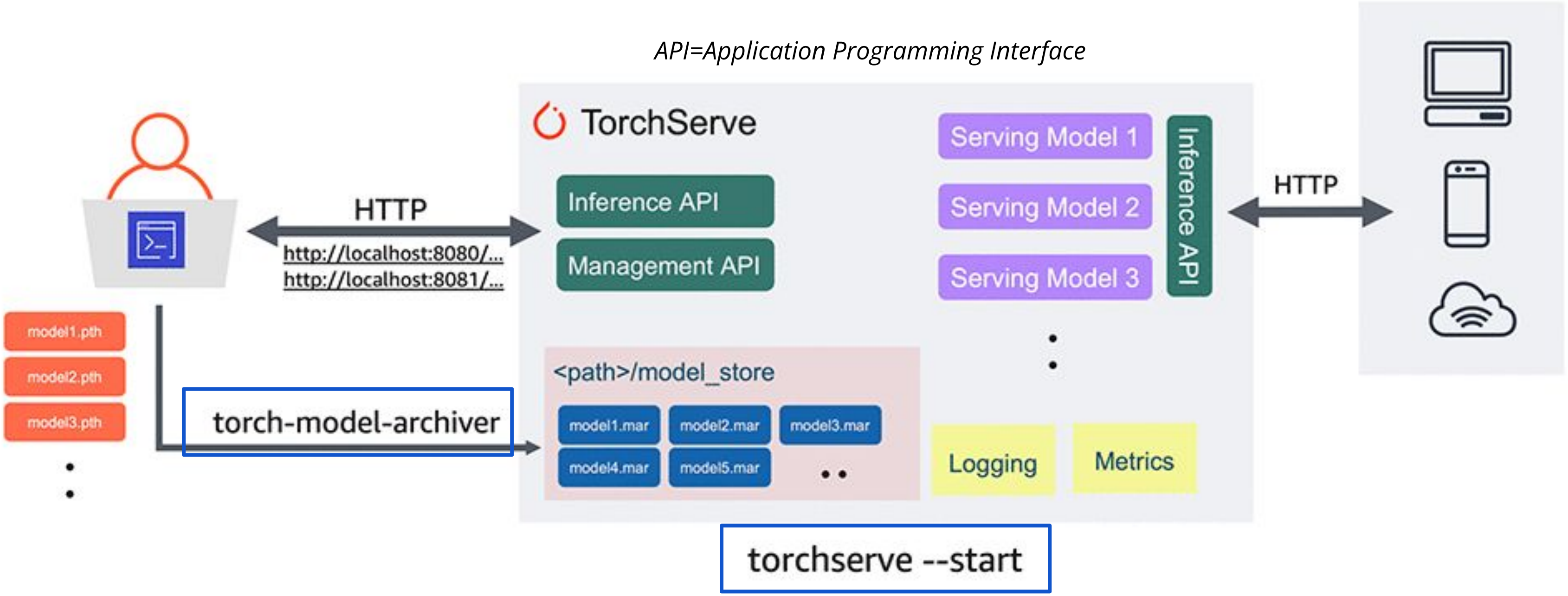


Image credit: https://aws.amazon.com/blogs/machine-learning/deploying-pytorch-models-for-inference-at-scale-using-torchserve/

