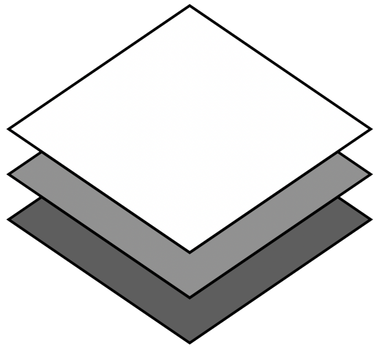
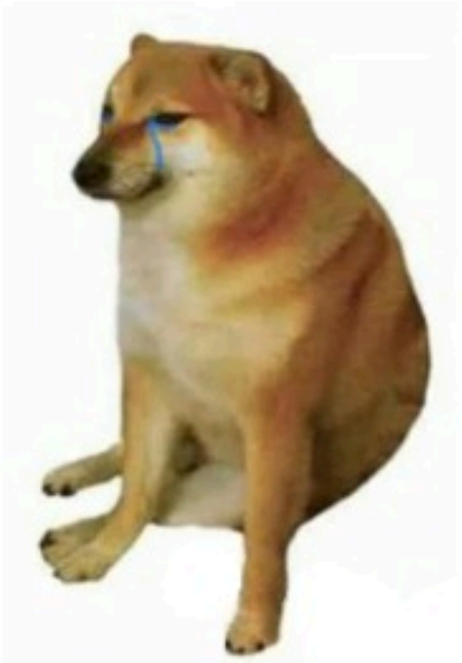


Deployment & Monitoring

ML model in production

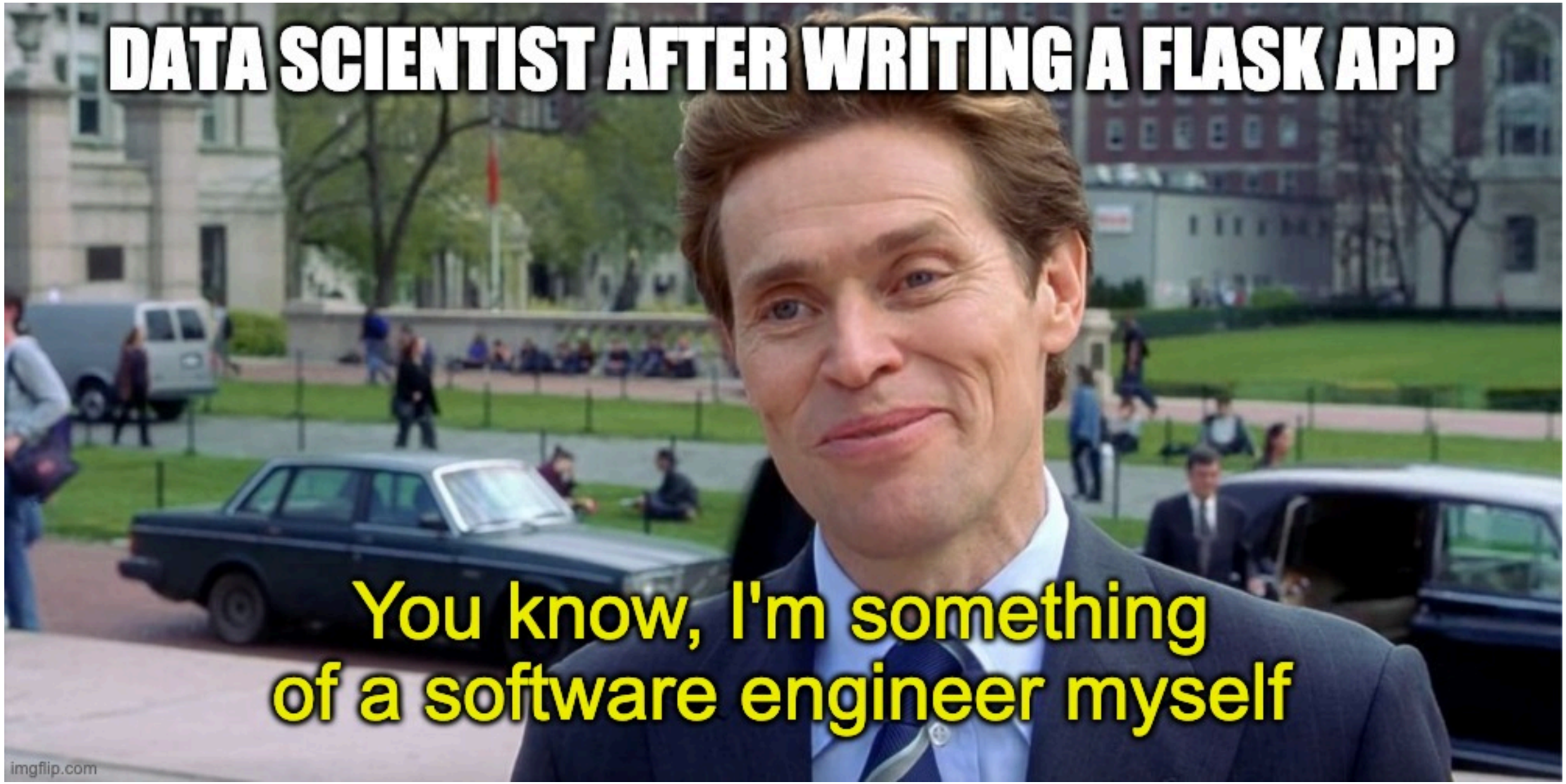
Scales to meet the
demands of 1,000s of users

ML model in a notebook



I promise it works
you just ran the
cells in the wrong order

imgflip.com


DATA SCIENTIST AFTER WRITING A FLASK APP
You know, I'm something
of a software engineer myself
imgflip.com

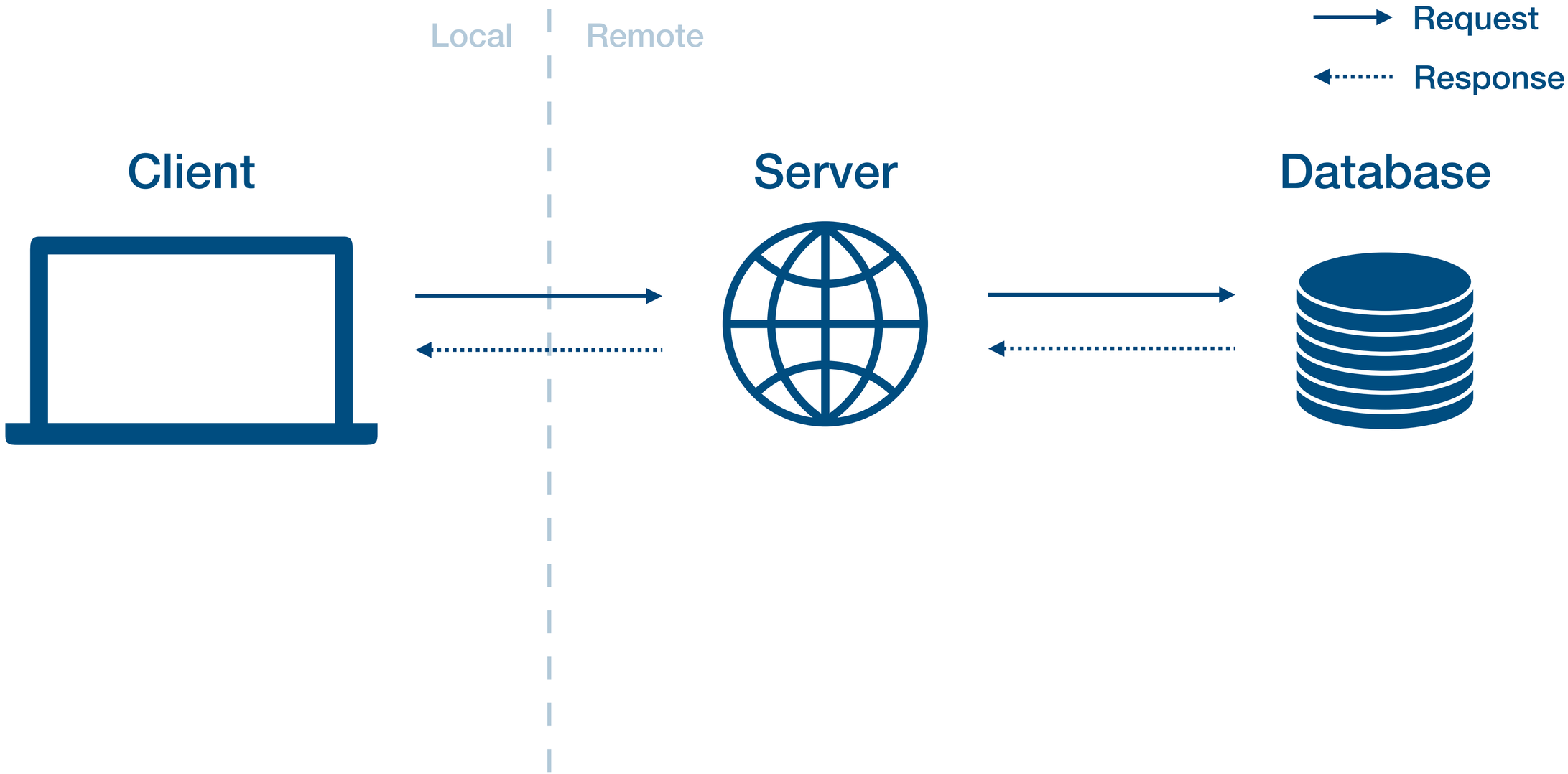
Today

- Deployment (AKA, how to get your model into production)
- Monitoring (AKA how to keep it healthy once it's there)

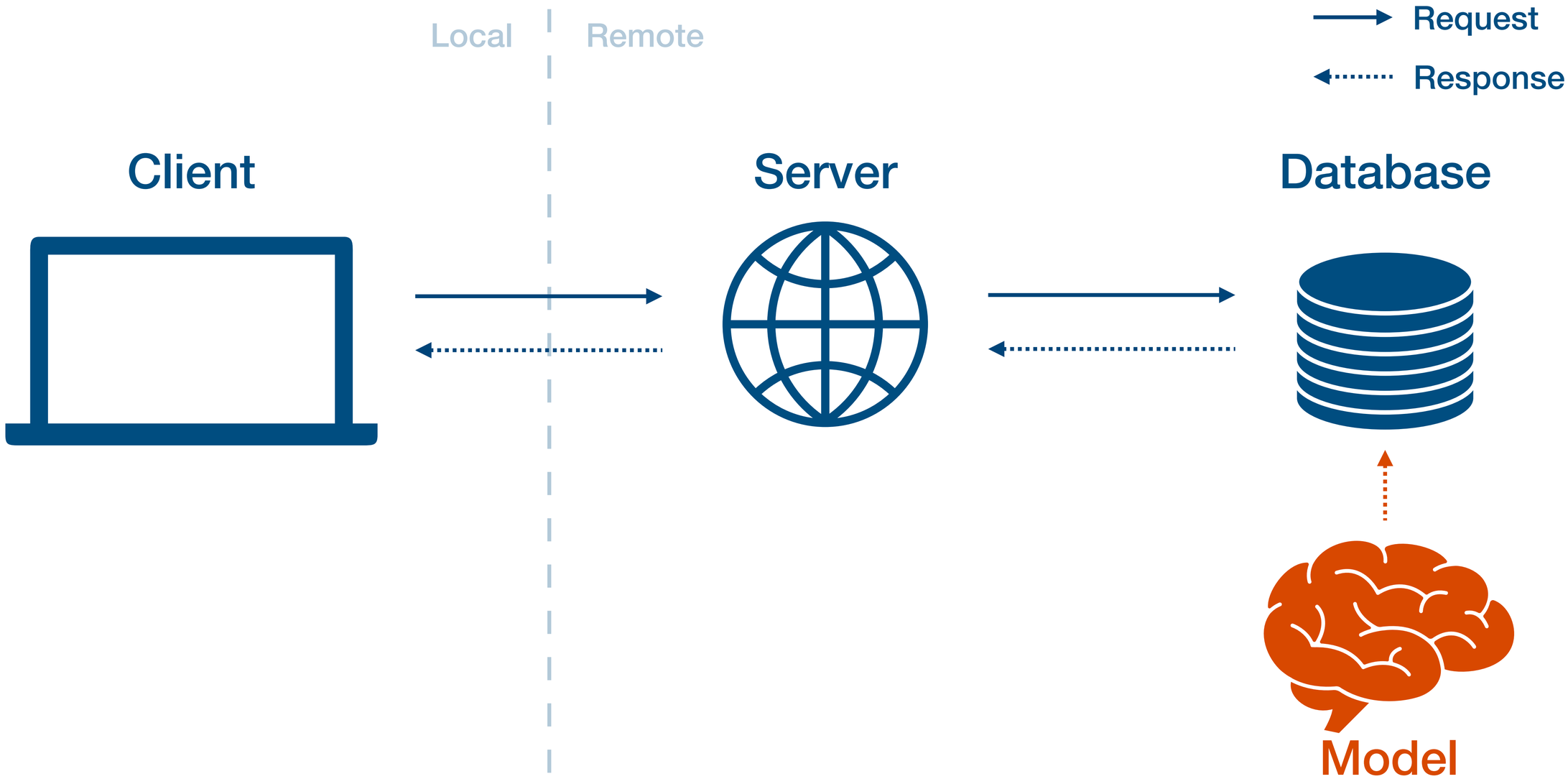
Today

- **Deployment**
- Monitoring

Where in the architecture should your model go?



Batch prediction



Batch prediction

- Periodically run your model on new data and cache the results in a database

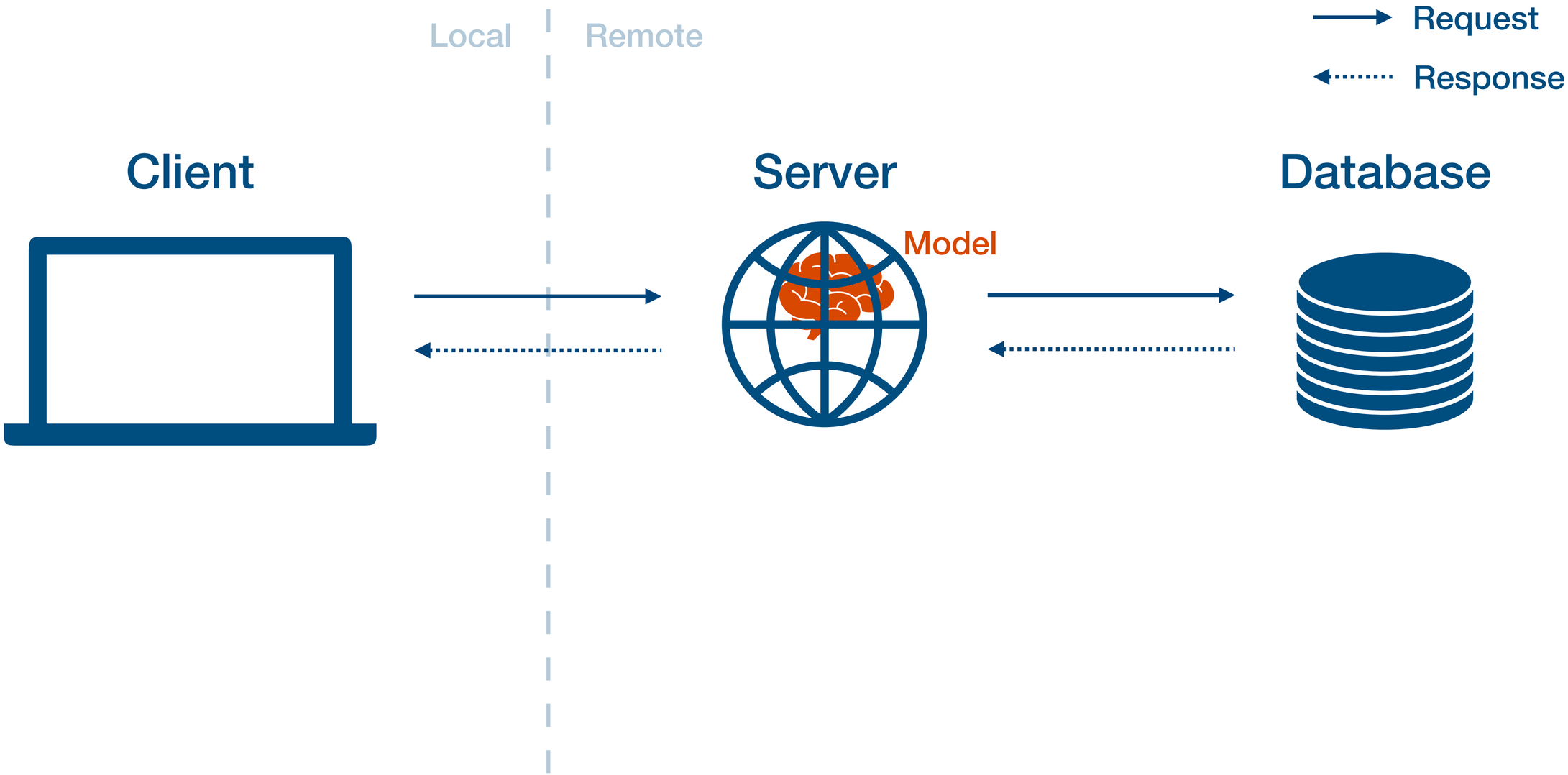
- Works if the universe of inputs is relatively small (e.g., 1 prediction per user)

Batch prediction

Pros	Cons
• Simple to implement • Relatively low latency to the user	• Doesn't scale to complex input types • Users don't get the most up-to-date predictions • Models frequently become "stale", which can be hard to detect

Questions?

