Productionising ML models Developed in R

a.k.a I have R Models, now what?!

Surya Avala

[Email](), [LinkedIN]()

15:00

Housekeeping

✓ Please stay on mute and turn your video off.

✓ If you have questions submit them via the chat feature.

We'll try to make the Q&A as interactive as possible.

✓ All our brownbags are recorded and published on the Eliiza

Youtube

✓ More MLOps / Productionisation Brown Bags coming.. Stay

Tuned by signing up to our mailing list

Working with Us

✓ If you're keen to explore some of these ideas further, optimise your machine learning models in production, or enhance your data science practices and tooling: *we can help*!

✓ If you want us to host something specific for you & your team - a brownbag, a training, or something else: *we can help*!

✓ Please get in touch directly at info@eliiza.com.au

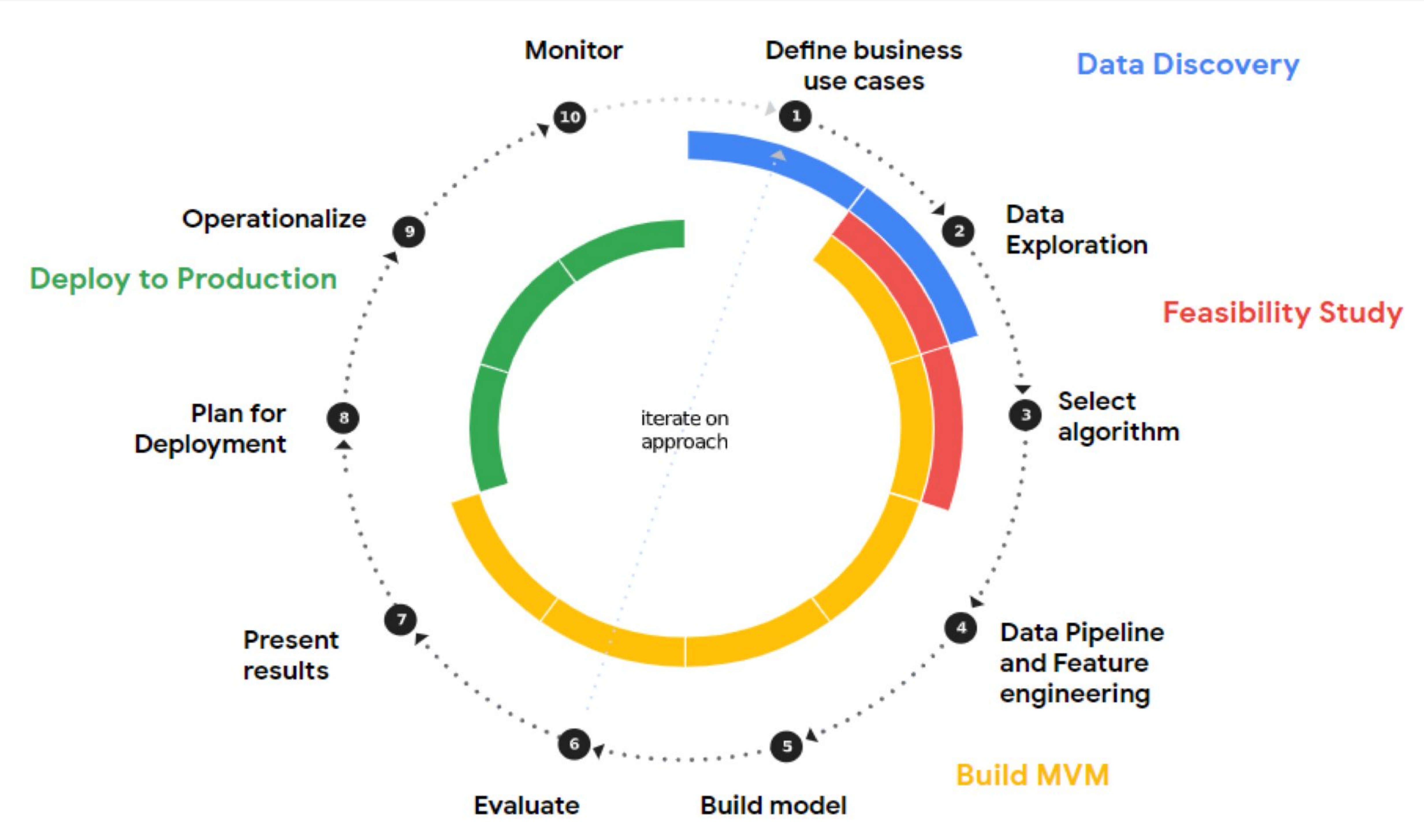
Productionising ML models Developed in R

a.k.a I have R Models, now what?!