Cloud deployment

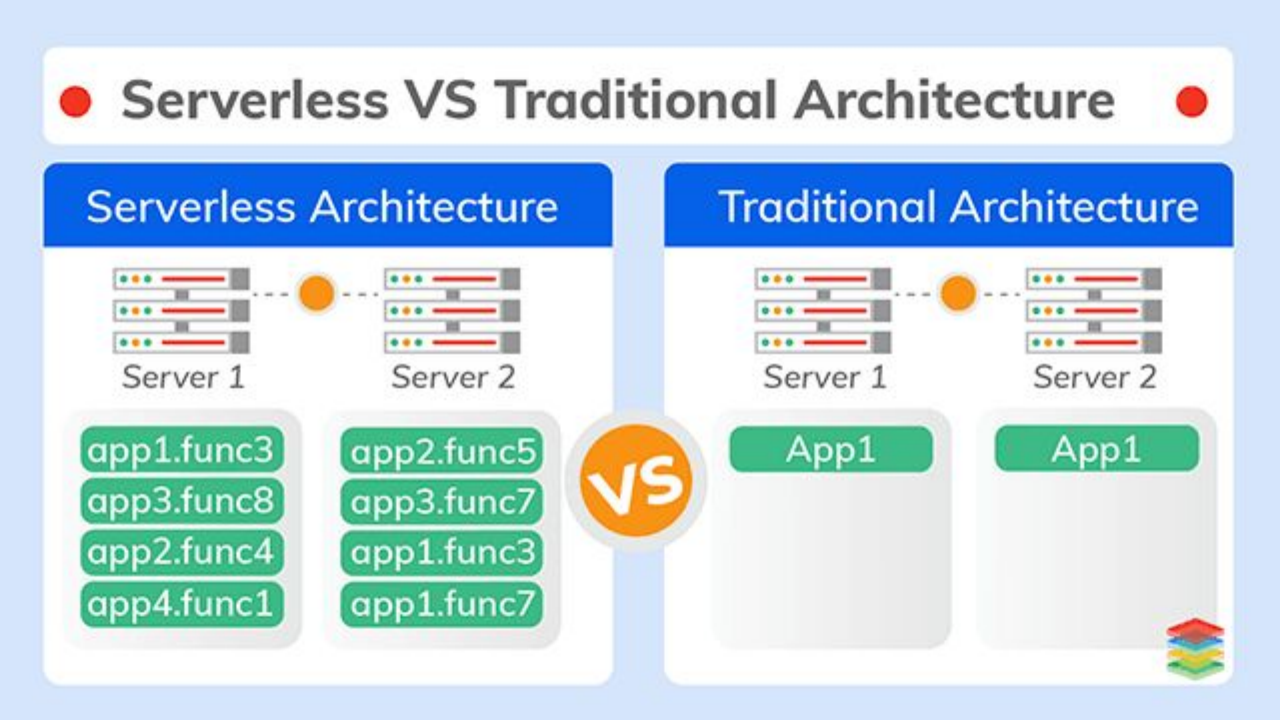


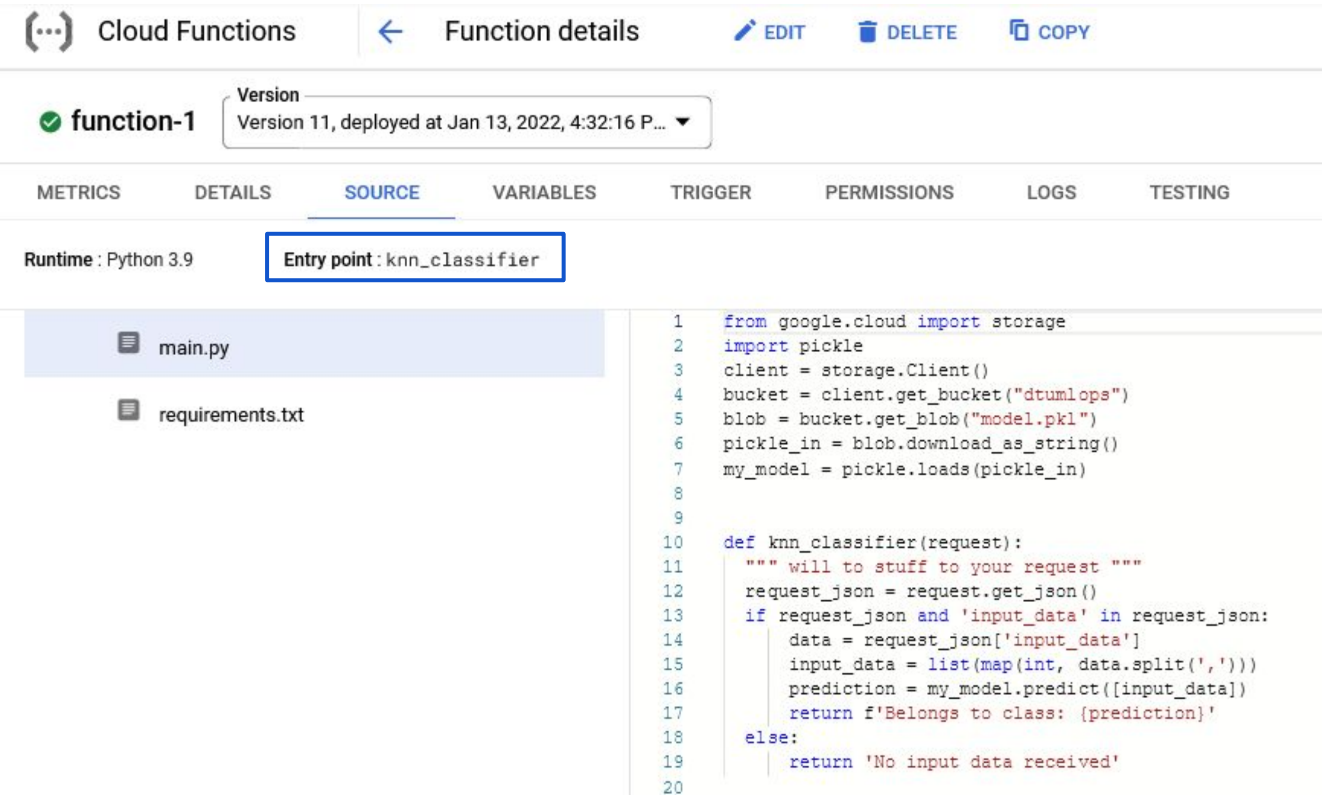
Image credit: https://medium.com/digital-transformation-and-platform-engineering/what-is-serverless-computing-serverless-architectures-xenonstack-36c37952b4c3