Model-in-service



Model-in-service

- Package up your model and include it in your deployed web server
- Web server loads the model and calls it to make predictions

Model-in-service

Pros

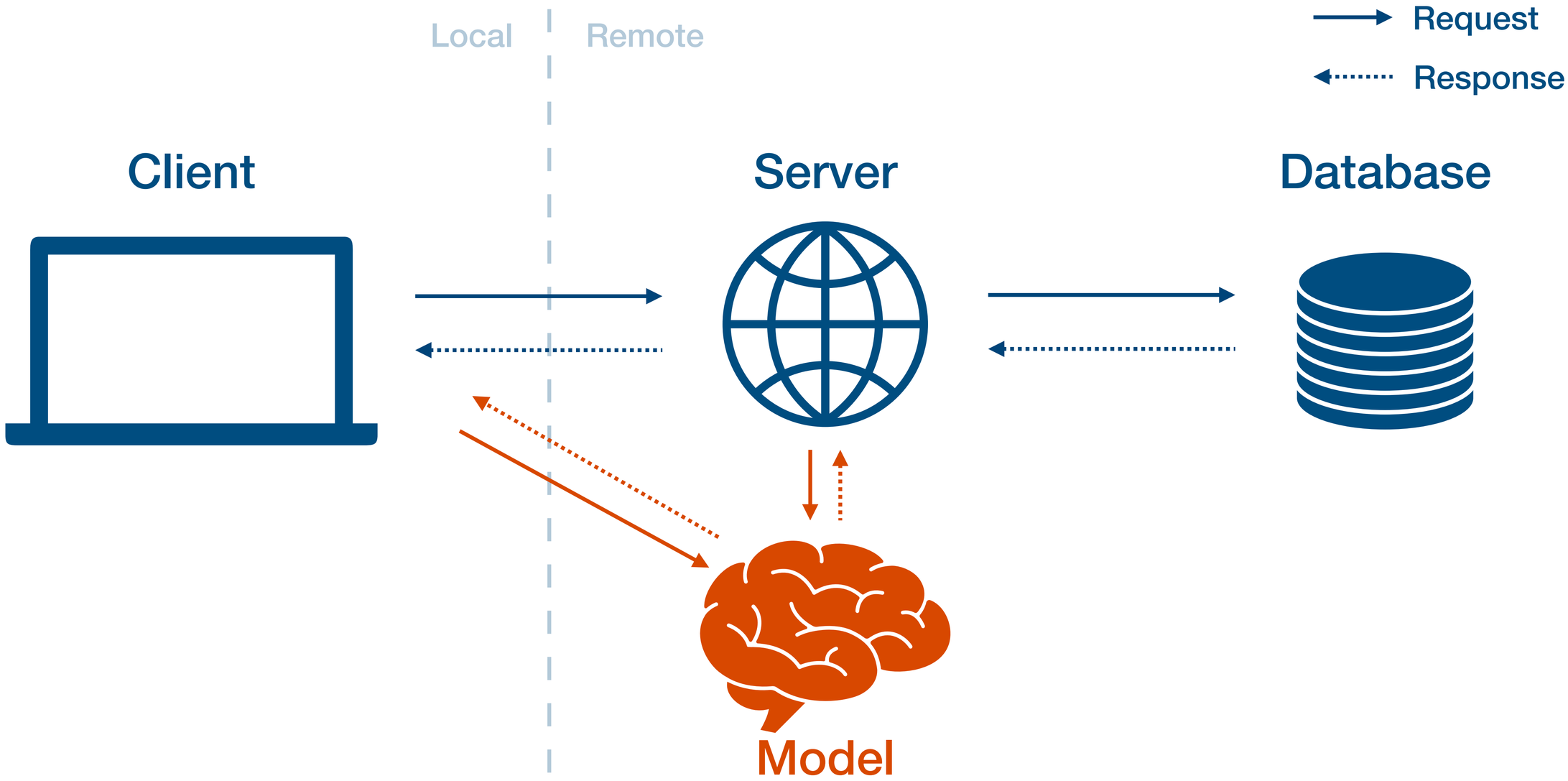
- Re-uses your existing infrastructure

Cons

- Web server may be written in a different language
- Models may change more frequently than server code
- Large models can eat into the resources for your web server
- Server hardware not optimized for your model (e.g., no GPUs)
- Model & server may scale differently

Questions?

Model-as-service



Model-as-service

- Run your model on its own web server
- The backend (or the client itself) interact with the model by making requests to the model service and receiving responses back

Model-as-service

Pros

- Dependability — model bugs less likely to crash the web app
- Scalability — choose optimal hardware for the model and scale it appropriately
- Flexibility — easily reuse a model across multiple apps

Cons

- Can add latency
- Adds infrastructural complexity
- Now you have to run a model service…

Building a model service: the basics

- REST APIs
- Dependency management
- Performance optimization
- Horizontal scaling
- Deployment
- Managed options

Building a model service: the basics

- **REST APIs**
- Dependency management
- Performance optimization
- Horizontal scaling
- Deployment
- Managed options

REST APIs

- Serving predictions in response to canonically-formatted HTTP requests
- There are alternatives like GRPC (which is actually used in tensorflow serving) and GraphQL (not terribly relevant to model services)

REST API example

```
josh@Joshs-MacBook-Pro ~ % curl -X POST --url https://api.fullstackdeeplearnin
g.com/predict -d '{"name": "josh", "year":2021, "subject": "ml", "pages": 20}'
```

Formats for requests and responses

- Sadly, no standard yet

Google Cloud

gcloud | REST API

Make each instance an item in a JSON array, and provide the array as the `instances` field of a JSON object. For example:

```
{"instances": [
  {"values": [1, 2, 3, 4], "key": 1},
  {"values": [5, 6, 7, 8], "key": 2}
]}
```

Azure

The REST API expects the body of the request to be a JSON document with the following structure:

JSON | Copy

```
{
    "data":
        [
            <model-specific-data-structure>
        ]
}
```

AWS Sagemaker

```
{
  "instances": [
    {"in0": [863], "in1": [882]},
    {"in0": [278], "in1": [311]},
    {"in0": [705], "in1": [578]},
    …
  ]
}
```

Questions?

Building a model service: the basics

- REST APIs
- **Dependency management**
- Performance optimization
- Horizontal scaling
- Deployment
- Managed options

Dependency management for model servers

- Model predictions depend on **code, model weights,** and **dependencies**. All need to be present on your web server

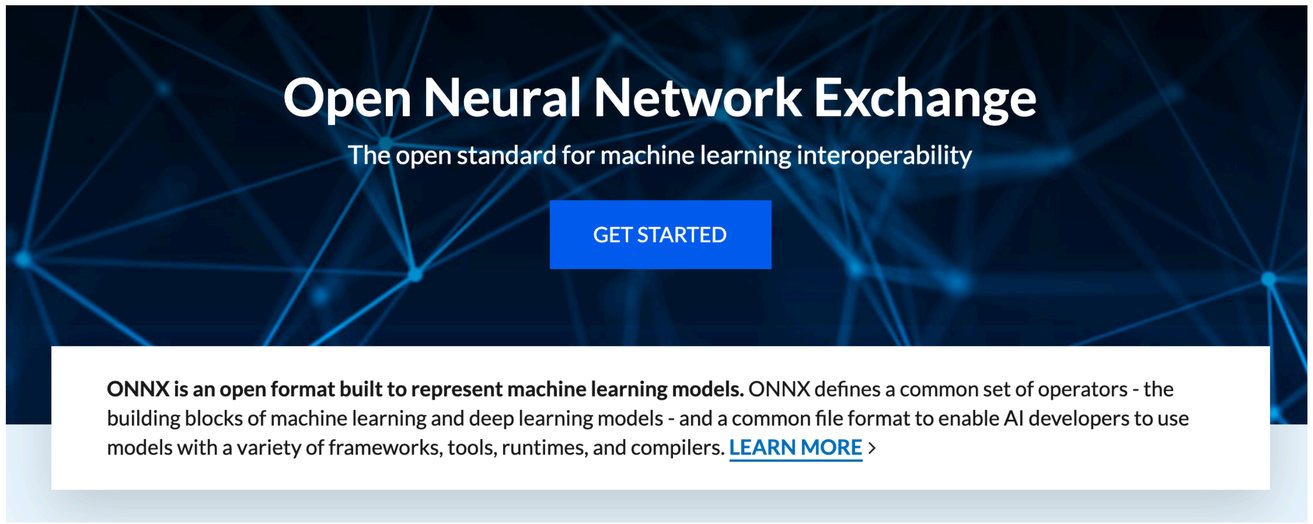
- **Dependencies cause trouble**. Hard to make consistent, hard to update, etc. Even changing a tensorflow version can change your model

- Two strategies:

  - Constrain the dependencies for your model

  - Use containers

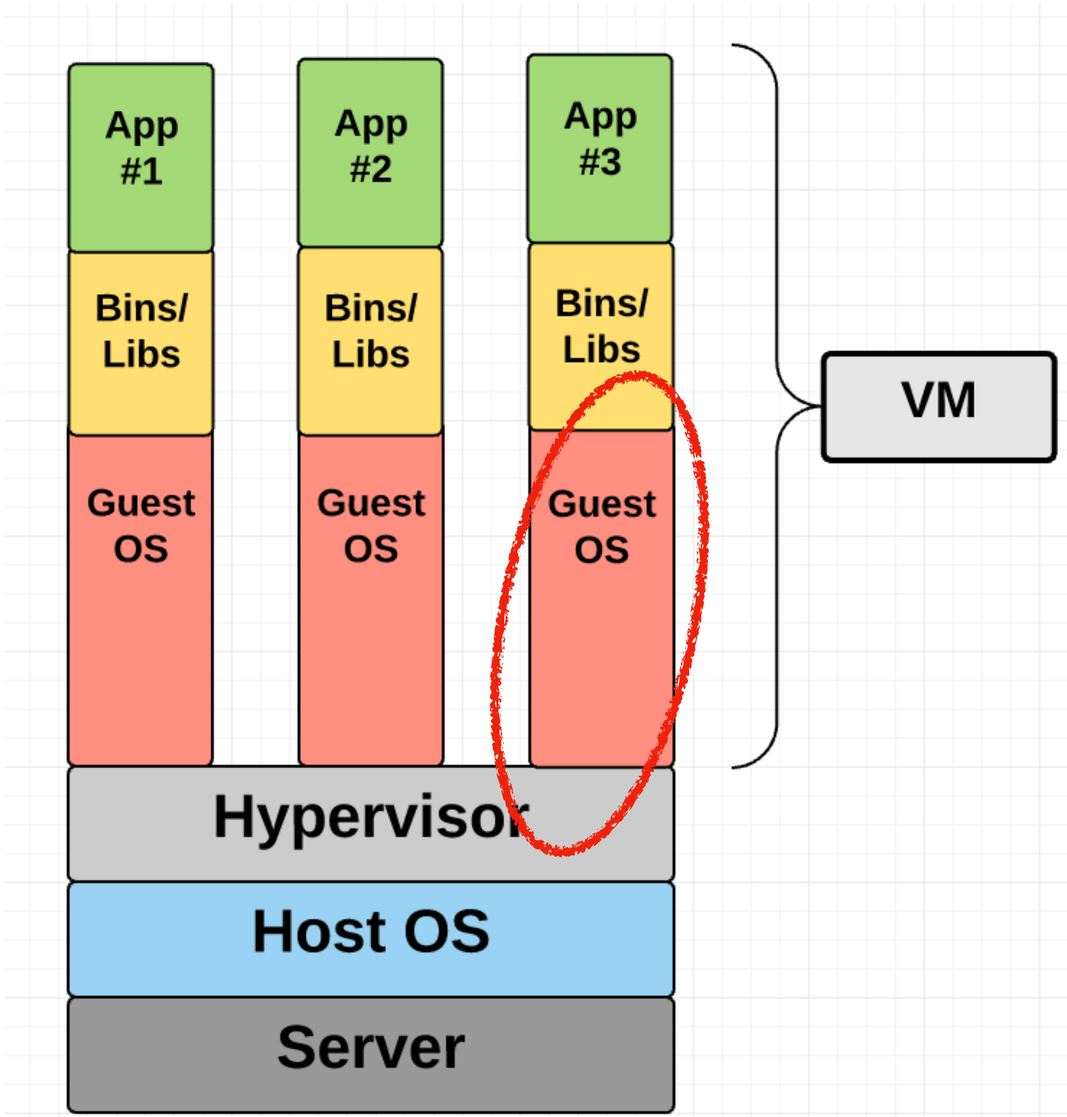
A standard neural net format: ONNX



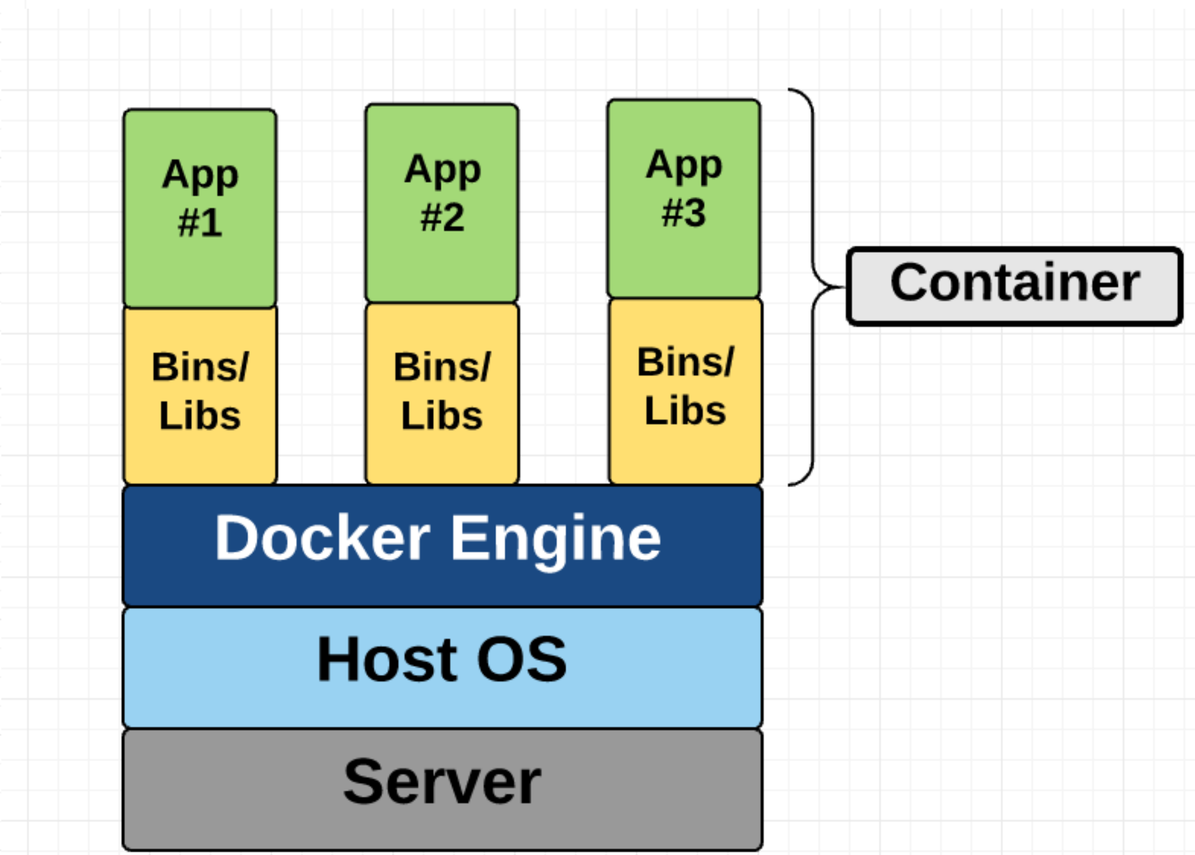
- The promise: define network in any language, run it consistently anywhere
- The reality: since the libraries change quickly, there are often bugs in the translation layer
- What about non-library code like feature transformations?

Managing dependencies with containers (i.e., Docker)

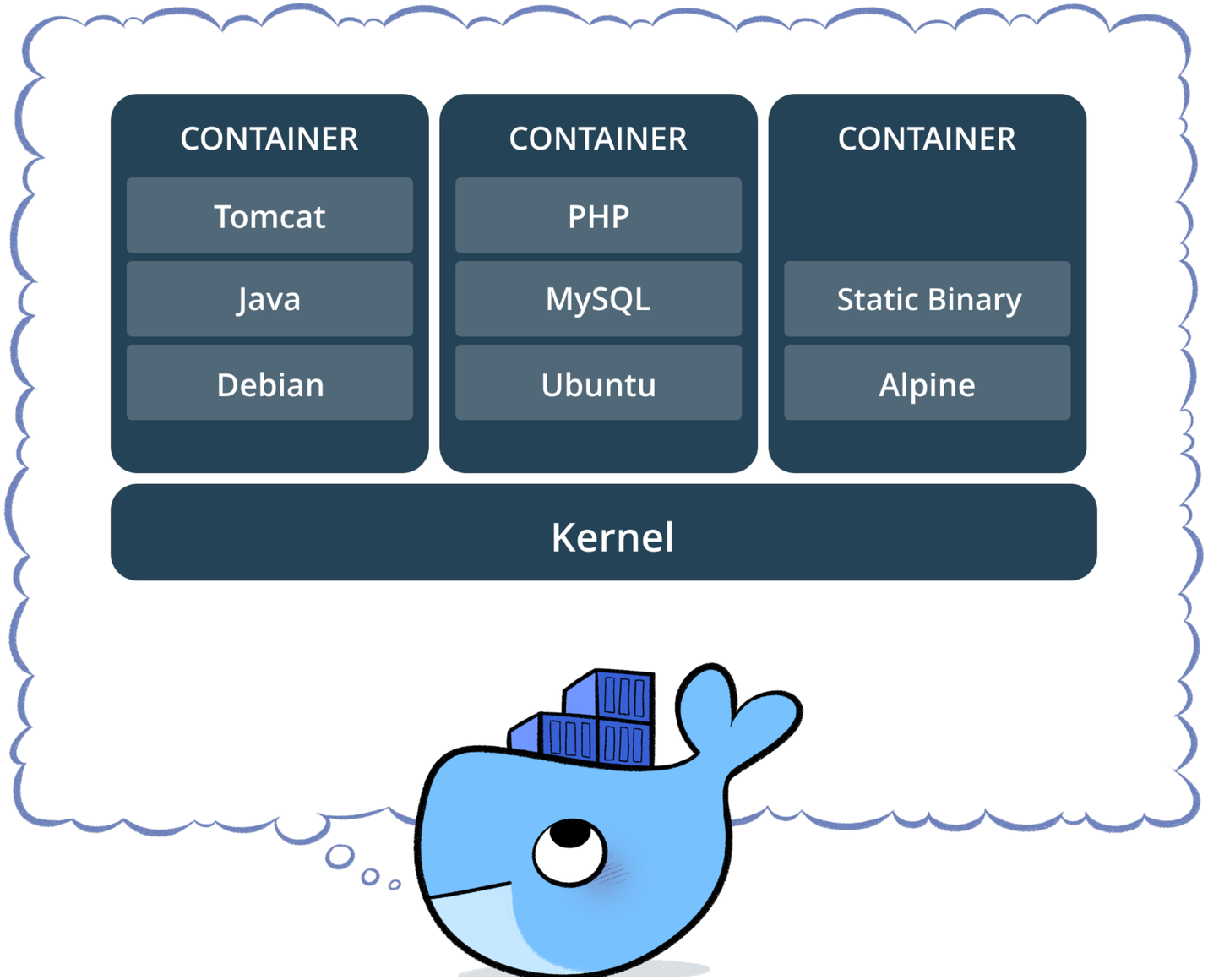
- Docker vs VM
- Dockerfile and layer-based images
- DockerHub and the ecosystem
- Kubernetes and other orchestrators