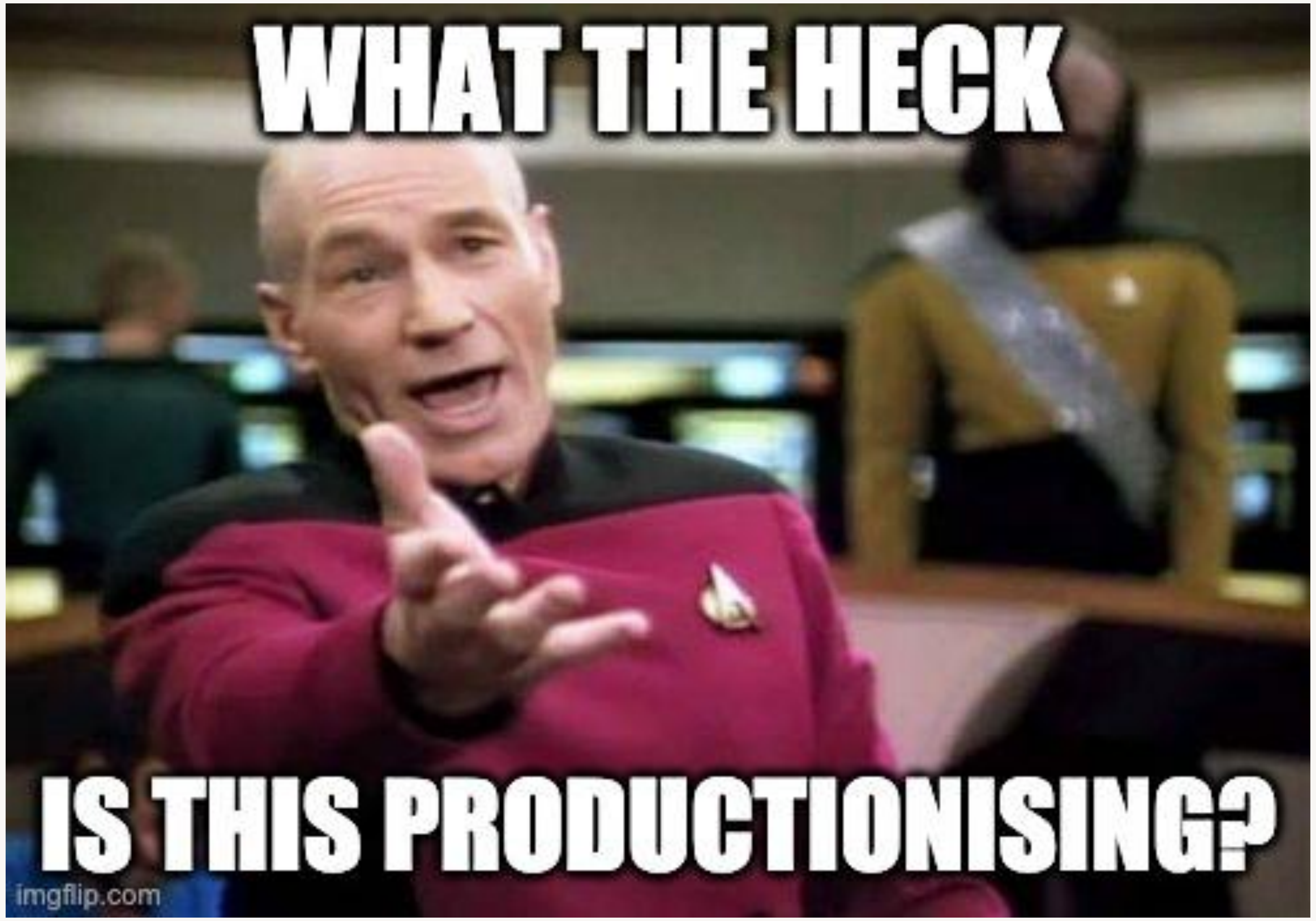
Surya Avala


WHAT IF I TOLD YOU
THAT THE BEST MODEL
IS THE ONE THAT'S DEPLOYED!
imgflip.com

ML Lifecycle



Img Source: Woolpert Blog


WHAT THE HECK
IS THIS PRODUCTIONISING?
imgflip.com

Productionise

productionise

/prəˈdʌkʃ(ə)nʌɪz/

verb

the process of taking a developed model, finalising it and making it available and ready in an operationally active, or production, environment.

"hey boss, watch out for the extra revenue, i've just productionised my model!"

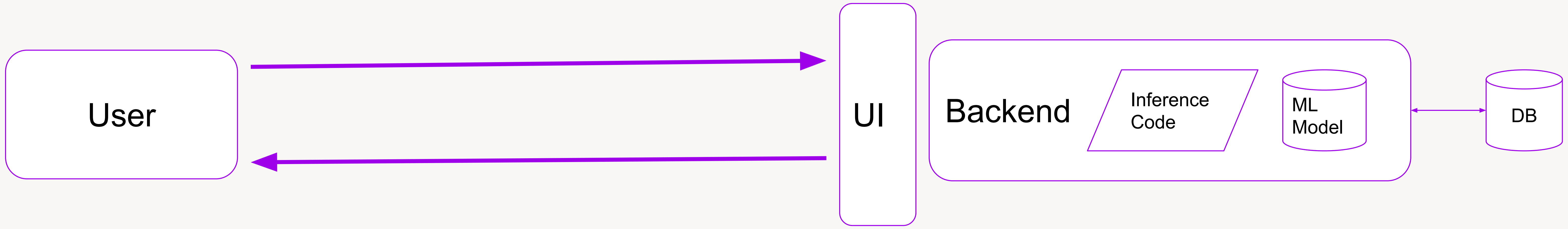
Source: Internet Archives

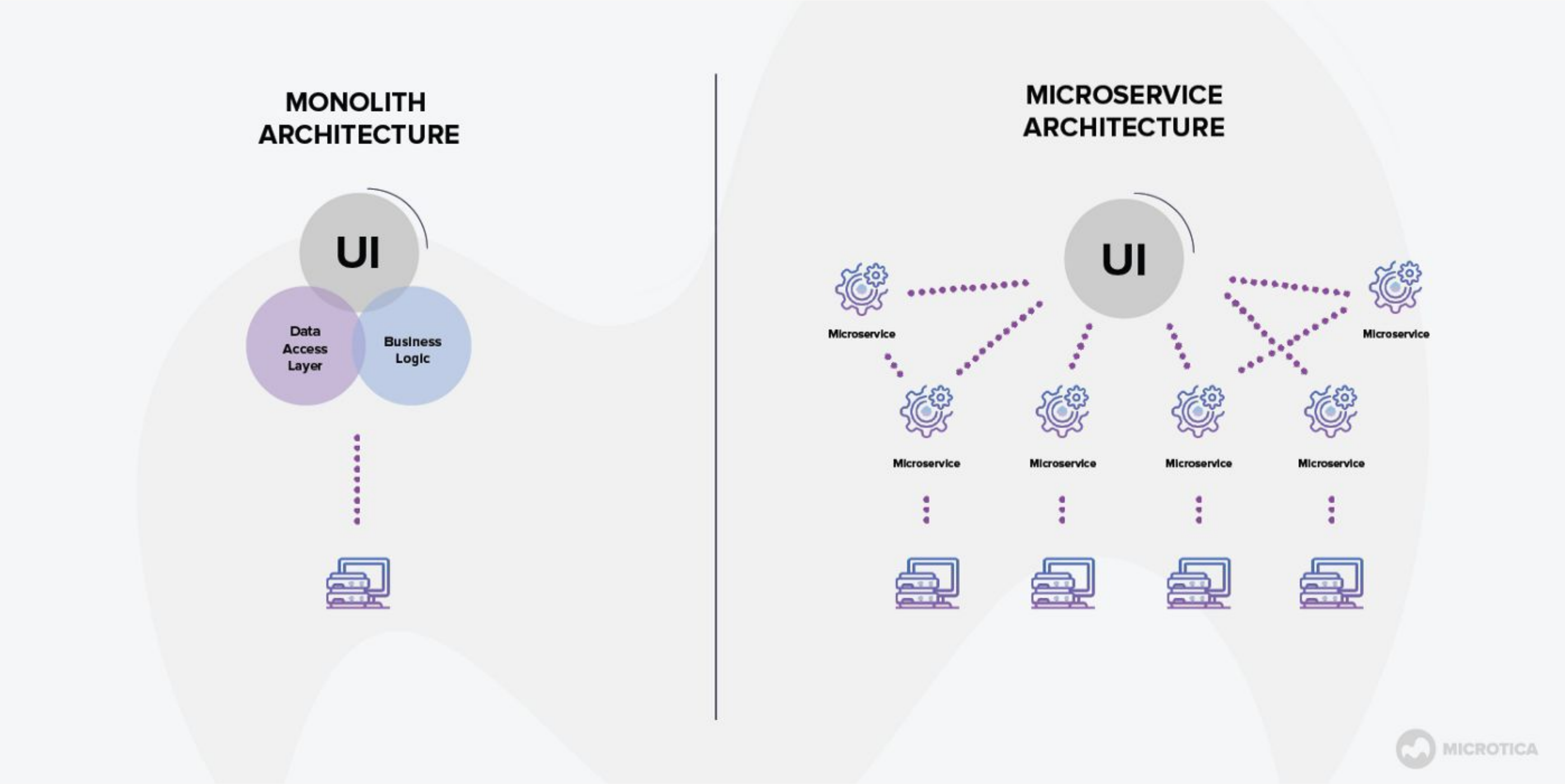

BUT HOW?
memegenerator

Standalone App

Core Idea

- Traditional Monolithic Architecture
- End user facing UI
- Backend Server
  - Processes prediction requests
  - Predictions displayed/visualised in UI
  - Logs some data into DB
- Shiny, Rmarkdown deployed to shinyapps, rconnect