GCP Functions

Simple two script deployment

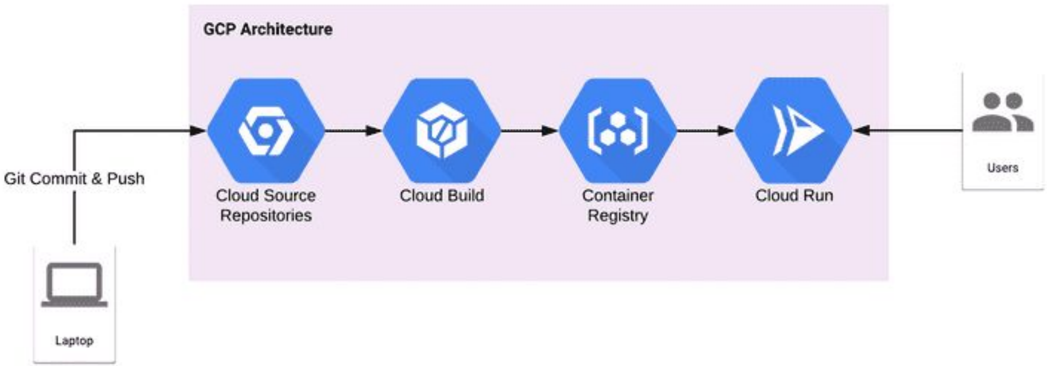
- dependencies
- single script

Limited in expressiveness



GCP Run

If you work with containers Run offers more expressive interface (because containers are more expressive)

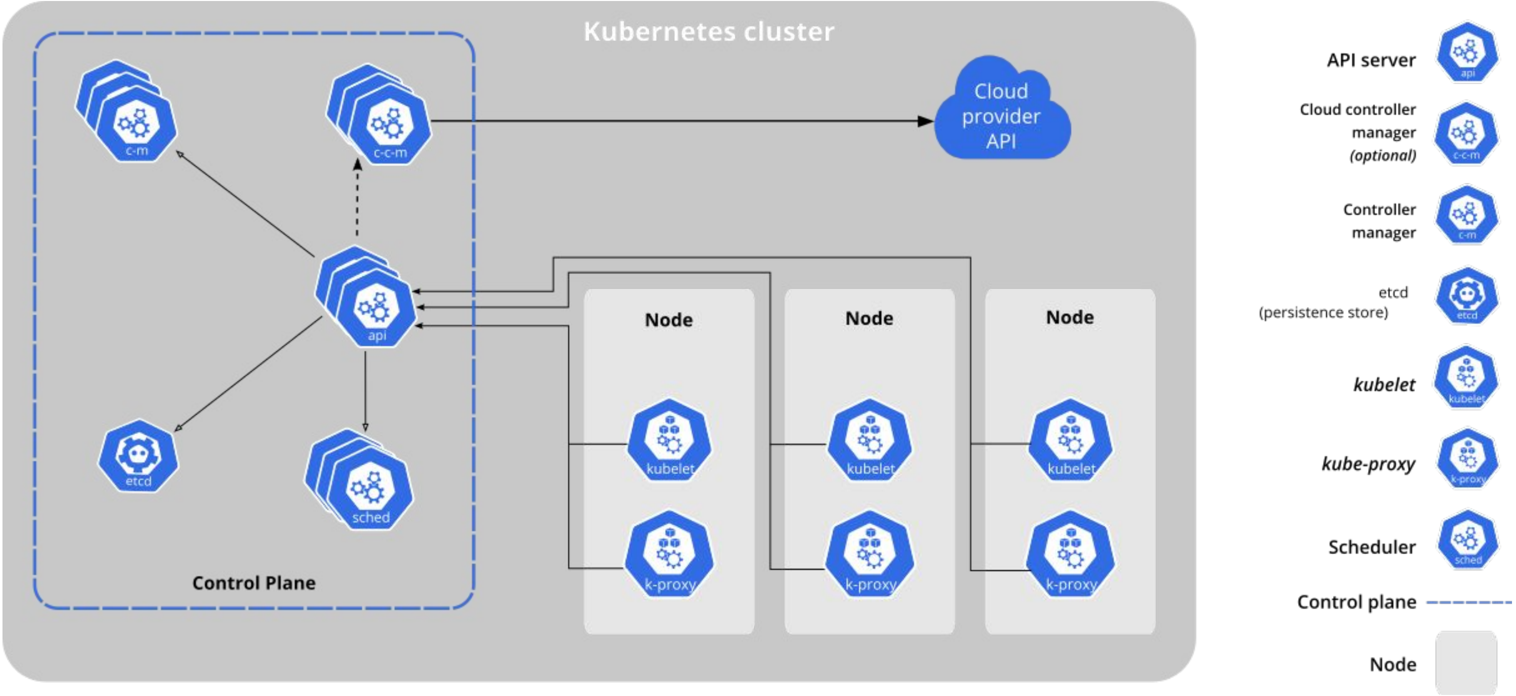


Run is still serverless deployment

Image credit: https://levelup.gitconnected.com/ci-cd-pipeline-on-gcp-using-cloud-source-repository-cloud-build-container-registry-and-cloud-run-b66413133933

Kubernetes (not part of the course, yet)

If you want to be in charge of the cluster



Meme of the day