No OS -> Light weight



https://medium.freecodecamp.org/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b



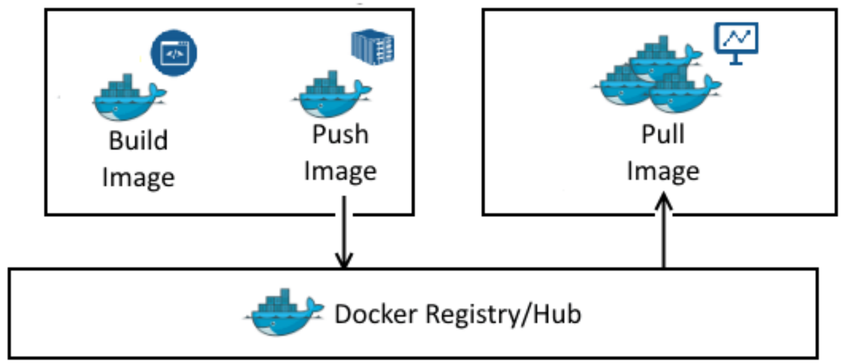
Light weight -> Heavy use

- Spin up a container for every discrete task
- For example, a web app might have four containers:
  - Web server
  - Database
  - Job queue
  - Worker

https://www.docker.com/what-container

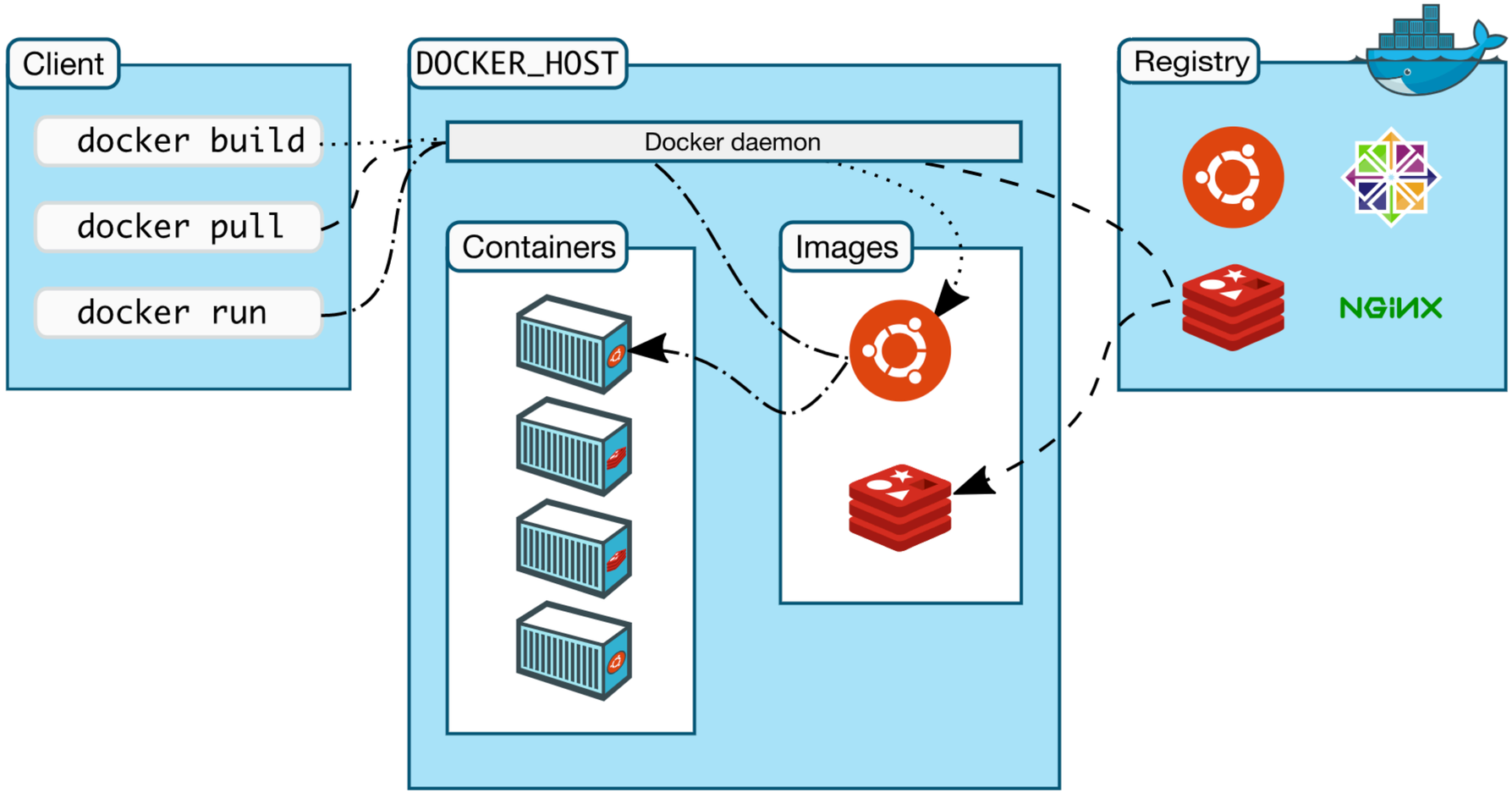
Dockerfile

```
# Use an official Python runtime as a parent image
FROM python:2.7-slim

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
ADD . /app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 80 available to the world outside this contain
EXPOSE 80

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "app.py"]
```



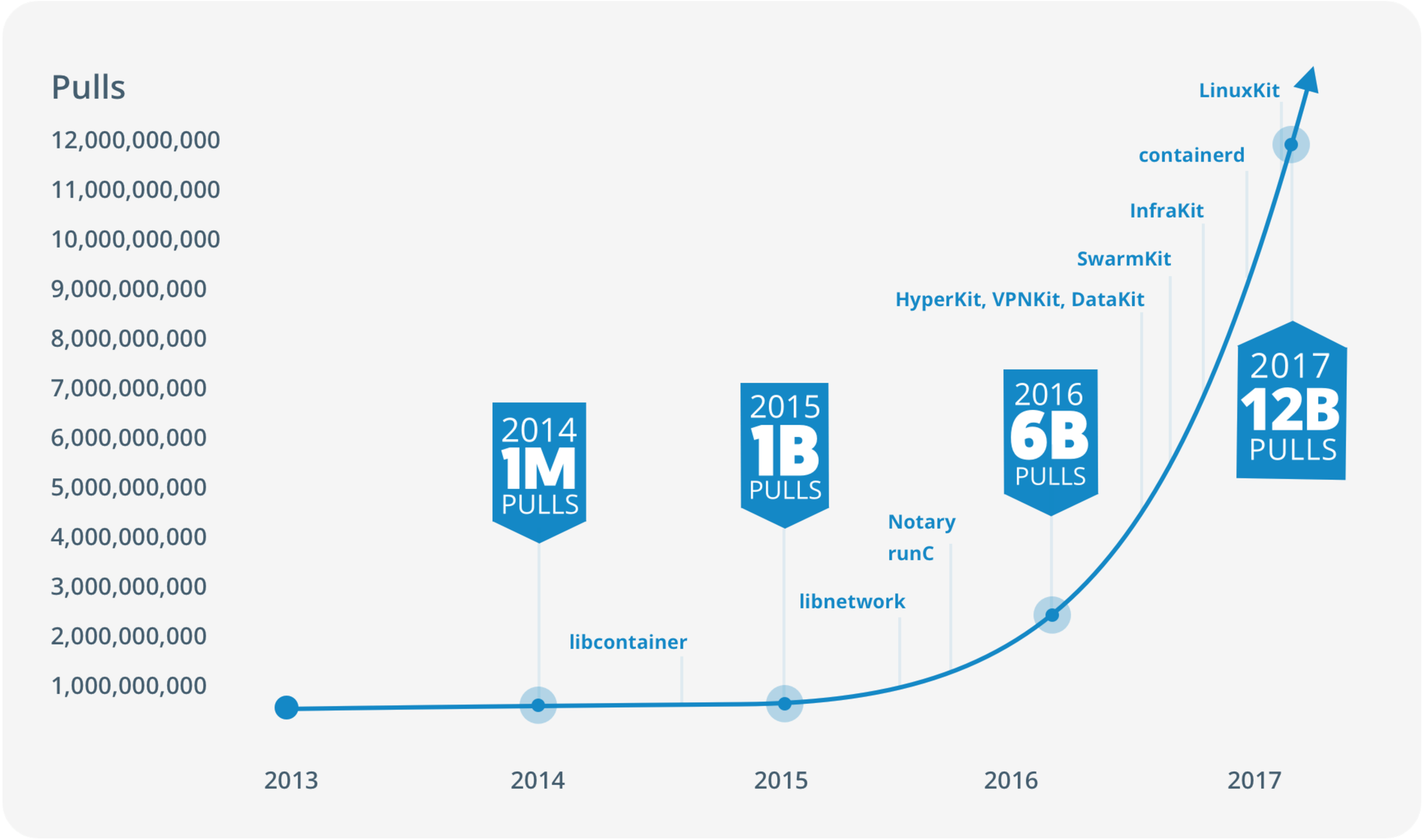

```
$ docker run -p 4000:80 gordon/get-started:part2
Unable to find image 'gordon/get-started:part2' locally
part2: Pulling from gordon/get-started
10a267c67f42: Already exists
f68a39a6a5e4: Already exists
9beaffc0cf19: Already exists
3c1fe835fb6b: Already exists
4c9f1fa8fcb8: Already exists
ee7d8f576a14: Already exists
fbccdcced46e: Already exists
Digest: sha256:0601c866aab2adcc6498200efd0f754037e909e5fd42069adeff72d1e2439068
Status: Downloaded newer image for gordon/get-started:part2
 * Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
```

Strong Ecosystem



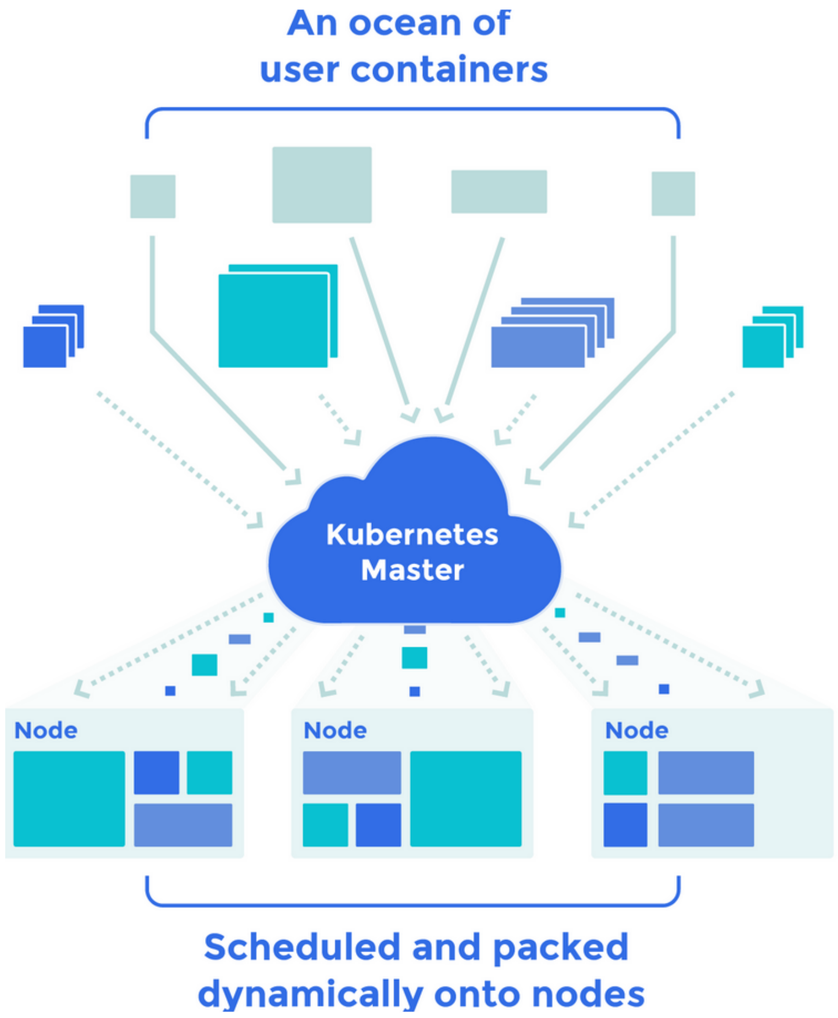
- Images are easy to find, modify, and contribute back to DockerHub
- Private images easy to store in same place

https://docs.docker.com/engine/docker-overview



https://www.docker.com/what-container#/package_software

Container Orchestration



- Distributing containers onto underlying VMs or bare metal
- Kubernetes is the open-source winner, but cloud providers have good offerings, too.

Questions?

Building a model service: the basics

- REST APIs
- Dependency management
- **Performance optimization**
- Horizontal scaling
- Deployment
- Managed options

Making inference on a single machine more efficient

- GPU or no GPU?
- Concurrency
- Model distillation
- Quantization
- Caching
- Batching
- Sharing the GPU
- Libraries

GPU or no GPU?

- **GPU pros**
  - Same hardware you trained on probably
  - In the limit of model size, batch size tuning, etc usually higher throughput
- **GPU cons**
  - More complex to set up
  - Often more expensive

Concurrency

- **What?**
  - Multiple copies of the model running on different CPUs or cores
- **How?**
  - Be careful about thread tuning

```python

import torch

class BertModelInference:
  def __init__(self, model_path, do_quantize=False):

    # Omitting code that loads the Bert model, for clarity

    torch.set_num_threads(1)

  def predict(self, message: str) -> List[float]:
    # Omitting code that calls the Bert model, for clarity
    ...
```

https://blog.roblox.com/2020/05/scaled-bert-serve-1-billion-daily-requests-cpus/

Model distillation

- **What?**
  - Train a smaller model to imitate your larger one
- **How?**
  - Several techniques outlined below
  - Can be finicky to do yourself — infrequently used in practice
  - Exception — pretrained distilled models like DistilBERT

https://heartbeat.fritz.ai/research-guide-model-distillation-techniques-for-deep-learning-4a100801c0eb

Quantization

- **What?**
  - Execute some or all of the operations in your model with a smaller numerical representation than floats (e.g., INT8)
  - Some tradeoffs with accuracy
- **How?**
  - PyTorch and Tensorflow Lite have quantization built-in
  - Can also run **quantization-aware training,** which often results in higher accuracy

https://pytorch.org/blog/introduction-to-quantization-on-pytorch/

Caching

- **What?**
  - For some ML models, some inputs are more common than others
  - Instead of always calling the model, first check the cache
- **How?**
  - Can get very fancy
  - Basic way uses functools

`@functools.cache(user_function)`

Simple lightweight unbounded function cache. Sometimes called "memoize".

Returns the same as `lru_cache(maxsize=None)`, creating a thin wrapper around a dictionary lookup for the function arguments. Because it never needs to evict old values, this is smaller and faster than `lru_cache()` with a size limit.

For example:

```
@cache
def factorial(n):
    return n * factorial(n-1) if n else 1

>>> factorial(10)      # no previously cached result, makes 11 recursive calls
3628800
>>> factorial(5)       # just looks up cached value result
120
>>> factorial(12)      # makes two new recursive calls, the other 10 are cached
479001600
```

New in version 3.9.