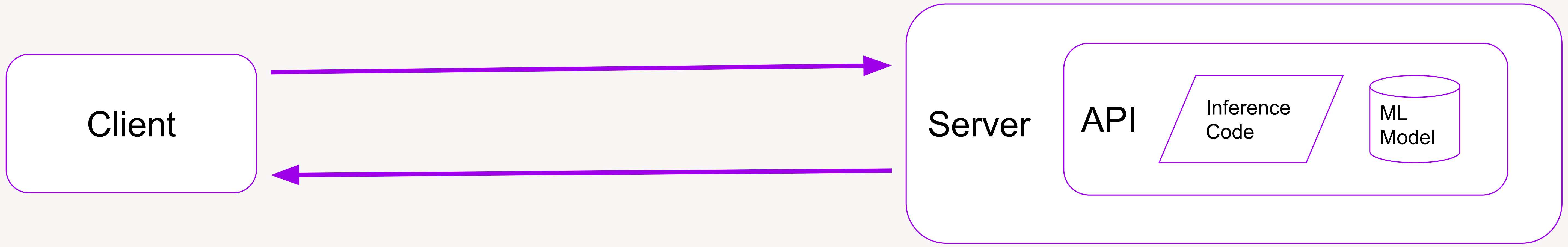


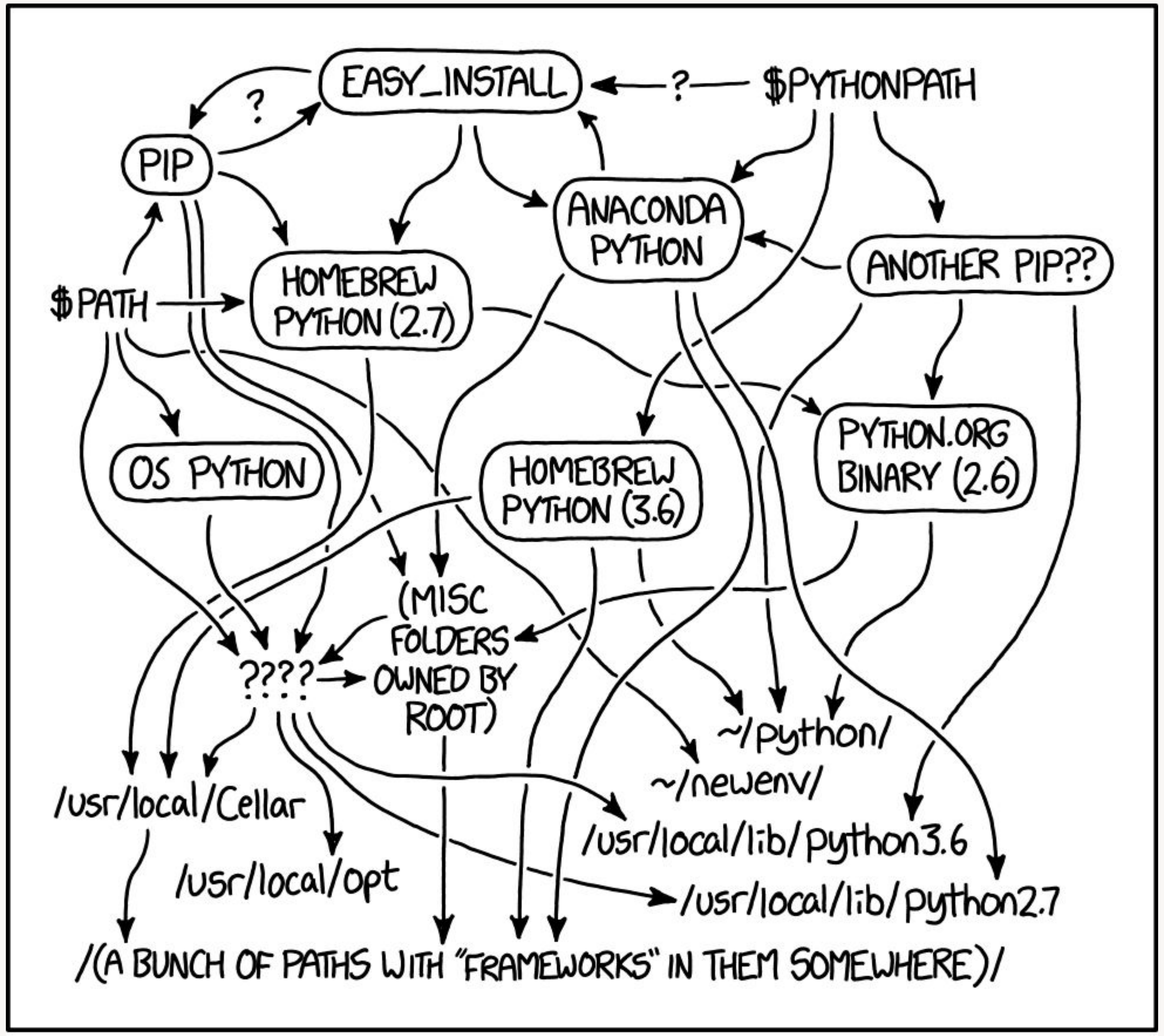
Img Source: Microtica

Model Server

Core Idea

- As an ML microservice
- Model and Inference code are wrapped in an API
- API on a Server
- Server
    - Takes prediction requests from users
    - and sends out predictions





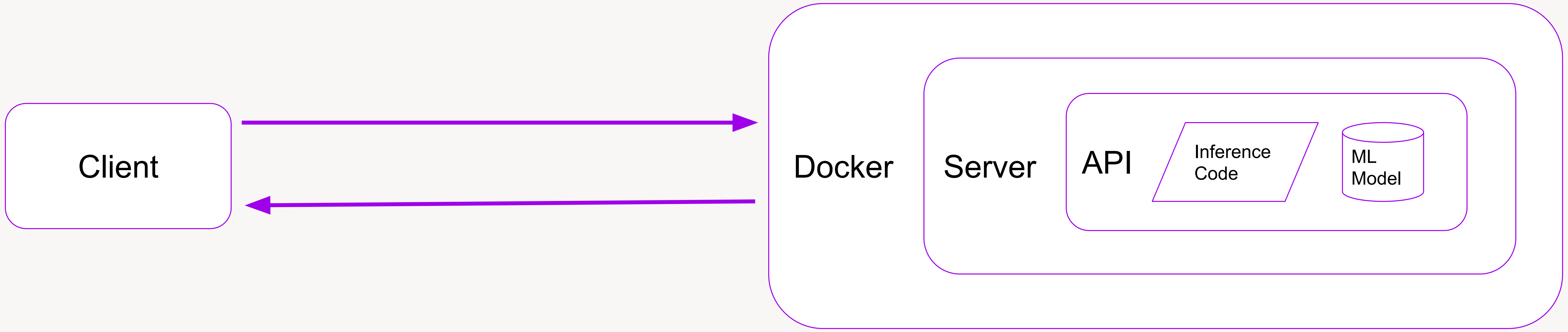
MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

Img Source: XKCD

Model Server

OS Independent

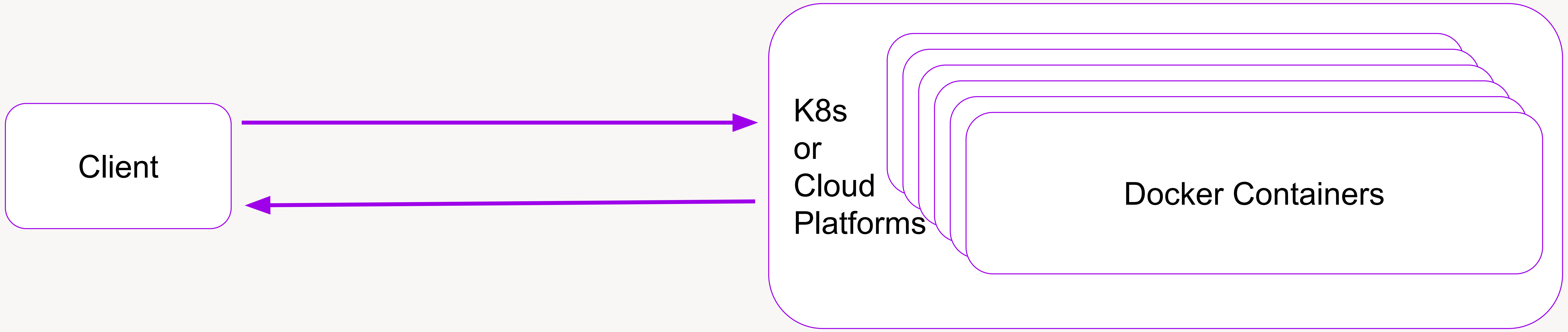
- Model Server is packaged up in a Docker Container (image)
- Docker Container
  - Encapsulates Dependencies
  - Ease of Deployments on various OS
  - Repeatable
  - Scalable



Model Server

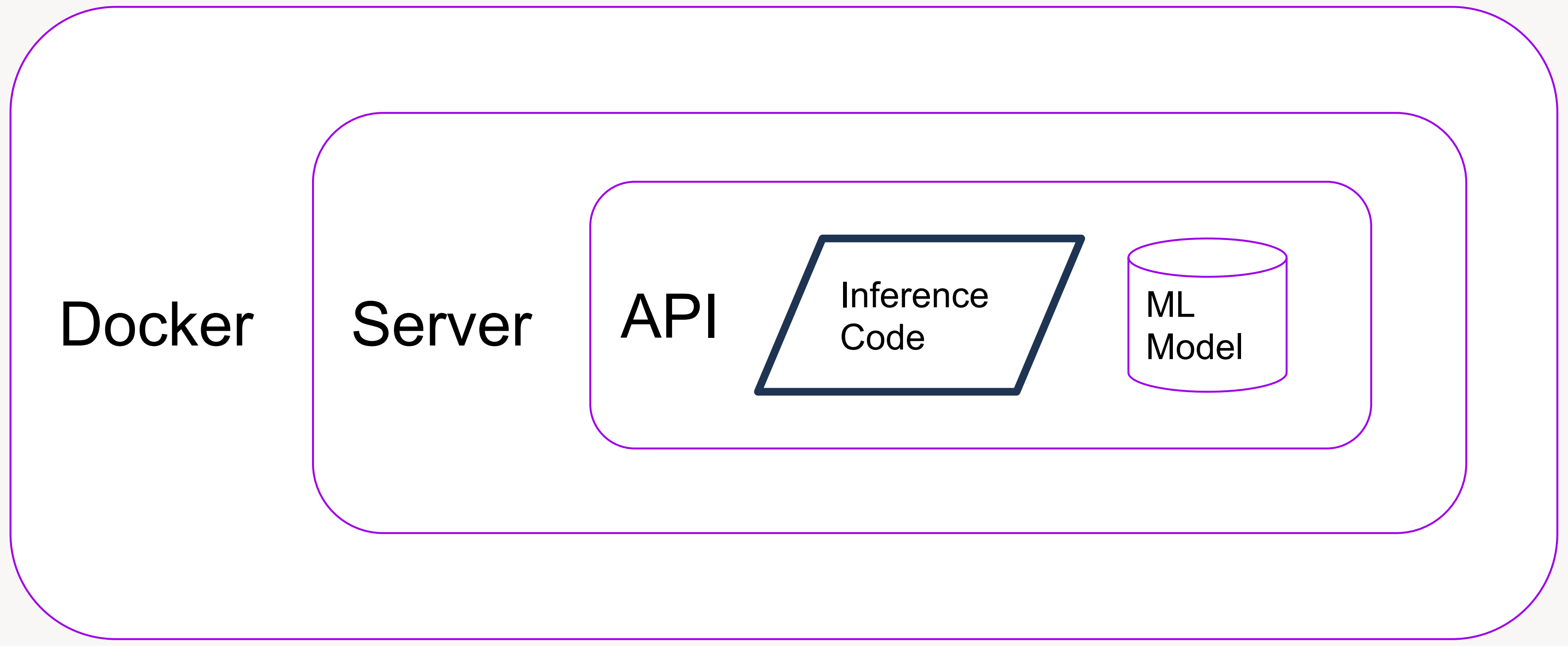
At Scale

- Deploy on Kubernetes or Cloud Managed Platforms
  - Complex deployment scenarios
  - Efficient use of computational resources
  - Hardware Agnostic
  - Autoscaling
  - Fault tolerant




TALK IS CHEAP
SHOW ME THE CODE
imgflip.com

Inference Code



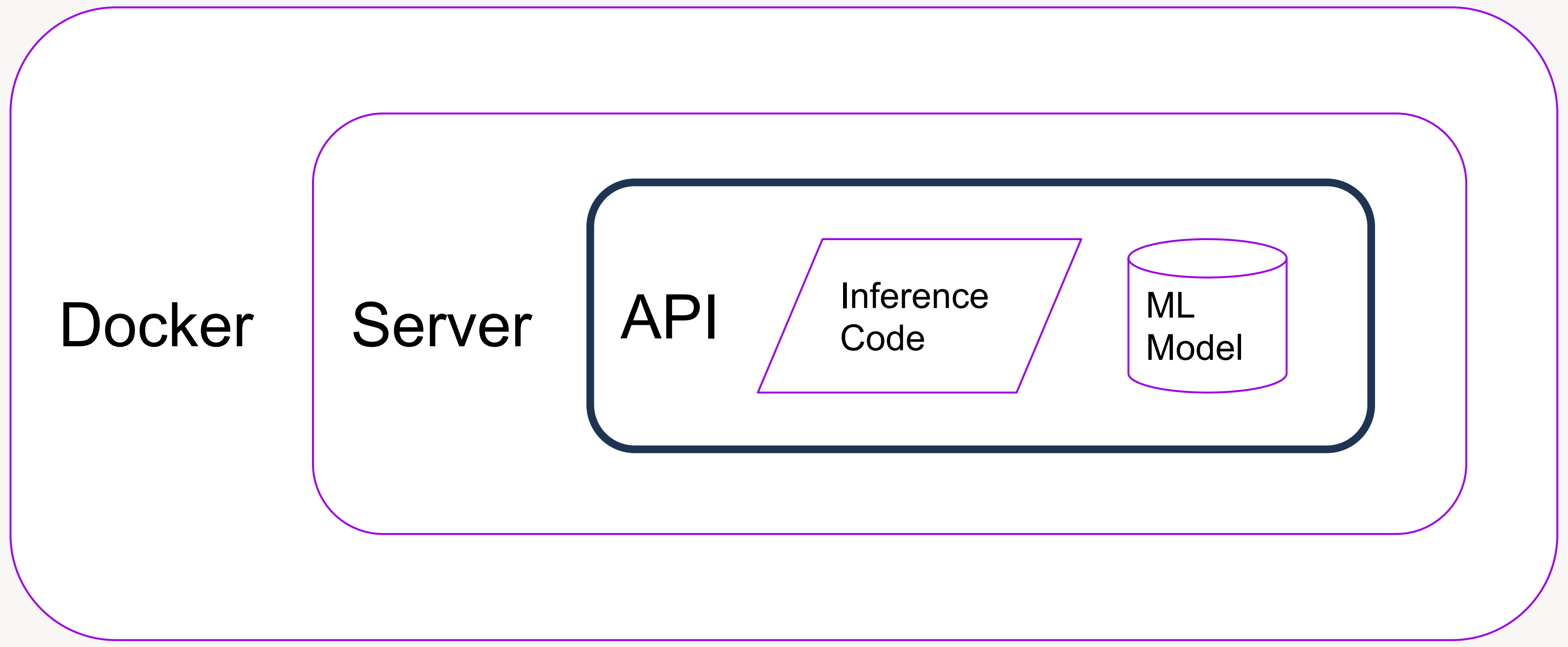
Inference Code

```
# predict.R
library(randomForest)

get_predictions <- function(request_data) {

    model.rf <- load_model()
    predictions<-predict(model.rf,newdata=request_data$instances)

    return (predictions)
}


load_model <- function() {

    prefix <- '/opt/ml'
    model_filename <- list.files(paste(prefix, 'model', sep='/'))[1]
    model_filepath <- paste(prefix, 'model', model_filename, sep='/')

    model.rf <- readRDS(model_filepath)

    return (model.rf)
}
```