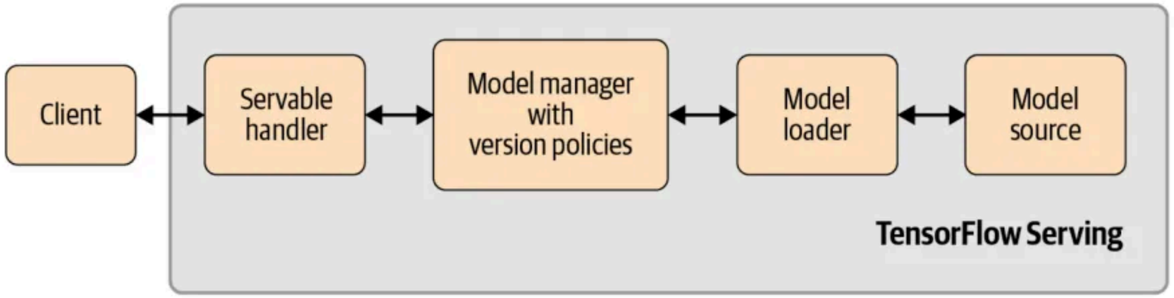
Batching

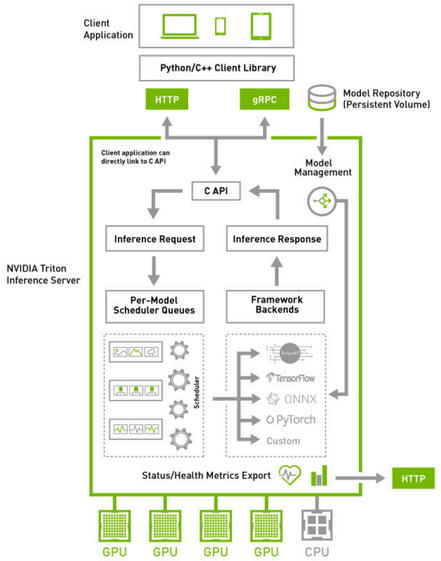
- **What?**
  - ML models often achieve higher throughput when doing prediction in parallel, especially in a GPU
- **How?**
  - Gather predictions until you have a batch, run prediction, return to user
  - Batch size needs to be tuned
  - You need to have a way to shortcut the process if latency becomes too long
  - Probably don’t want to implement this yourself

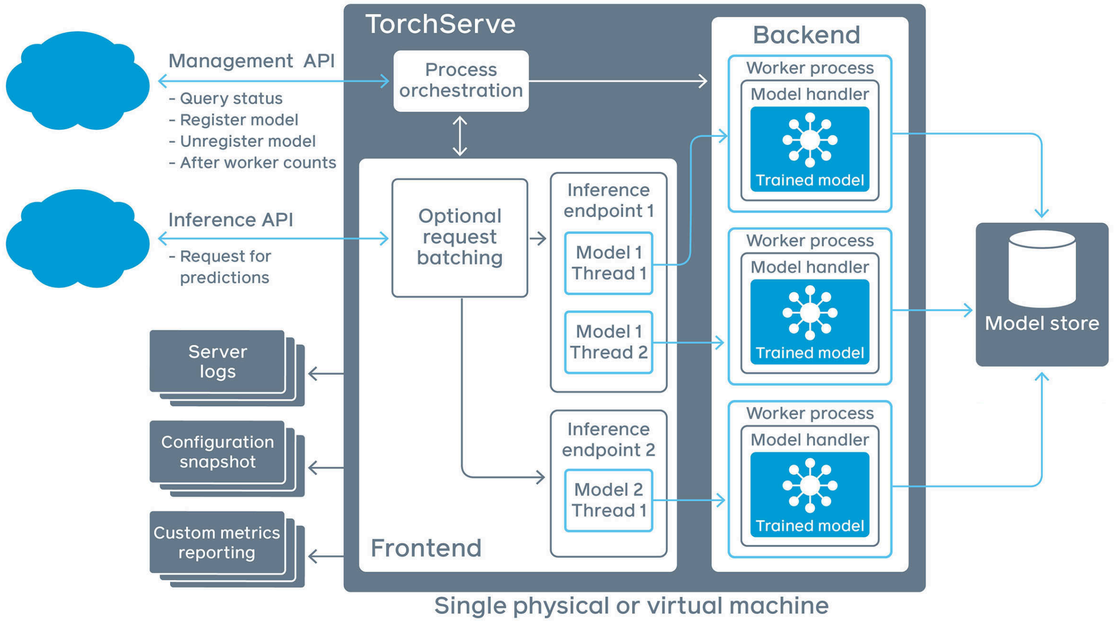
Sharing the GPU

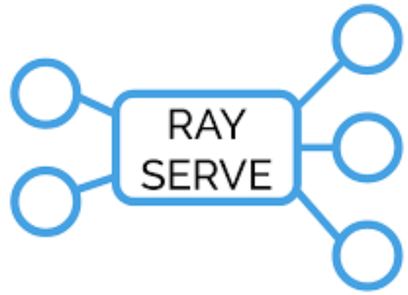
- **What?**
  - Your model may not take up all of the GPU memory with your inference batch size. Why not run multiple models on the same GPU?
- **How?**
  - You'll probably want to use a model serving solution that supports this out of the box

Model serving libraries









Questions?

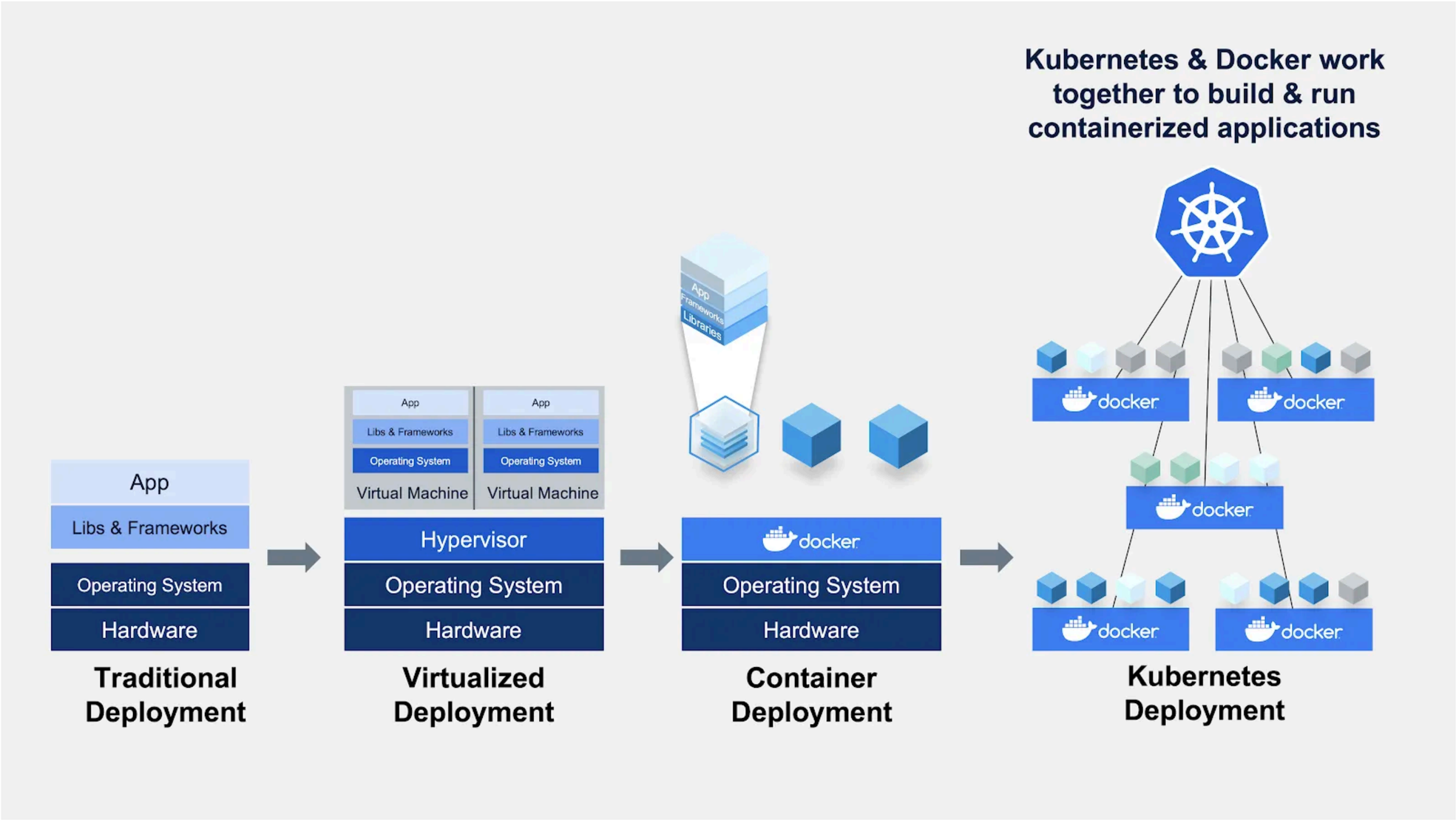
Building a model service: the basics

- REST APIs
- Dependency management
- Performance optimization
- **Horizontal scaling**
- Deployment
- Managed options

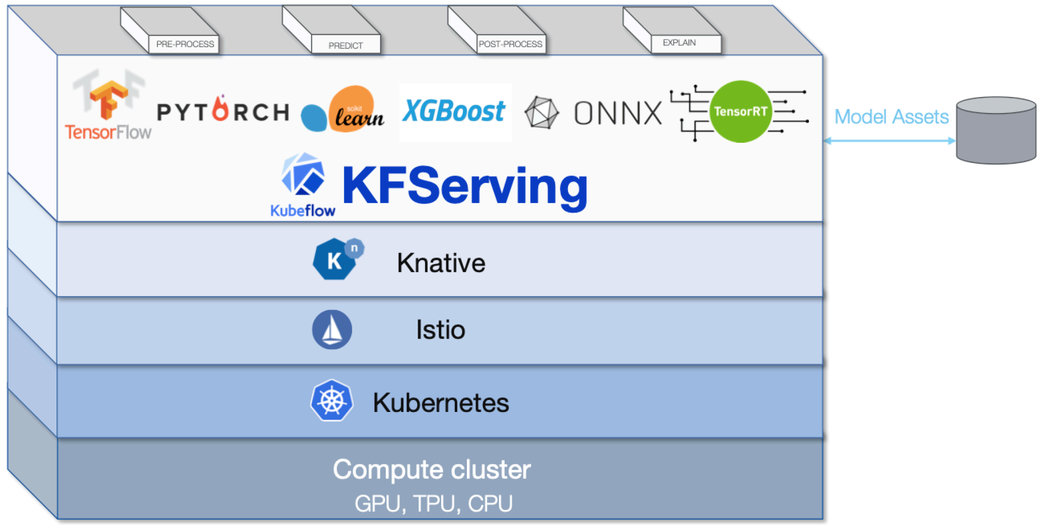
Horizontal scaling

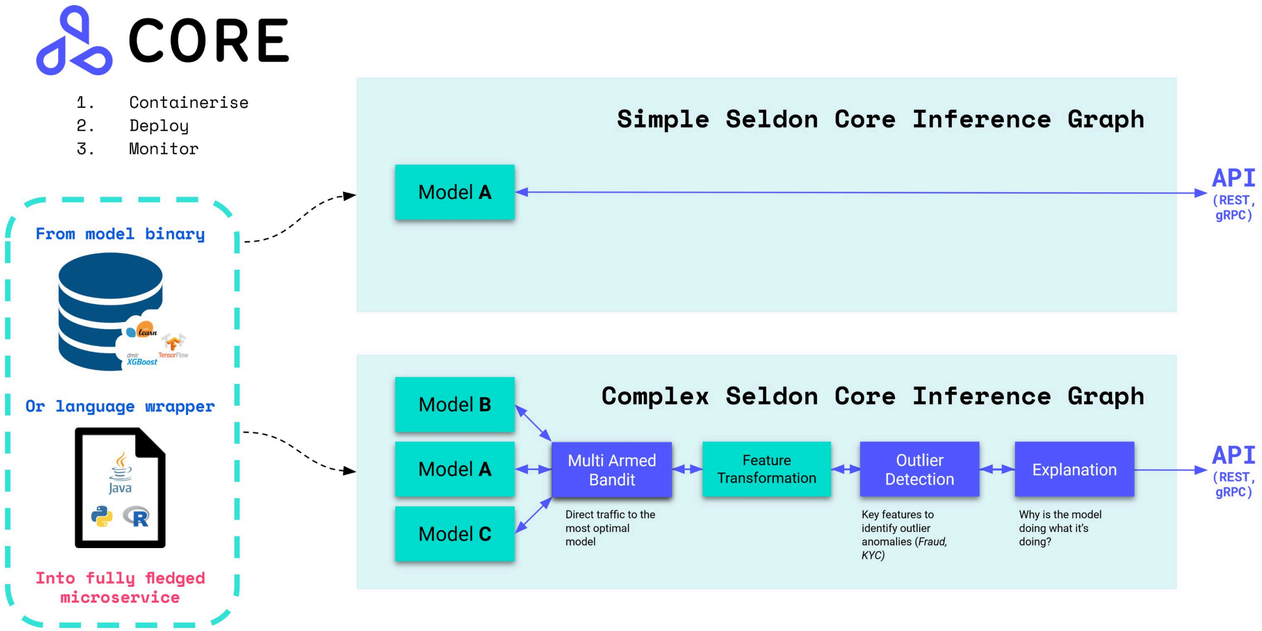
- **What?**
  - If you have too much traffic for a single machine, split traffic among multiple machines
- **How?**
  - Spin up multiple copies of your service and split traffic using a load balancer
  - In practice, two common methods
    - Container orchestration (i.e., Kubernetes)
    - Serverless (e.g., AWS Lambda)

Container orchestration

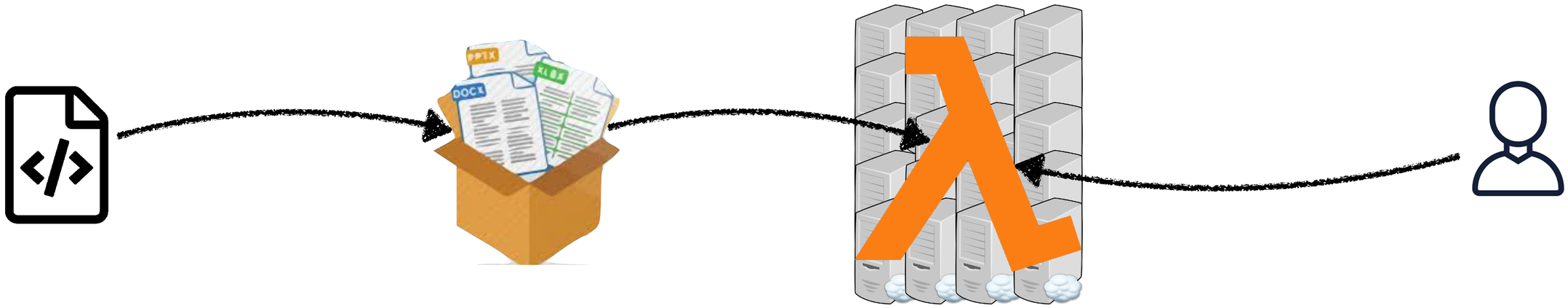


Frameworks for ML deployment on kubernetes

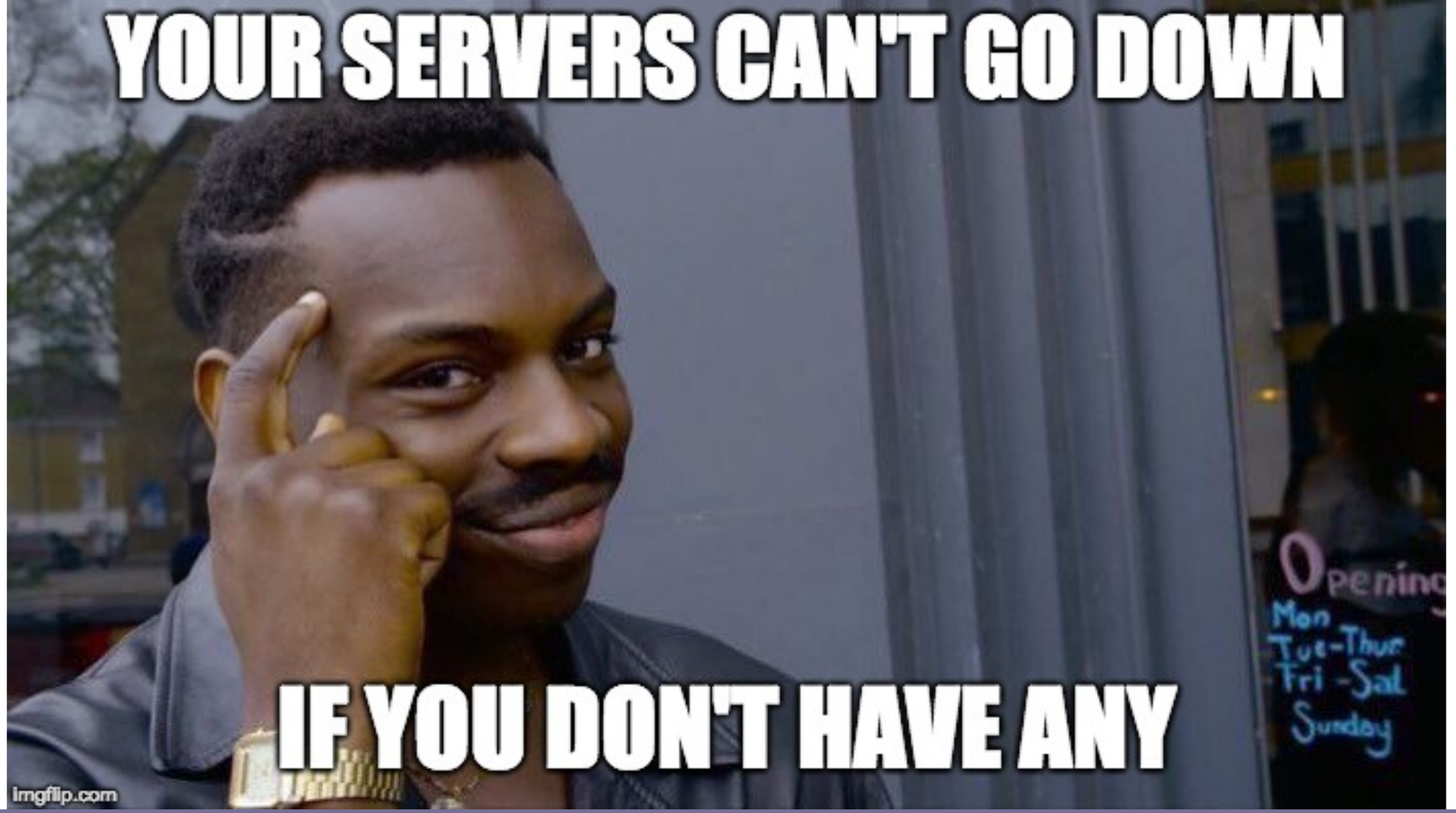




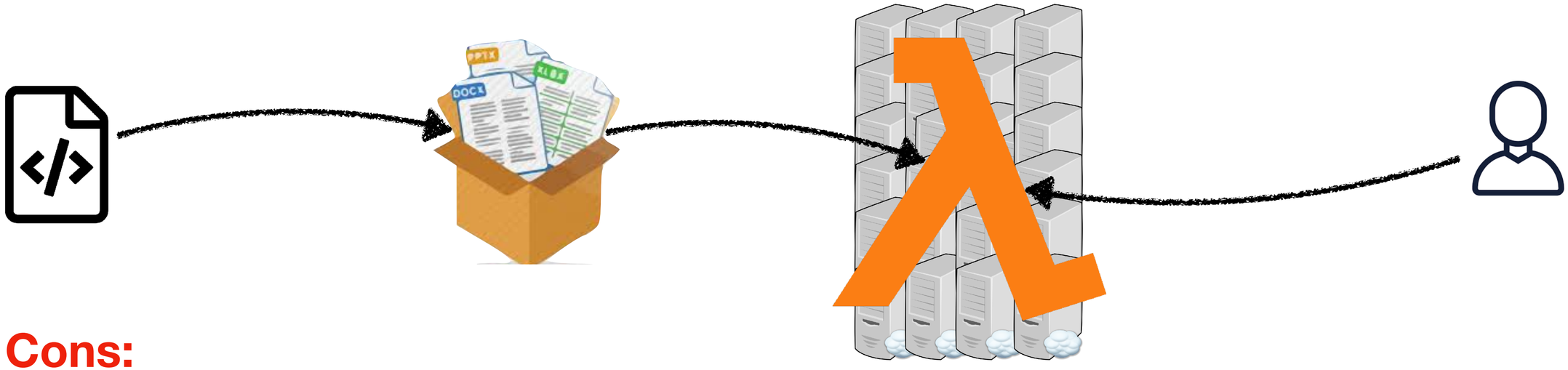
Deploying code as serverless functions



- App code and dependencies are packaged into .zip files or docker containers with a single entry point function
- AWS Lambda (or Google Cloud Functions, or Azure Functions) manages everything else: instant scaling to 10,000+ requests per second, load balancing, etc.
- Only pay for compute-time.