- *get_predictions* Function
  - Take data points for Inference
  - Loads Model using *load model* function
  - Does predictions on inference data
  - Returns predictions

- *load_model* Function
  - Reads Model File
  - Loads it into a variable
  - Returns model

API

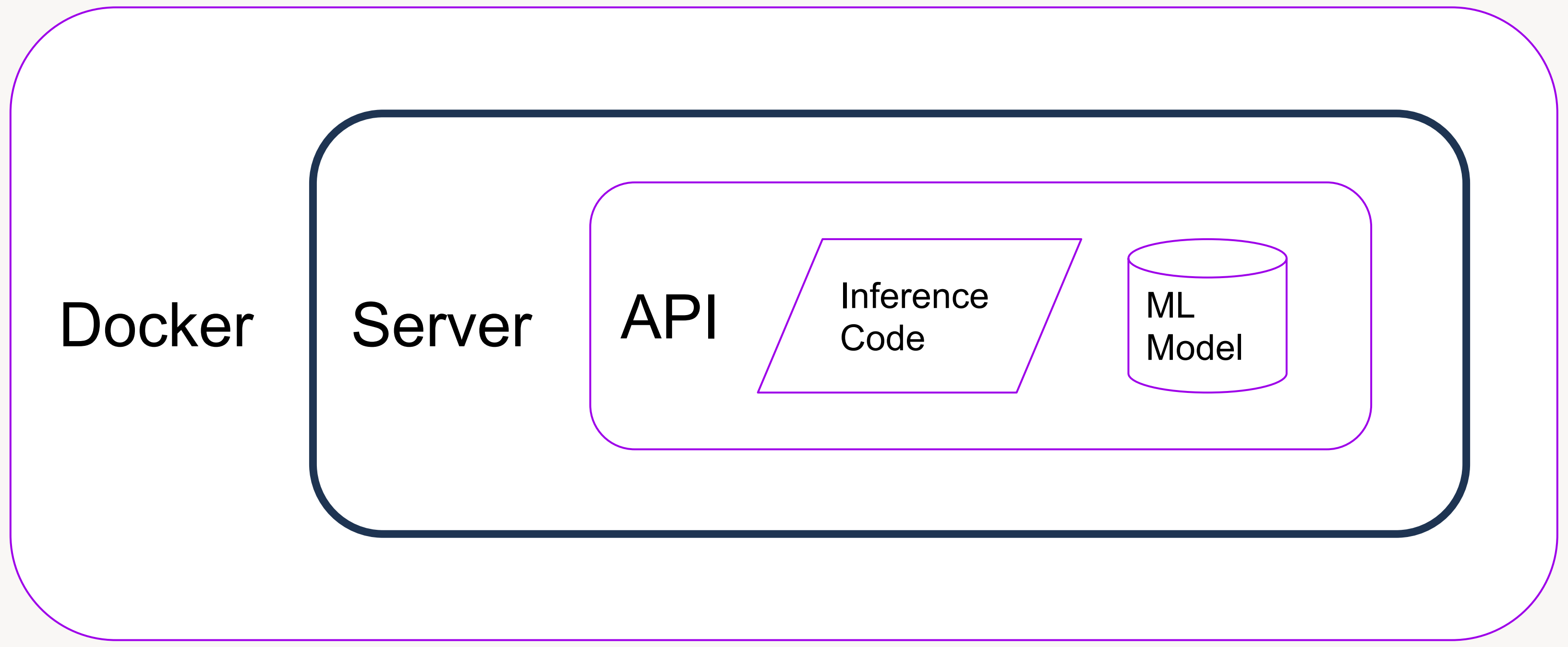


API

```r
# plumber.R

#* Ping to show server is there
#* @get /ping
function() {
  return('')
}


#* Parse input and return prediction from model
#* @parser json
#* @post /invocations
function(req){

  predictions <- get_predictions(req$body)

  return (predictions)
}
```

- Special Comments
  - Denoted by *#**
  - Define HTTP routes by *#* @method /route*
  - Other special plumber things *#* @keyword thing*
- Routes
  - */ping*
    - ↦ Health check
    - ↦ Returns empty string
  - */invocations*
    - ↦ Calls get_predictions function
    - ↦ Returns predictions