YOUR SERVERS CAN'T GO DOWN
IF YOU DON'T HAVE ANY
imgflip.com
Opening
Mon
Tue-Thur
Fri -Sat
Sunday

Deploying code as serverless functions



- **Cons:**
  - Limited size of deployment package
  - CPU-only, limited execution time
  - Can be challenging to build pipelines of models
  - Little to no state management (e.g., for caching)
  - Limited deployment tooling

Questions?

Building a model service: the basics

- REST APIs
- Dependency management
- Performance optimization
- Horizontal scaling
- **Deployment**
- Managed options

Model deployment

- **What?**
  - If serving is how you turn a model into something that can respond to requests, **deployment** is how you roll out, manage, and update these services
- **How?**
  - You probably want to be able to **roll out gradually**, **roll back instantly**, and **deploy pipelines of models**
  - Your deployment library may take care of this for you

Building a model service: the basics

- REST APIs
- Dependency management
- Performance optimization
- Horizontal scaling
- Deployment
- **Managed options**

Managed options

Google AI

ALGORITHMIA

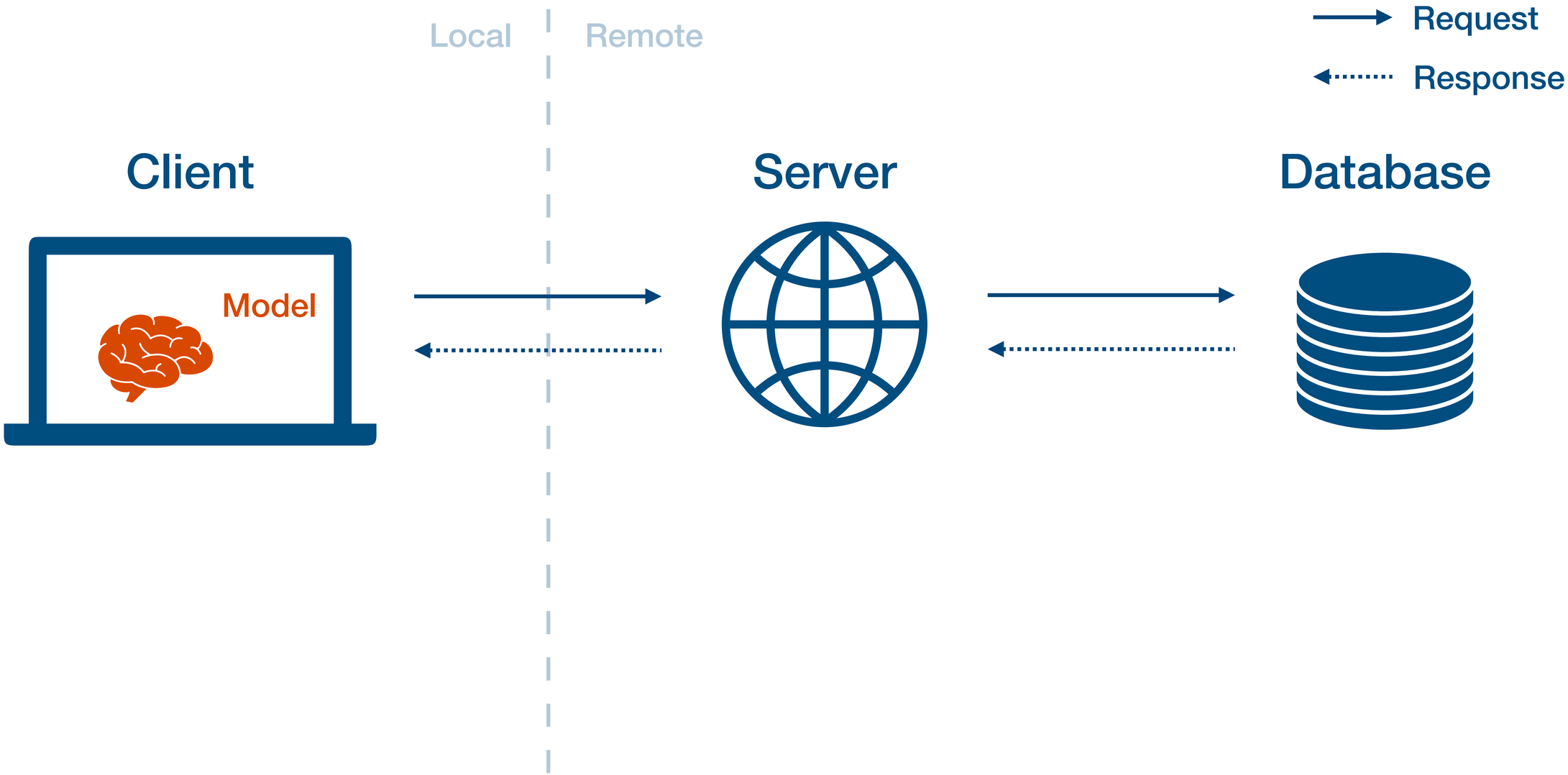
```
$ cortex deploy apis.yaml

• creating text-generator (realtime API)
• creating image-classifier (batch API)
• creating video-analyzer (async API)

all APIs are ready!
```

Building a model service: takeaways

- If you are doing CPU inference, can get away with scaling by launching more servers, or going serverless.
- Serverless makes sense if you can get away with CPUs and traffic is spiky or low-volume.
- If using GPU inference, serving tools like TF serving, Triton, and torch serve will save you time
- Worth keeping an eye on the startups in this space for GPU inference

Questions?

Edge prediction



Edge prediction

- Send model weights to the client device
- Client loads the model and interacts with it directly

Model-as-service

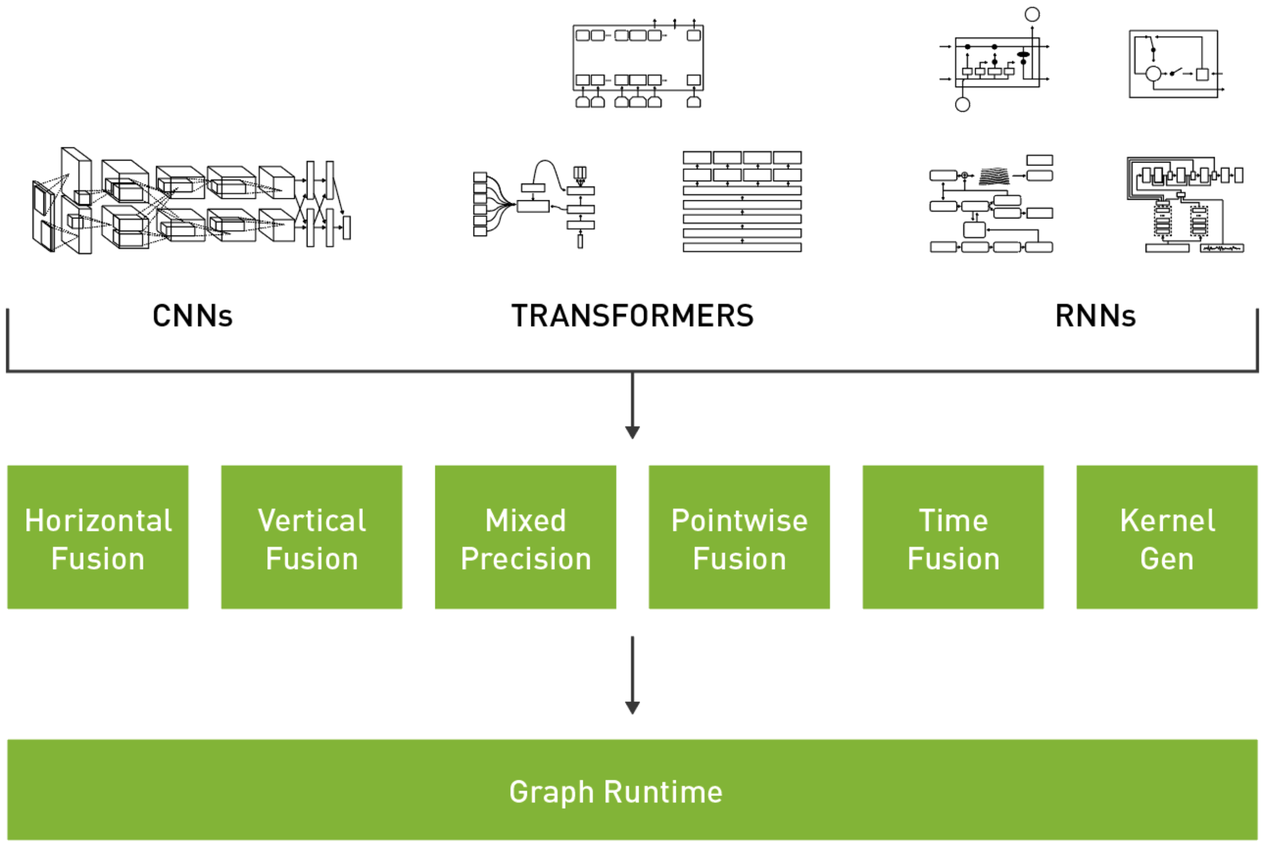
Pros

- Low-latency
- Does not require an internet connection
- Data security — data doesn't need to leave the user's device

Cons

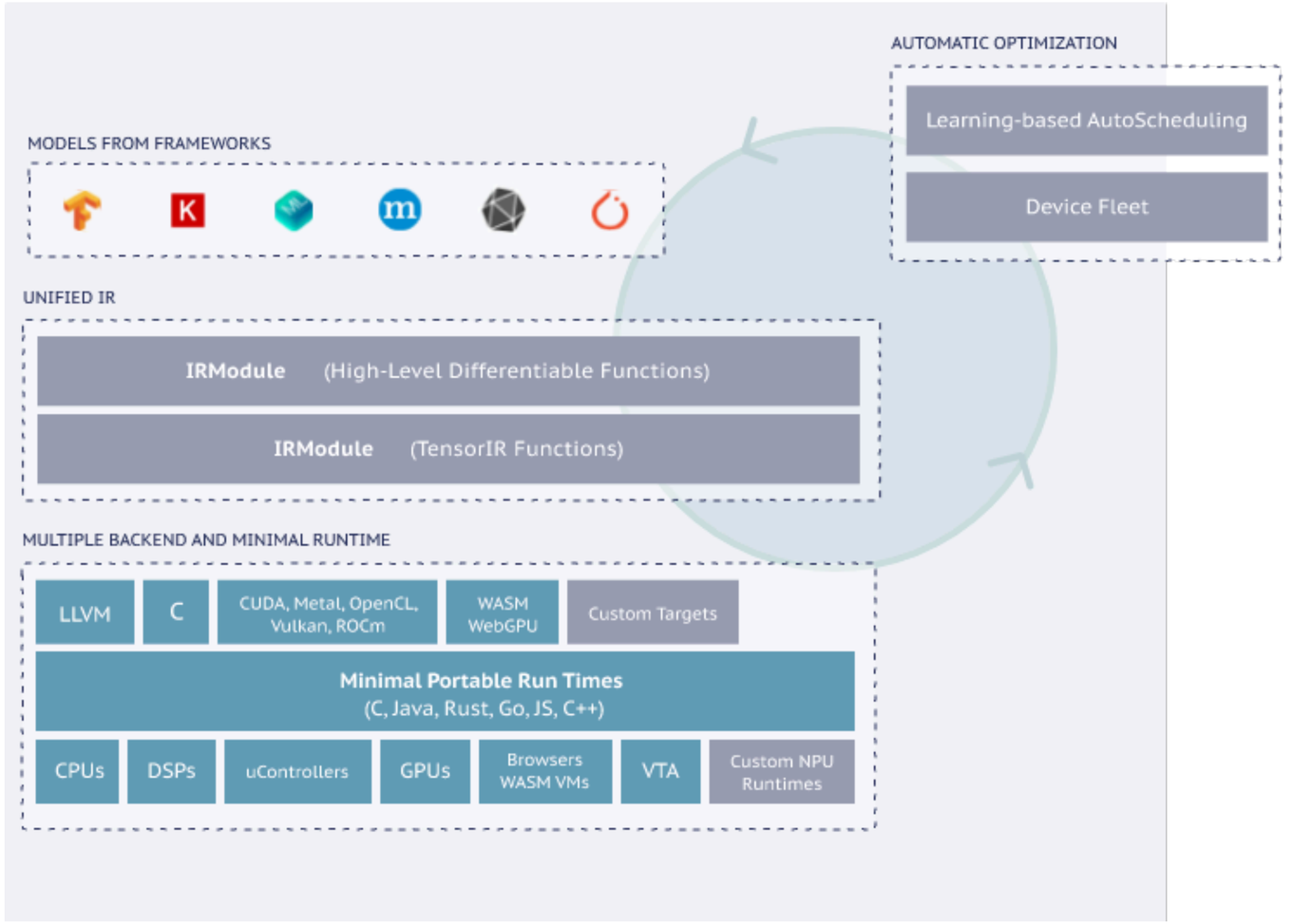
- Often limited hardware resources available
- Embedded and mobile frameworks are less full featured than tensorflow / pytorch
- Difficult to update models
- Difficult to monitor and debug when things go wrong

Tools for edge deployment



TensorRT: Model optimizer and inference runtime for tensor flow + NVIDIA devices

Tools for edge deployment



Apache TVM: Library-agnostic and target-device agnostic inference runtime

Tools for edge deployment

Deploy machine learning models on mobile and IoT devices

TensorFlow Lite is an open source deep learning framework for on-device inference.

See the guide — Guides explain the concepts and components of TensorFlow Lite.

See examples — Explore TensorFlow Lite Android and iOS apps.

See tutorials — Learn how to use TensorFlow Lite for common use cases.

How it works

Pick a model

Pick a new model or retrain an existing one.

Read the developer guide →

Convert

Convert a TensorFlow model into a compressed flat buffer with the TensorFlow Lite Converter.

Read the developer guide →

Deploy

Take the compressed .tflite file and load it into a mobile or embedded device.

Read the developer guide →

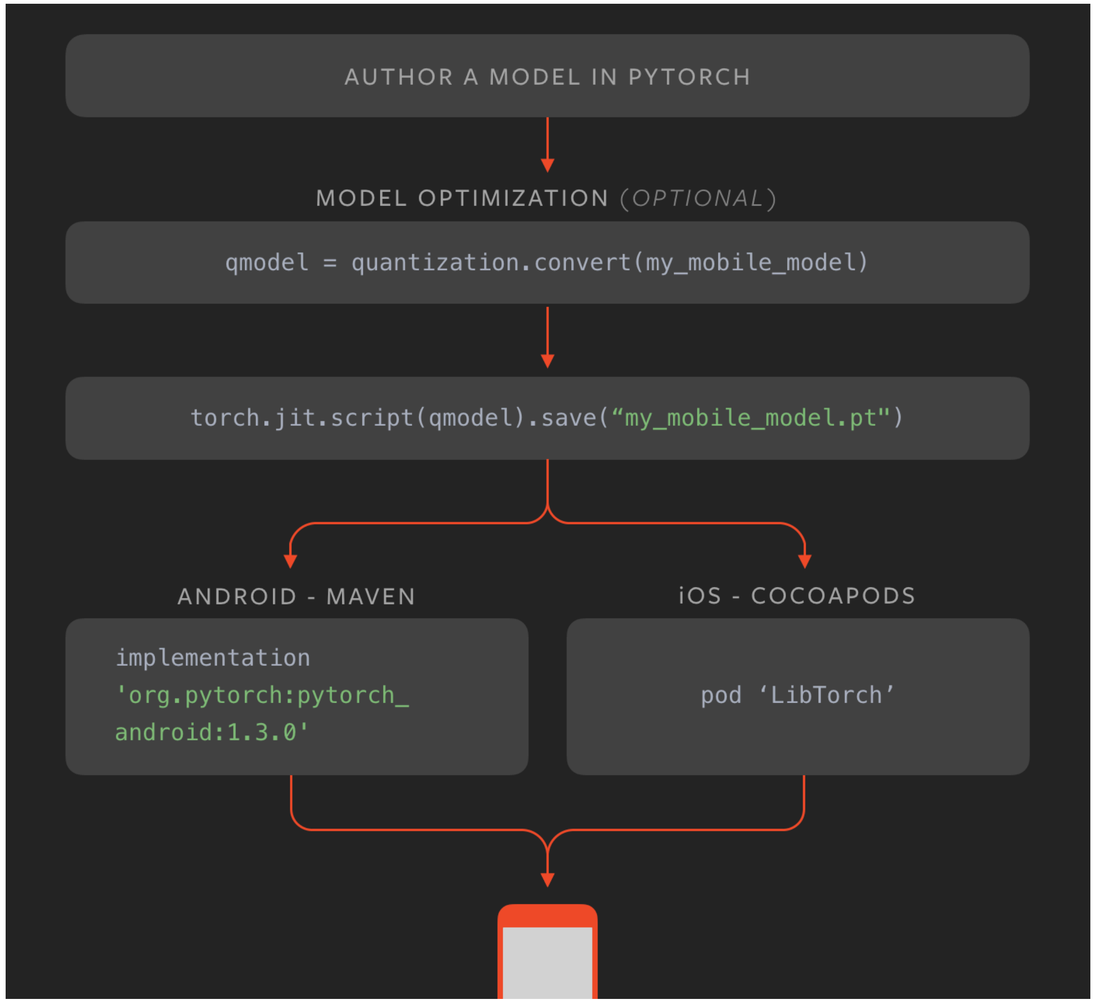
Optimize

Quantize by converting 32-bit floats to more efficient 8-bit integers or run on GPU.