Server

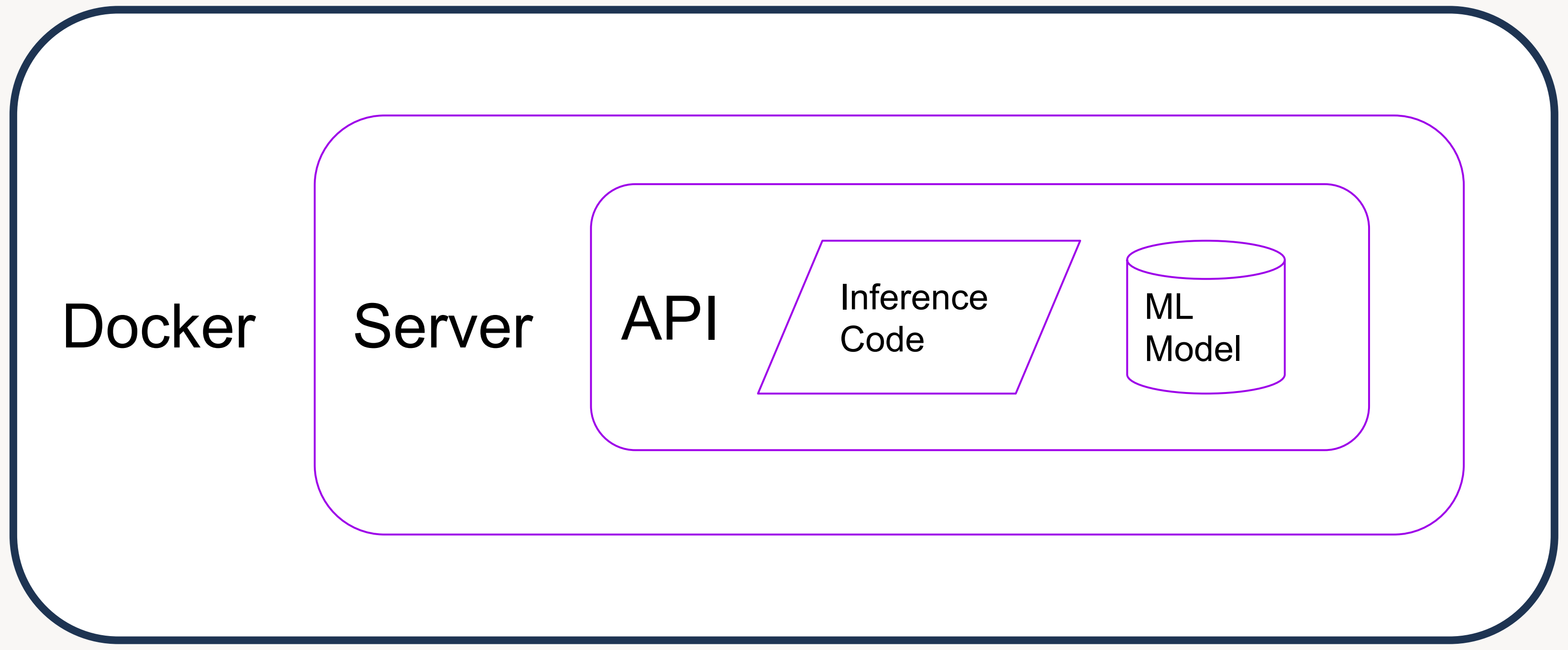


Server

```r
#app.R
library(plumber)

# sourcing predict.R
source("predict.R")

# Setup serving function
serve <- function() {
    app <- plumb(file='plumber.R', dir=".")
    app$run(host='0.0.0.0', port=8080)
}

# Run at start-up
args <- commandArgs()
if (any(grepl('serve', args))) {
    serve()
}
```

- *serve()* function
  - *plumb()* processes and loads api defined in plumber.R
  - *run()*starts the web server on specified address/port
- app.R
  - Expects *serve* keyword as a command line argument
  - Start with following command *Rscript app.R serve*

Docker Image



Docker Image

```
# Base image
FROM rocker/ml:4.0.2

WORKDIR prodr

EXPOSE 8080

# Copy stuff
COPY app.R app.R
COPY plumber.R plumber.R
COPY predict.R predict.R

COPY ./models/iris_rf/iris_model.rds /opt/ml/model/iris_model.rds

# Install packages
RUN install2.r --error \
    argparser \
    randomForest \
    plumber

# Start server
CMD ["serve"]
ENTRYPOINT [ "/usr/local/bin/Rscript", "--no-save", "app.R" ]
```

- *rocker/ml* Base image
  - comes preinstalled with R, Rstudio, Tensorflow, tidyverse etc..
  - Checkout rocker project
- *COPY* Copies relevant files
- *RUN* Installs additional required packages
- *ENTRYPOINT CMD* Starts plumber api server
- *Checkout Containerit


PITFALLS
PITFALLS EVERYWHERE
imgflip.com

!Things to consider

API

- Can lead to Non standard APIs → Keep'em Simple, Standardise APIs across R ML services
- API and Inference code can become tightly coupled → Keep them as separate modules

Server

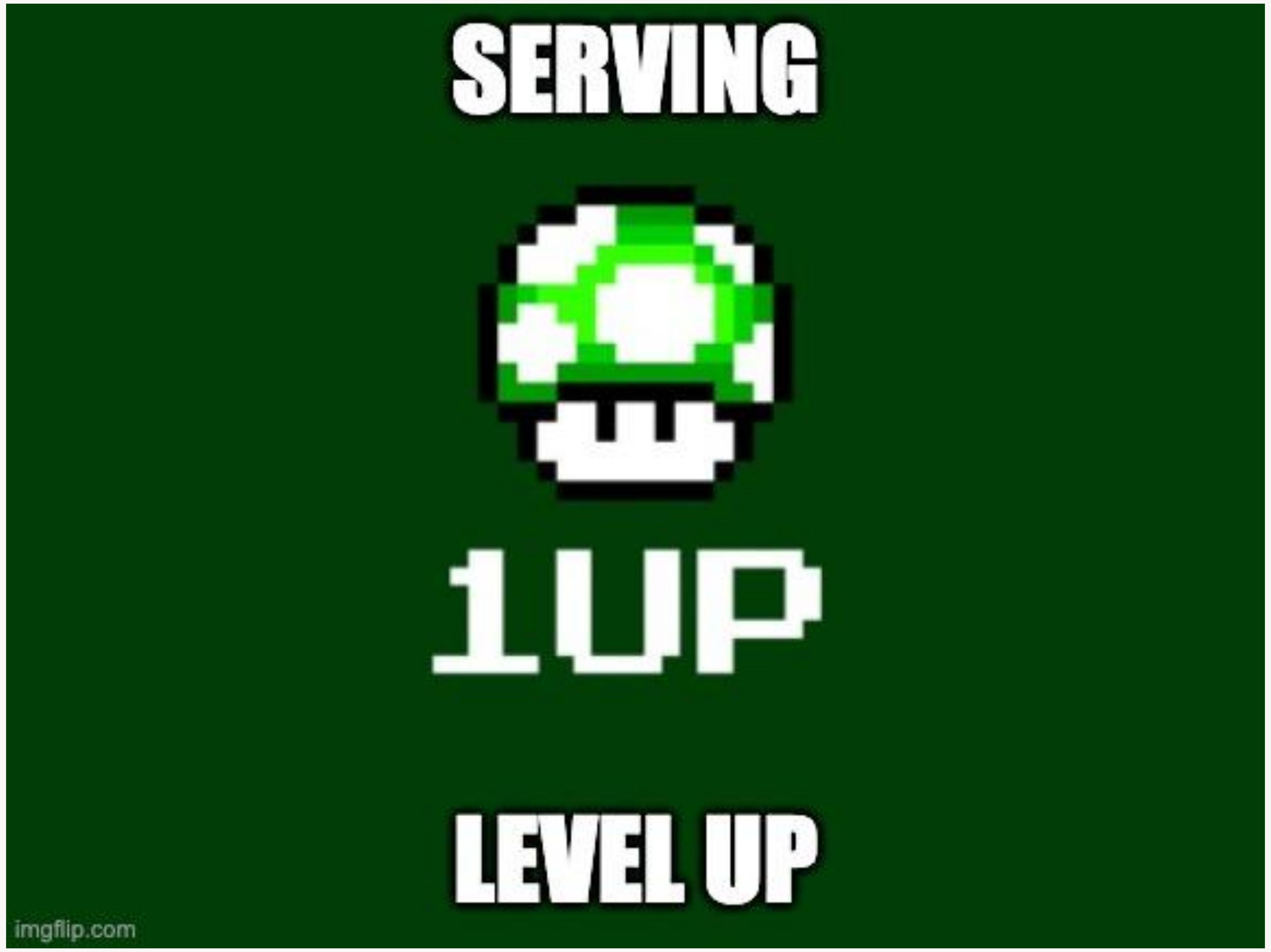
- Plumber server/application can go down without much visibility to upstream processes → Health checks (ping) , Autorestarts