Read the developer guide →

TFLite: tensor flow on mobile / edge devices

Tools for edge deployment



PyTorch mobile: PyTorch on iOS and Android

Tools for edge deployment

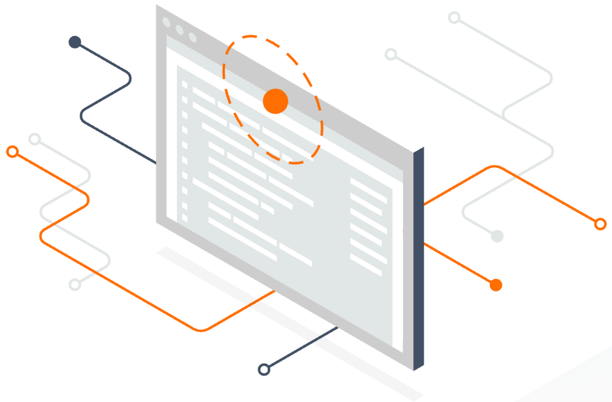
TensorFlow.js is a library for machine learning in JavaScript

Develop ML models in JavaScript, and use ML directly in the browser or in Node.js.

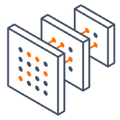
See tutorials — Tutorials show you how to use TensorFlow.js with complete, end-to-end examples.

See models — Pre-trained, out-of-the-box models for common use cases.

See demos — Live demos and examples run in your browser using TensorFlow.js.



How it works



Run existing models

Use off-the-shelf JavaScript models or convert Python TensorFlow models to run in the browser or under Node.js.

Use official TensorFlow.js models →
Convert Python models →

Retrain existing models

Retrain pre-existing ML models using your own data.

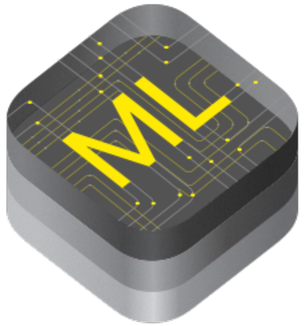
Use Transfer Learning to customize models →

Develop ML with JavaScript

Build and train models directly in JavaScript using flexible and intuitive APIs.

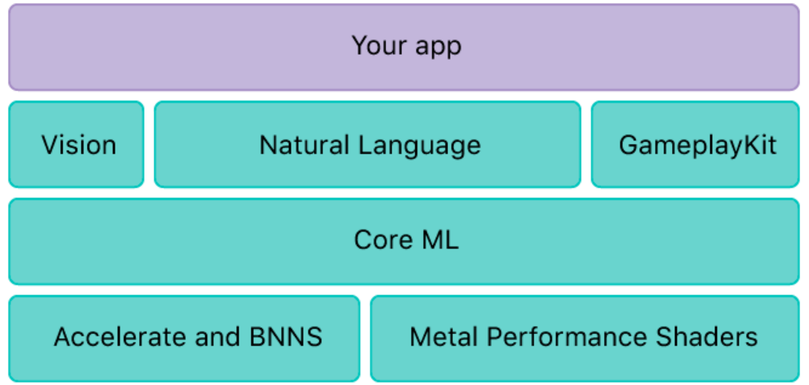
Get started with TensorFlow.js →

TensorFlow.js: tensorflow in the browser



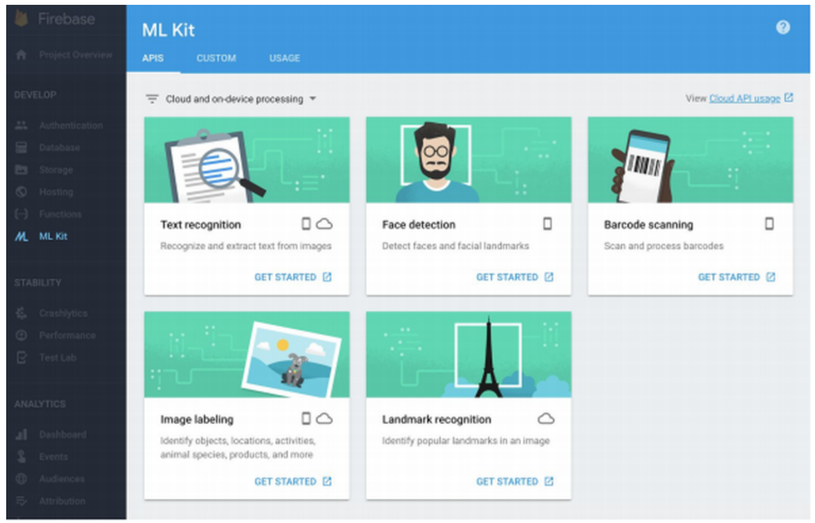
CoreML

- Released at Apple WWDC 2017
- Inference only
- https://coreml.store/



ML Kit

- Announced at Google I/O 2018
- Either via API or on-device
- Offers pre-trained models, or can upload Tensorflow Lite model



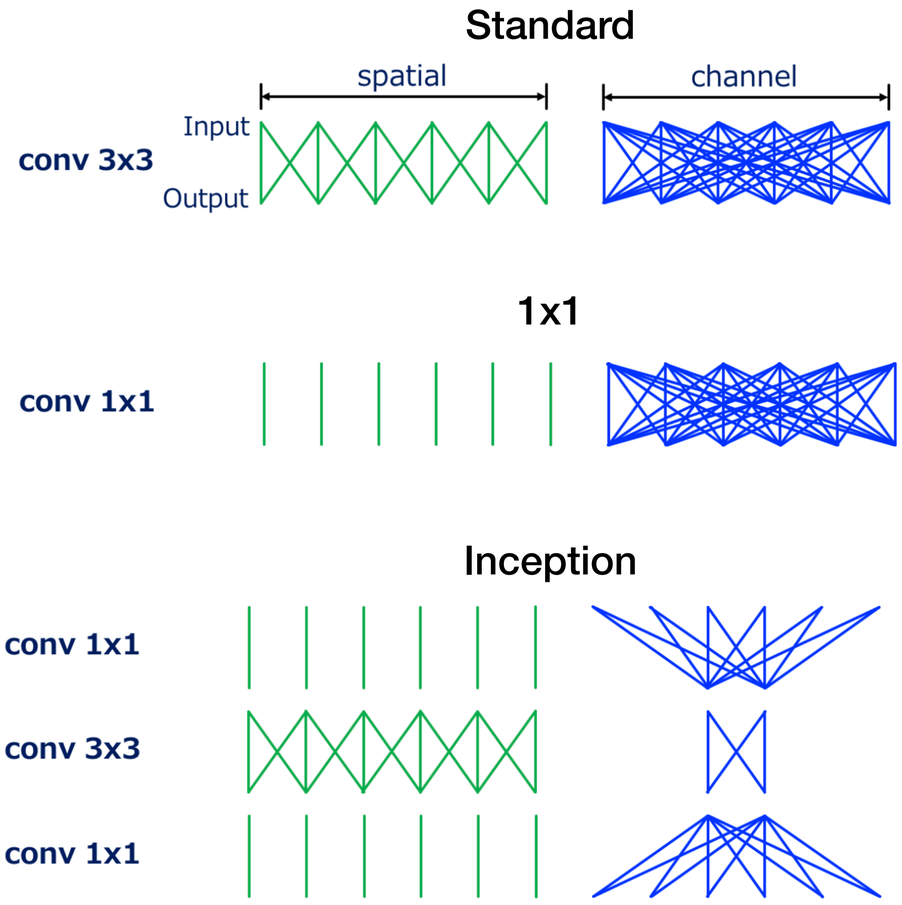
- Supposed to work with both CoreML and MLKit

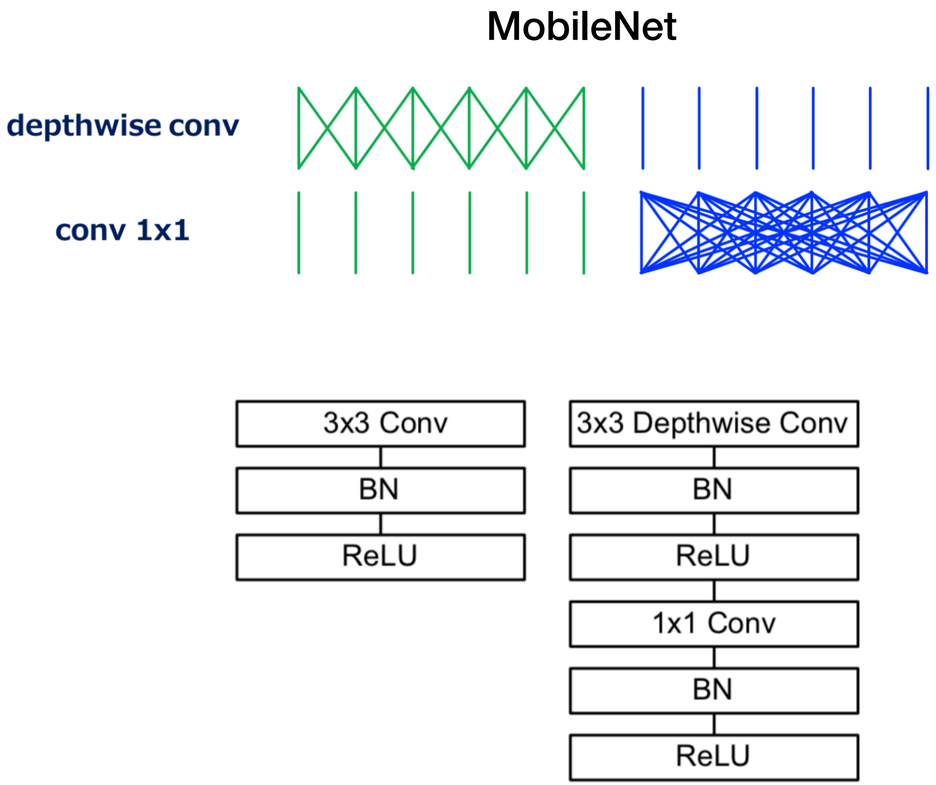
https://heartbeat.fritz.ai/core-ml-vs-ml-kit-which-mobile-machine-learning-framework-is-right-for-you-e25c5d34c765

More efficient models

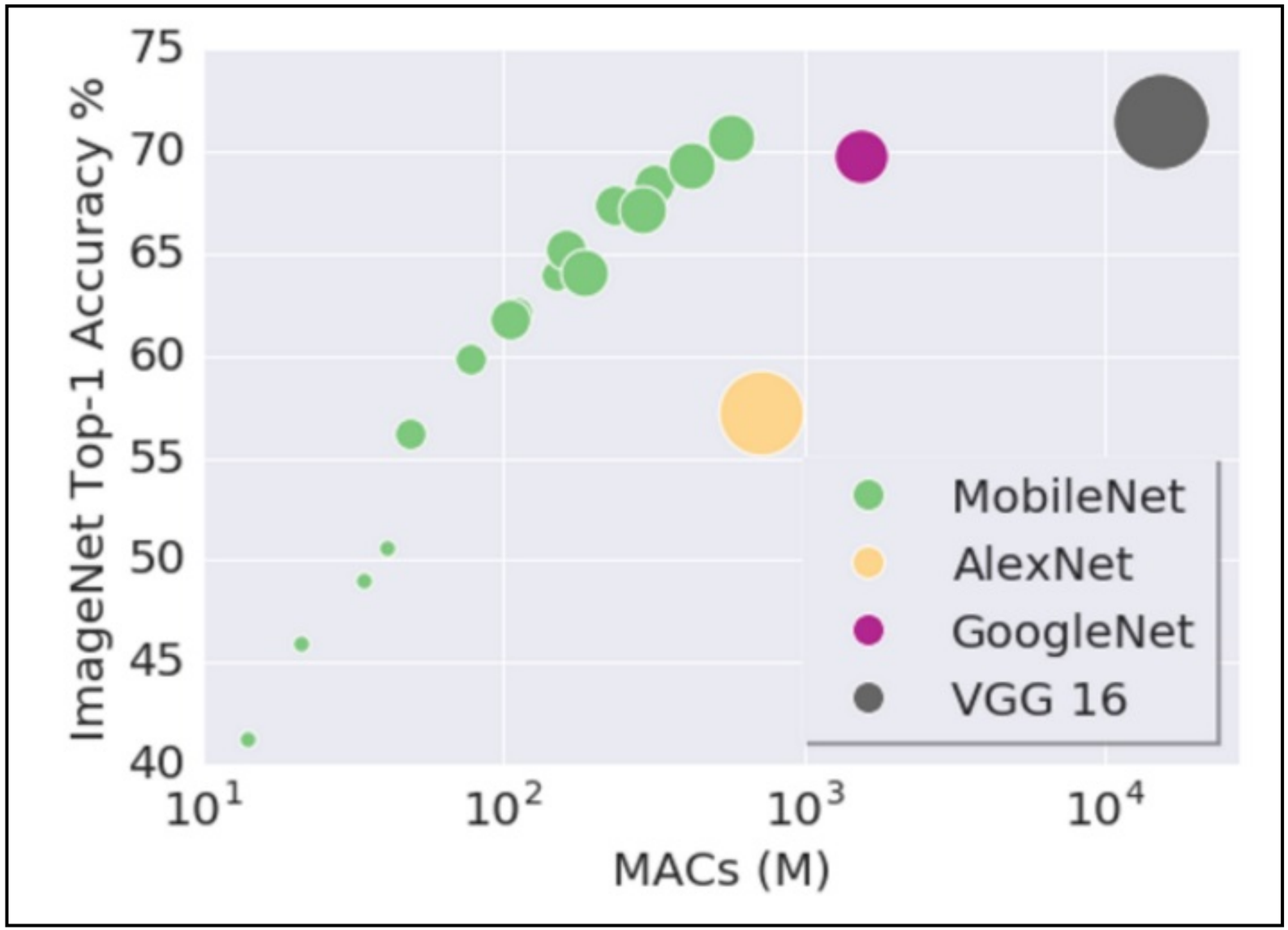
- Quantization and distillation from above
- Mobile-friendly model architectures

MobileNets





https://medium.com/@yu4u/why-mobilenet-and-its-variants-e-g-shufflenet-are-fast-1c7048b9618d



https://medium.com/@yu4u/why-mobilenet-and-its-variants-e-g-shufflenet-are-fast-1c7048b9618d

Recommended case study: DistilBERT

Using the teacher signal, we are able to train a **smaller language model,** we call **DistilBERT,** from the **supervision of BERT** 🤱 (we used the English `bert-base-uncased` version of BERT).

Following Hinton et al., the training loss is a linear combination of the *distillation loss* and the masked *language modeling loss*. Our student is a small version of BERT in which we *removed the token-type embeddings and the pooler* (used for the next sentence classification task) and kept the rest of the architecture identical while reducing the numbers of layers by a factor of two.

> Overall, our distilled model, DistilBERT, has about half the total number of parameters of BERT base and retains 95% of BERT's performances on the language understanding benchmark GLUE.

https://medium.com/huggingface/distilbert-8cf3380435b5

Mindsets for edge deployment

- Choose your architecture with your target hardware in mind
  - You can make up a factor of 2-10 through distillation, quantization, and other tricks, but not more than that
- Once you have a model that works on your edge device, you can iterate locally as long as you add model size and latency to your metrics and avoid regressions
- Treat tuning the model for your device as an additional risk in the deployment cycle and test it accordingly
  - E.g., always test your models on production hardware before deploying
- Since models can be finicky, it's a good idea to build fallback mechanisms into the application in case the model fails or is too slow

Edge deployment: conclusion

- Web deployment is easier, so use it if you need to
- Choose your framework to match the available hardware and corresponding mobile frameworks, or try TVM to be more flexible
- Start considering hardware constraints at the beginning of the project and choose architectures accordingly

Questions?

Today

- Deployment
- **Monitoring**

Train, test, deploy. You're done, right?

- Validation loss is below your target performance
- Test loss is not much worse than validation

Training & troubleshooting lecture

- Your model performs well across all critical slices and metrics
- Qualitatively the predictions make sense
- You verified that the prod model has the same performance characteristics as the dev model
- You verified that the prod model is indeed better than the previous one

Testing lecture

Train, test, deploy. You're done, right?

- Validation loss is below your target performance
- Test loss is not much worse than validation
- Your model pe
  metrics
- Qualitatively th
- You verified th
  performance c
- You verified that the prod model is indeed better than the previous one

Training & troubleshooting lecture

ng lecture

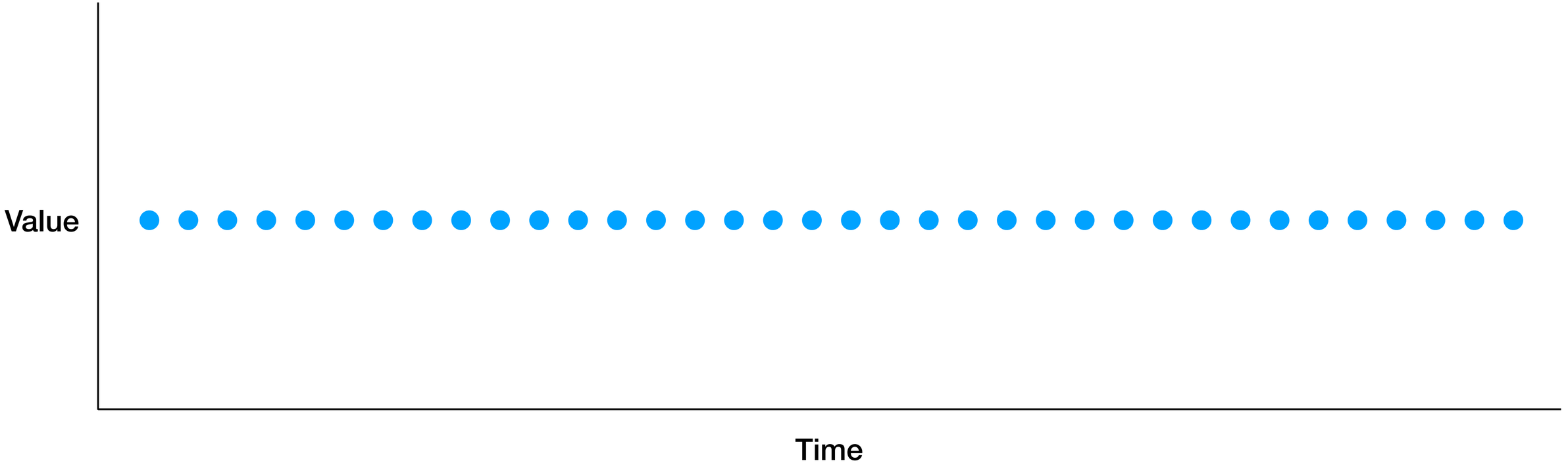
What else could go wrong?

Model performance degrades post-deployment

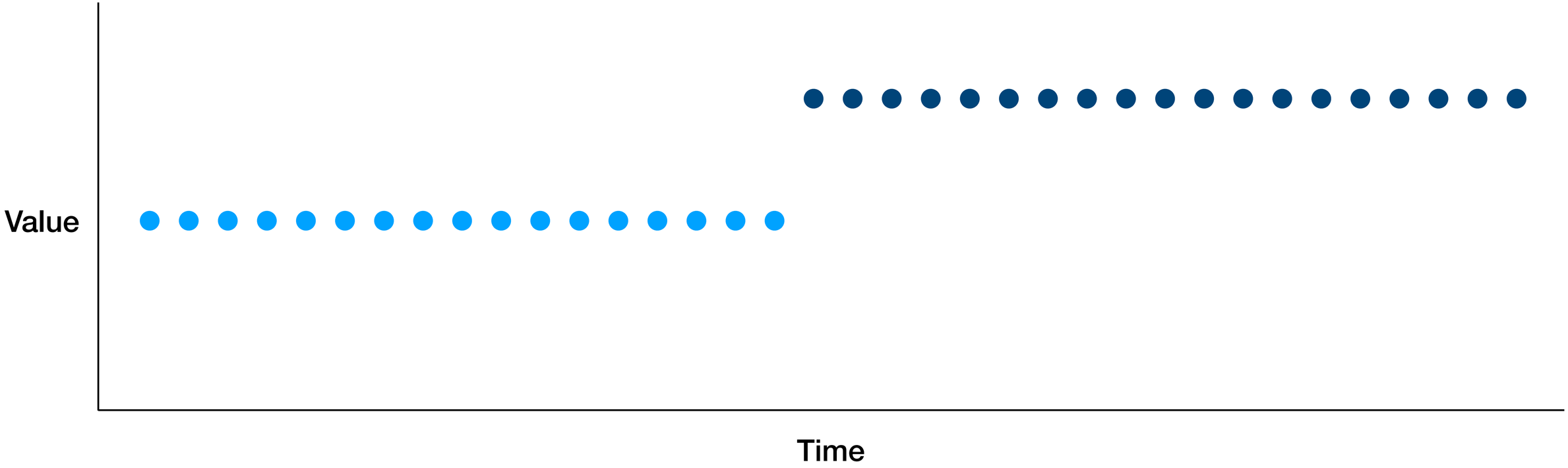
Supervised ML: approximate $p(y \mid x)$ by learning a parameterized model f_θ on data sampled from $p(x, y)$

What can go wrong?	Names	Examples
p(x) changes	• Data drift	• Bug in upstream data pipeline • Malicious users • Launch in a new region • Add new users with different demographics
p(y | x) changes	• Model drift • Concept drift	• Users behavior changes (in response to your model!)
Sample doesn't adequately approximate p(x, y)	• The long tail • Domain shift	• Tasks where outliers matter • Bug in training data pipeline • Bias in the sampling process

Types of data drift



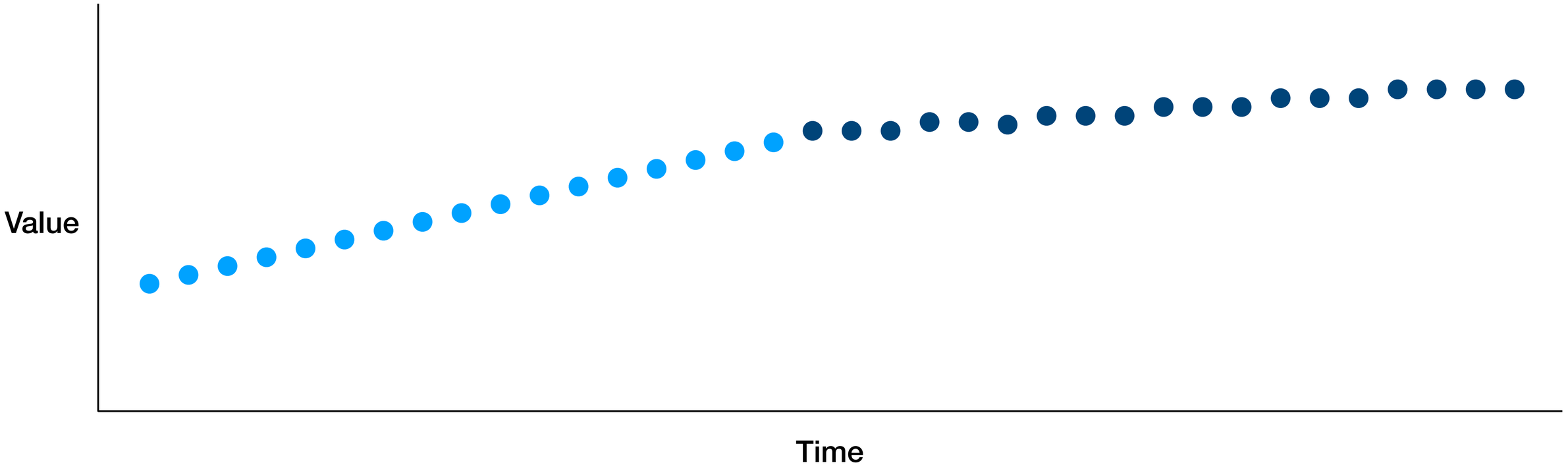
Instantaneous drift (i.e., shift)



Examples

- Model deployed in a new domain
- A bug is introduced in the preprocessing pipeline
- Big external shift, like Covid

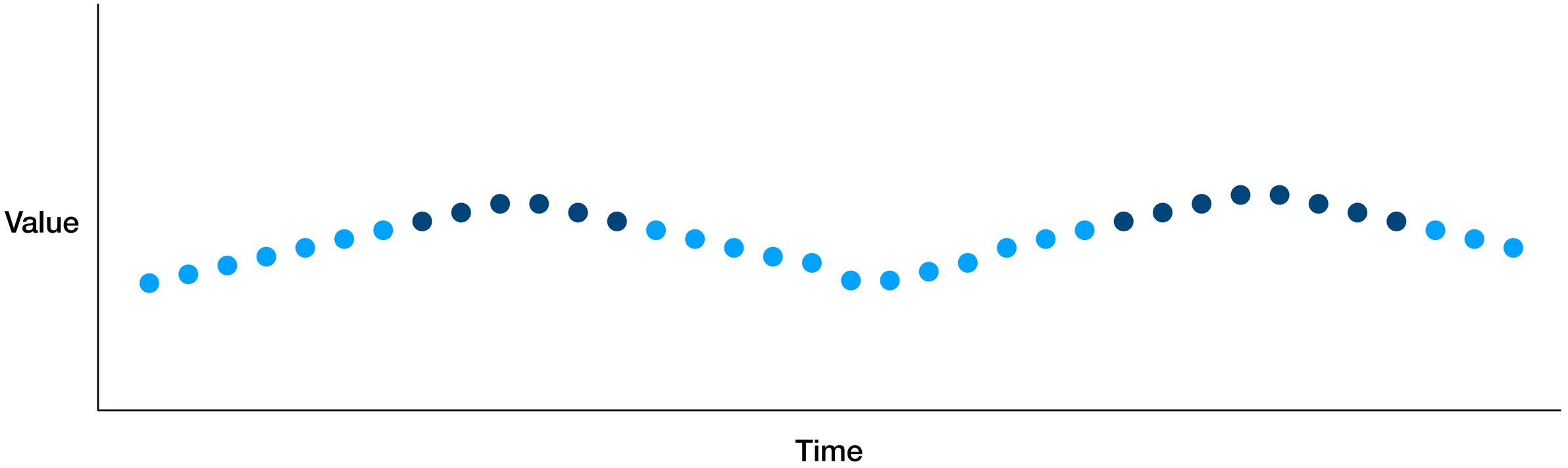
Gradual drift



Examples

- Users preferences change over time
- New concepts get introduced to the corpus over time

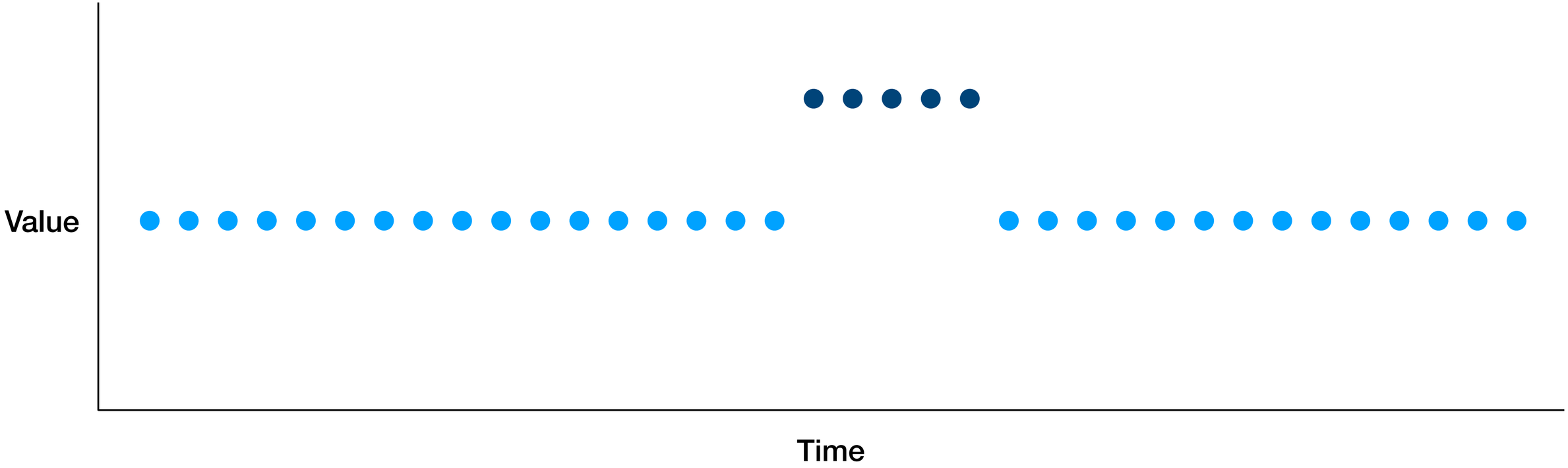
Periodic drift



Examples

- Users preferences are seasonal
- People in different timezones use your model differently

Temporary drift



Examples

- Malicious user attacks your model
- A new user with different demographics tries your product and then churns
- Someone uses your product in a way it was not intended to be used

This can have a huge impact in practice

Artificial intelligence / Machine learning

Our weird behavior during the pandemic is messing with AI models

Machine-learning models trained on normal behavior are showing cracks — forcing humans to step in to set them straight.

by **Will Douglas Heaven** May 11, 2020

How bad the situation is depends on whom you talk to. According to Pactera Edge, a global AI consultancy, "automation is in tailspin." Others say they are keeping a cautious eye on automated systems that are just about holding up, stepping in with a manual correction when needed.

Story from global e-commerce company:

- Bug in retraining pipeline caused recommendations not to be updated for new users
- New users got the same stale recommendations for weeks before the bug was caught
- Estimated $Ms in revenue lost

Questions?

Strategies for monitoring ML models

Outline

- What should I monitor?
- How do I measure if it's changed?
- How do I tell if that change is bad?
- Tools for monitoring
- Monitoring and your broader ML system

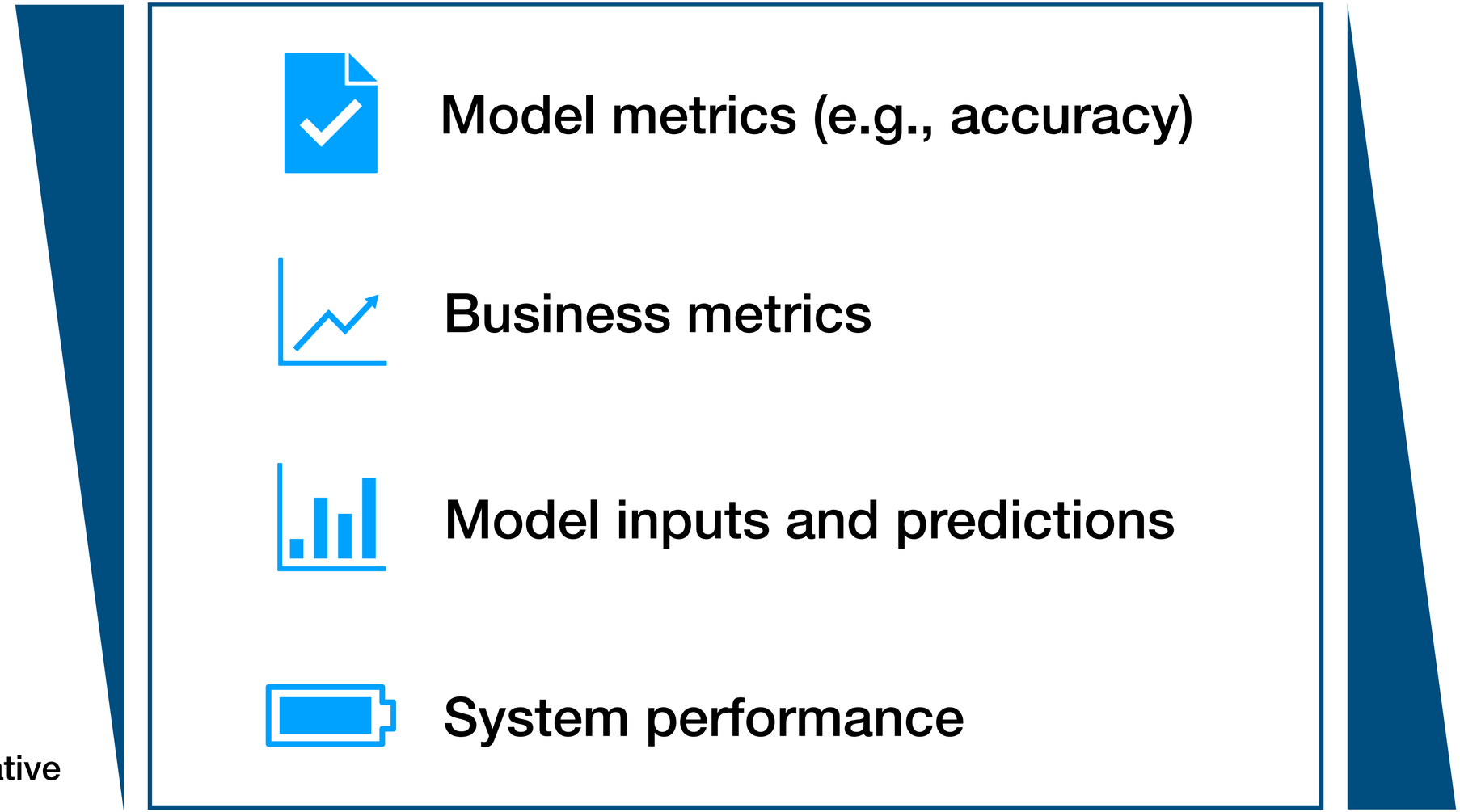
Outline

- **What should I monitor?**
- How do I measure if it's changed?
- How do I tell if that change is bad?
- Tools for monitoring
- Monitoring and your broader ML system