Docker Image

- Prebuilt images can get bloated (~7GB) → Build your own images

Computationally Inefficient

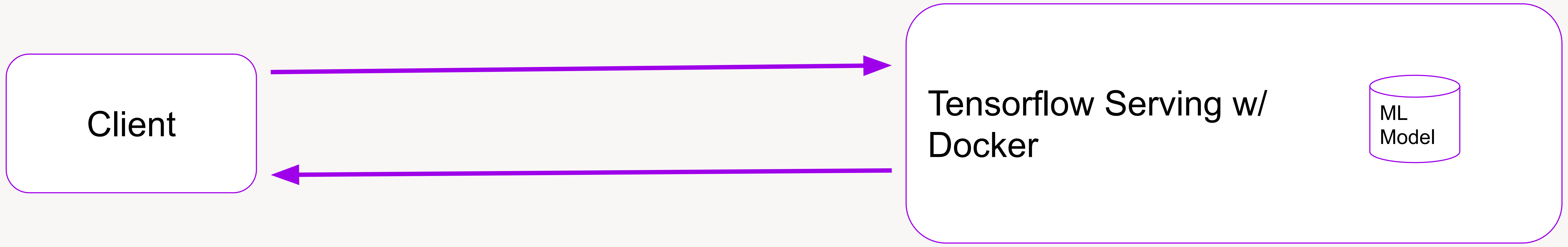
- Single threaded → Multiple lightweight containers with an external load balancer


SERVING
1UP
LEVEL UP
imgflip.com

Model Server

Tensorflow Serving

- Off the shelf ← Only requires model file*
- *Models exported by tensorflow; TF R Wrapper
- Consistent APIs (gRPC, REST) w/ separation between API code and Models
- Out of the box support for
  - health checks, model versioning, batching, etc..
  - complex deployment scenarios w/ multiple versions



TF serving

```
docker run \
    -p 8501:8501 \
    --mount type=bind,source=$(pwd)/models/mnist_tf,target=/models/mnist/1 \
    -e MODEL_NAME=mnist \
    -t tensorflow/serving
```

```
curl \
--data "@./data/curl_data_tfx.json" \
-X POST http://localhost:8501/v1/models/mnist:predict
```

- *tensorflow/serving* target image
  - forwarding local traffic w/ *-p*
  - mounting model onto container w/ *--mount*
  - model name as environment variable w/ *-e*
  - naming docker container w/ *--name*

- http request with *curl*
  - *POST* type
  - *--data* from file
  - On *localhost*


BRACE YOURSELF
IT'S A LIVE DEMO
imgflip.com

Model Servers

Other Options

- torch serve
  - Docker + Server + API + Inference Code
  - Language agnostic as long as models are serialised/archived using inbuilt functions

- seldon core
  - Docker + Server + API + Inference Code
  - R wrapper in alpha release
  - need to define few files according to seldon standards

- mlflow serving
  - Docker + Server + API + Inference Code
  - R API
  - models trained and saved/registered as an MLFlow model

Compute

Deployment Targets

- Container as Service
  - AWS Elastic Container Service
  - Azure Container Instances
  - Google Cloud Run

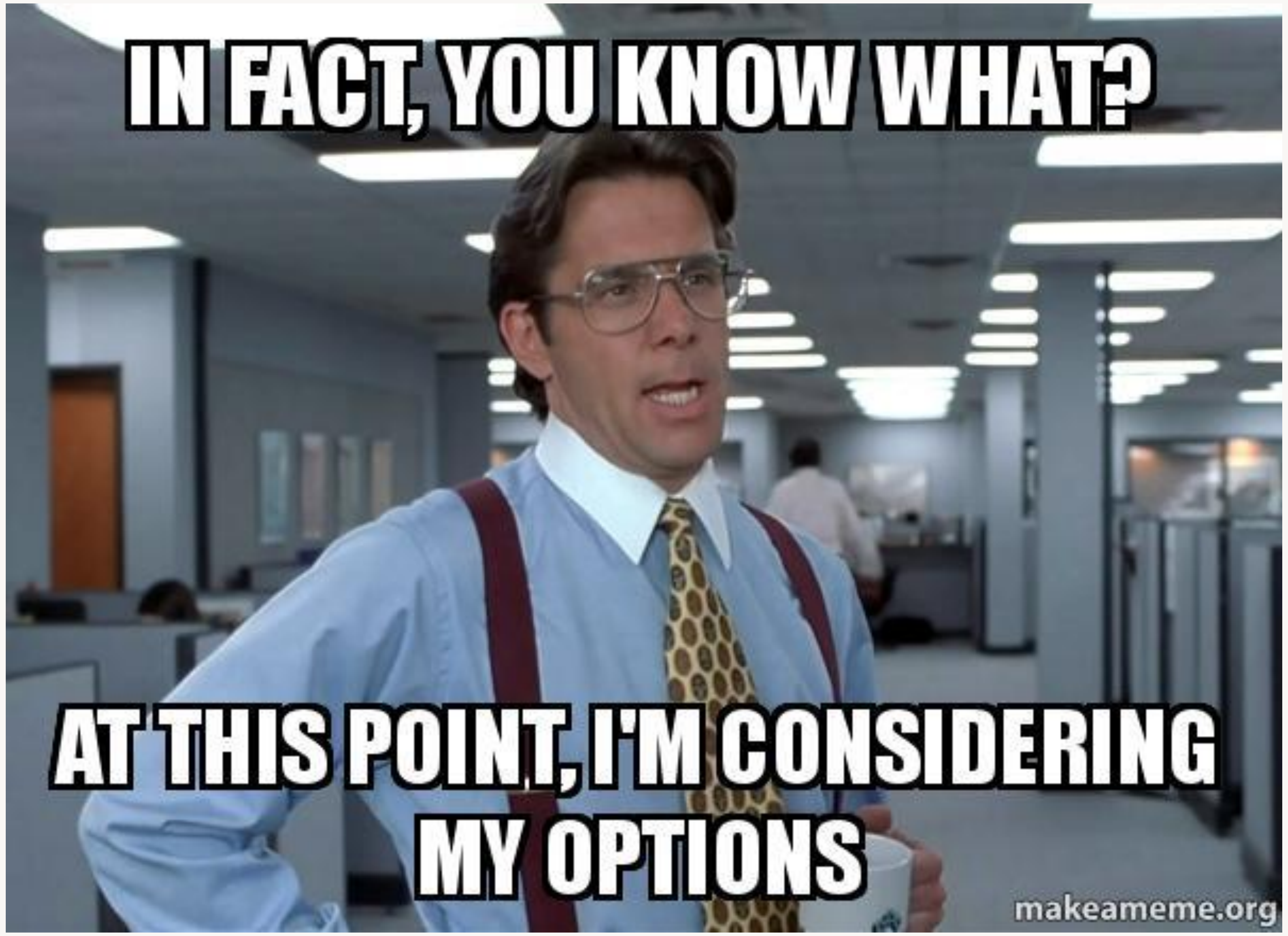
- Kubernetes
  - Native Deployment and expose as a Service
  - Kubeflow - KFServing, TF Serving, Seldon Core Serving

- Cloud ML Platforms
  - Azure ML BYO Docker
  - AWS Sagemaker BYO Docker

Platform as a Service

Model Server + Compute

- Typically
  - ↦ Limited Support for Libraries / Models - Even more Limited for R
  - ↦ Serverless - You don't have to manage the underlying server
  - ↦ Scalable - Zero to world domination
  - ↦ Pay per use - For computational resources when serving
  - ↦ Pricey - Price markup on top of underlying compute
- Typical Deployment Flow
  1. Upload model to Storage
  2. Register Model Resource on the Platform
  3. Supply Inference Code / Script
  4. Define Deployment / Endpoint Config
- Offered by all major cloud providers
  - ✓ AWS Sagemaker
  - ✓ Azure Machine Learning
  - ✓ Google AI Platform


IN FACT, YOU KNOW WHAT?
AT THIS POINT, I'M CONSIDERING MY OPTIONS
makeameme.org

Model Servers

Comparison

	DIY Model Server	Out of the Box Servers	PaaS
R Support	□ Available	□ Limited	🔴 Very Limited
Library Support	□ Any library / model type	□ Often Limited	□ Limited
Learning Curve	🔴 High	□ Medium to High	□ Medium
Development Costs	🔴 High to Medium	□ Medium to High	□ Medium
Maintenance Costs	🔴 High to Medium	□ Medium to Low	□ Low to Medium
Price	□ Can be Low	□ Can be Low	🔴 Can be High
Complex Scenario Deployments	🔴 Usually w/ Manual Configs	□ Usually out of the box support	□ Out of the box
Monitoring	🔴 Manual	□ Usually out of the box	□ Out of the box
Choose this if you need	Flexibility	Most other cases	Low Maintenance


BUILD ML MODEL
DEPLOY ML MODEL
OPERATIONALISED ML DEPLOYMENTS
imgflip.com

Operationalise

Overview

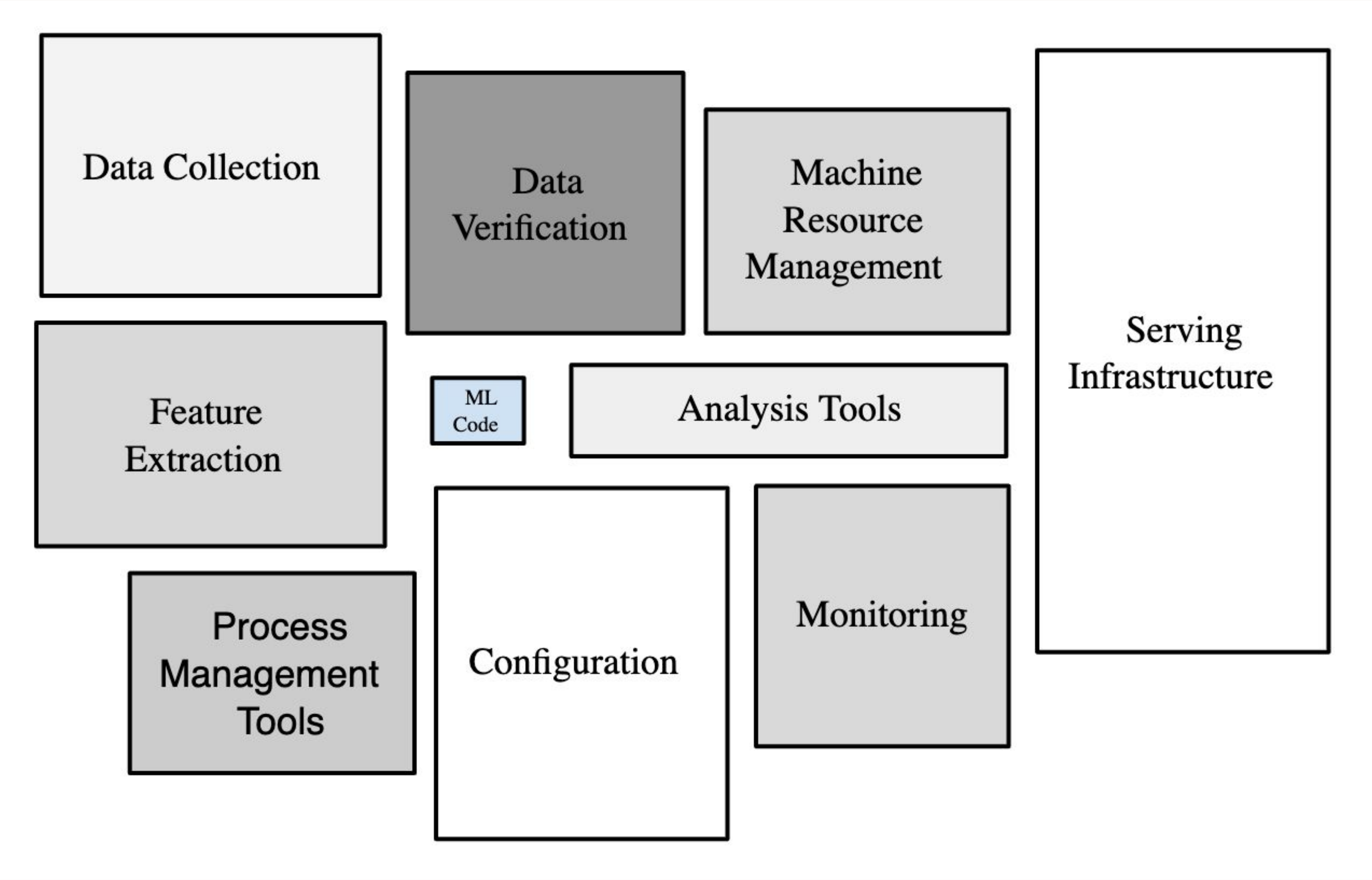
1. Version Control
   a. Code w/ Git
   b. Data w/ DVC, Dolt, pachyderm, Kubeflow - Rok
   c. Models, Experiments w/ TFX, MLFlow, KubeFlow, DVC, pachyderm
2. Testing w/ usethis, testthat
   Test Driven ML by Tim Fist
3. CI/CD
   - Automate Training, Deployment and Testing w/ CI/CD
   - Available Tools include → Gitlab CI, Github Actions, Jenkins, Travis etc..
4. Monitoring
   Concept Drift with Eike Germann

* w/ Cloud Provider Custom Solutions

ML Pipelines → Often we combine a few steps in ML Workflow into a Pipelines; Brownbags by Xin and Lucas

Machine Learning System

Overview



Source: Sculley et al., Hidden Technical Debt in Machine Learning Systems


MY TWO CENTS
PLEASE USE PYTHON IF YOU CAN!
imgflip.com

Please use Python

Friendly Suggestion

- More and More ML / DS libraries, tools are being developed in/for Python
- IMO, It's easier to find answers in the Python world when stuck
- IMO, Practices / Patterns to accomplish a task are more standardised in Python world
- First class Support for Python by Cloud Providers, Open Source Tools, ML/DS Platforms
- Python is usually faster and more efficient than R for a given task


I HOPE YOU GOT SOMETHING OUT OF
THIS BROWNBAG
CHEERS!
imgflip.com

References

1. Inspired by
   a. MLOps for non-DevOps folks, a.k.a. "I have a model, now what?! - Hannes Hapke - ML4ALL 2019
   b. Building and Deploying robust APIs in R using Plumber - James Blair
2. Demo Repo: https://github.com/suryaavala/prodr
3. Docs
   a. Docker, containerit
   b. Plumber
   c. ML Servers
      i. Tensorflow Serving with Docker, Tensorflow R package
      ii. torch for R
      iii. Seldon R language Wrapper
      iv. MLFlow R API
      v. PaaS
         1. AWS Sagemaker
         2. Azure Machine Learning
         3. Google AI Platform
4. Other Useful Stuff
   a. Shiny in Production workshop

eliiza