Four types of signals to monitor

More informative

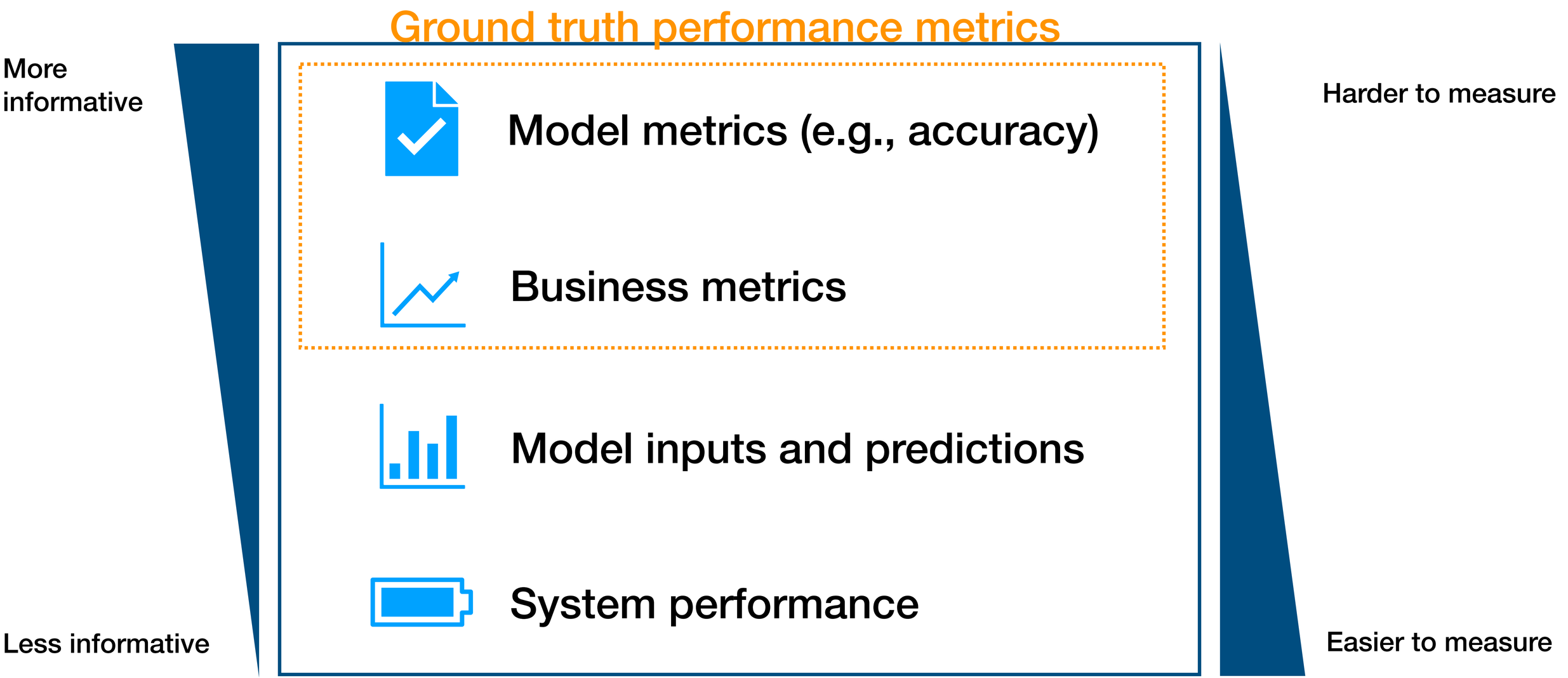
Less informative



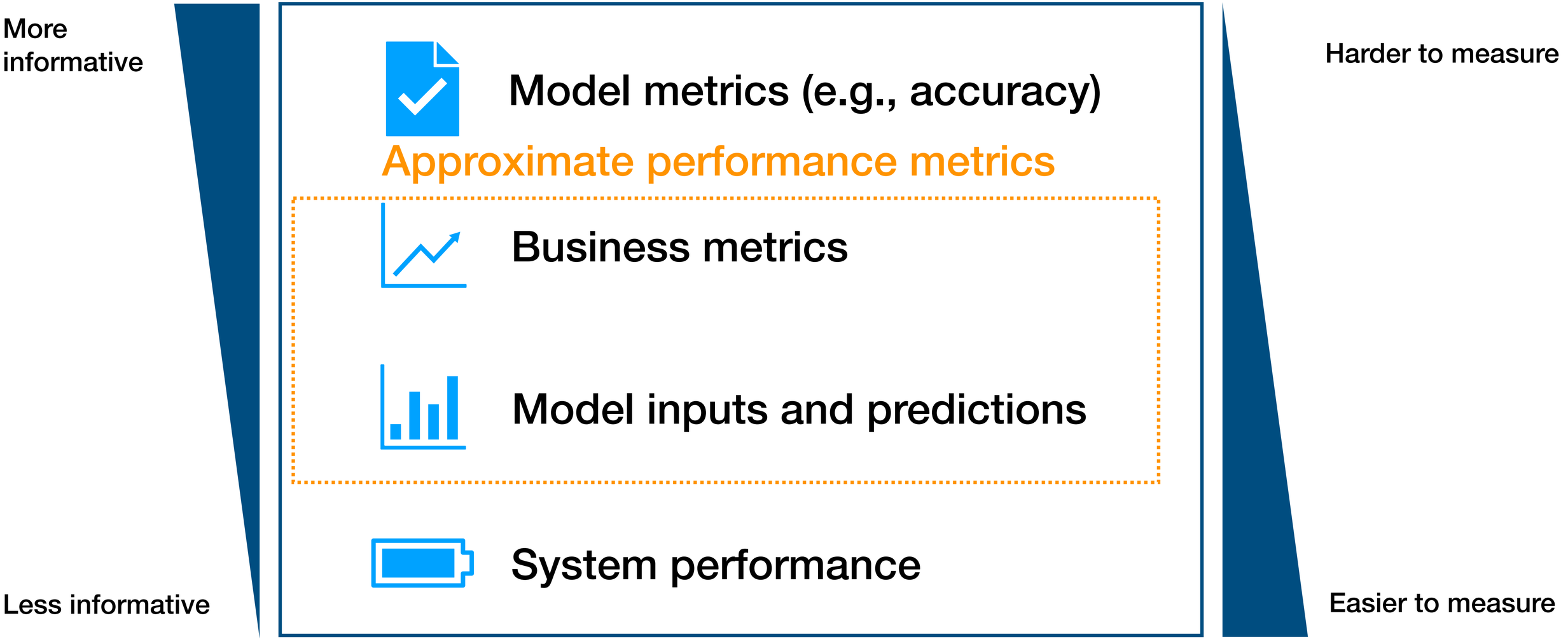
Harder to measure

Easier to measure

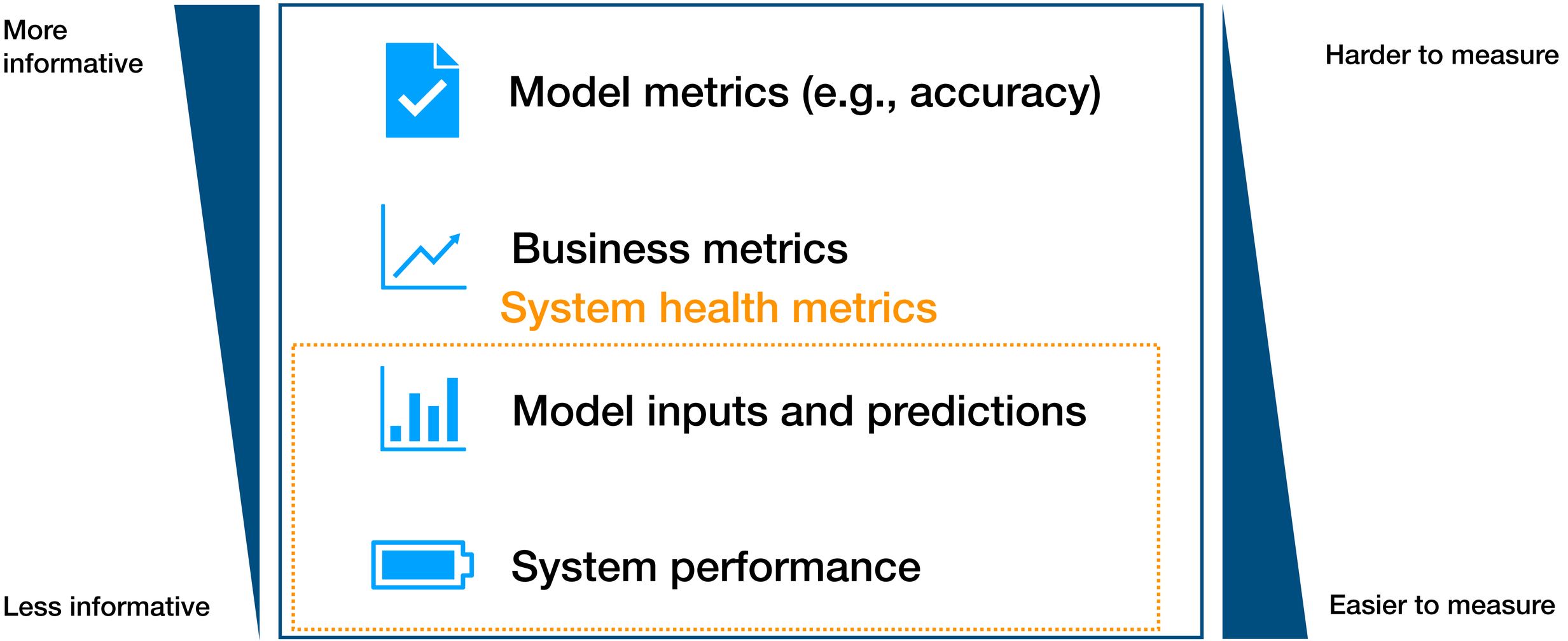
Four types of signals to monitor



Four types of signals to monitor



Four types of signals to monitor

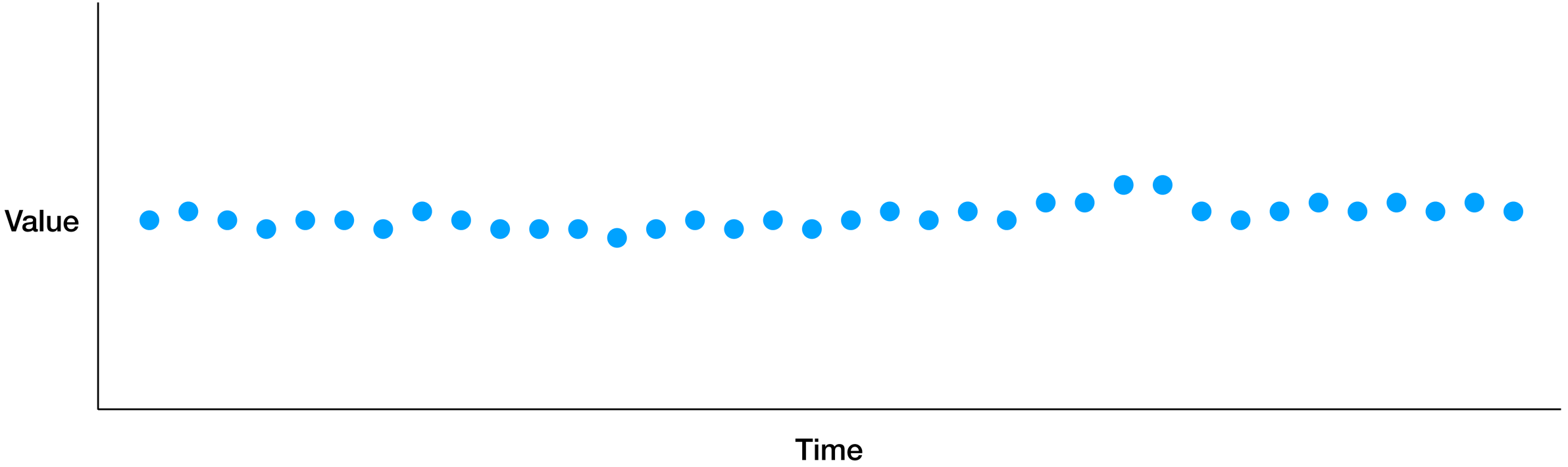


Questions?

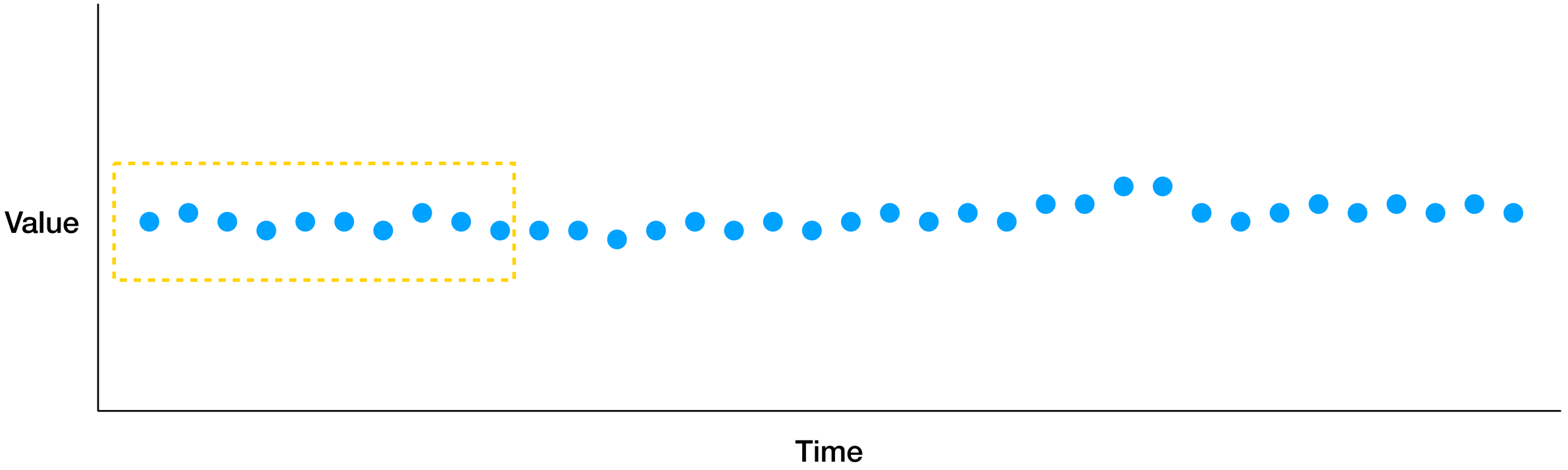
Outline

- What should I monitor?
- **How do I measure if it's changed?**
- How do I tell if that change is bad?
- Tools for monitoring
- Monitoring and your broader ML system

Measuring distribution change



Measuring distribution change



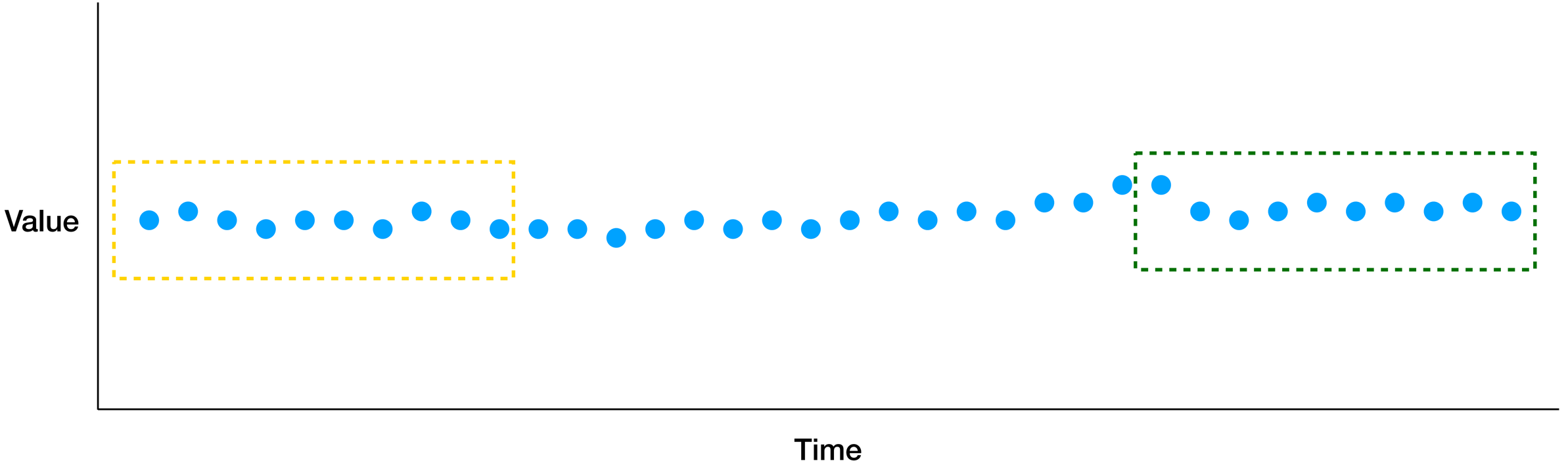
1. Select a window of “good” data to serve as a *reference*

Selecting a reference window

- You can use a fixed window of production data you believe to be healthy
- Some papers [1] advocate for using a sliding window of production data
- In practice, most of the time you probably should use your training or eval data as the reference

[1] Automatic Model Monitoring for Data Streams, Pinto et al. (https://arxiv.org/abs/1908.04240)

Measuring distribution change



2. Select a new window of data to measure distance on

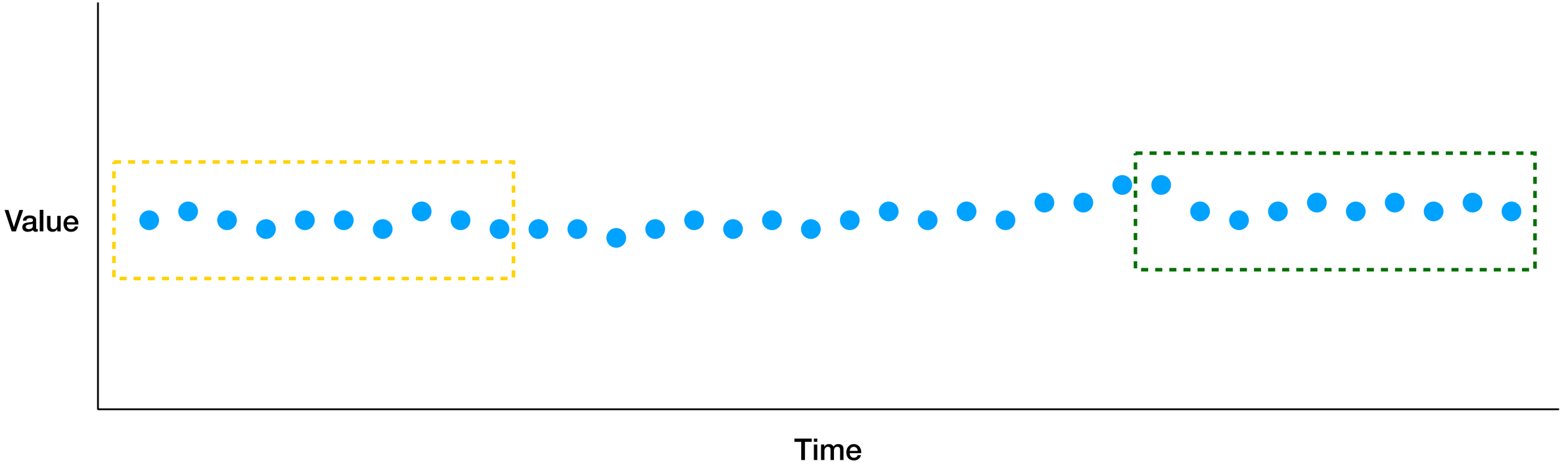
Selecting the measurement window

- Problem-dependent — not aware of a principled approach
- Pragmatic solution: pick one or several window sizes with a reasonable amount of data and slide them [1]
- Special case: window size = 1, i.e., outlier detection [2]

[1] To avoid recomputing metrics over and over again when you slide the window, it's worth looking into the literature on mergeable (quantile) sketching algorithms (https://www.sketchingbigdata.org/)

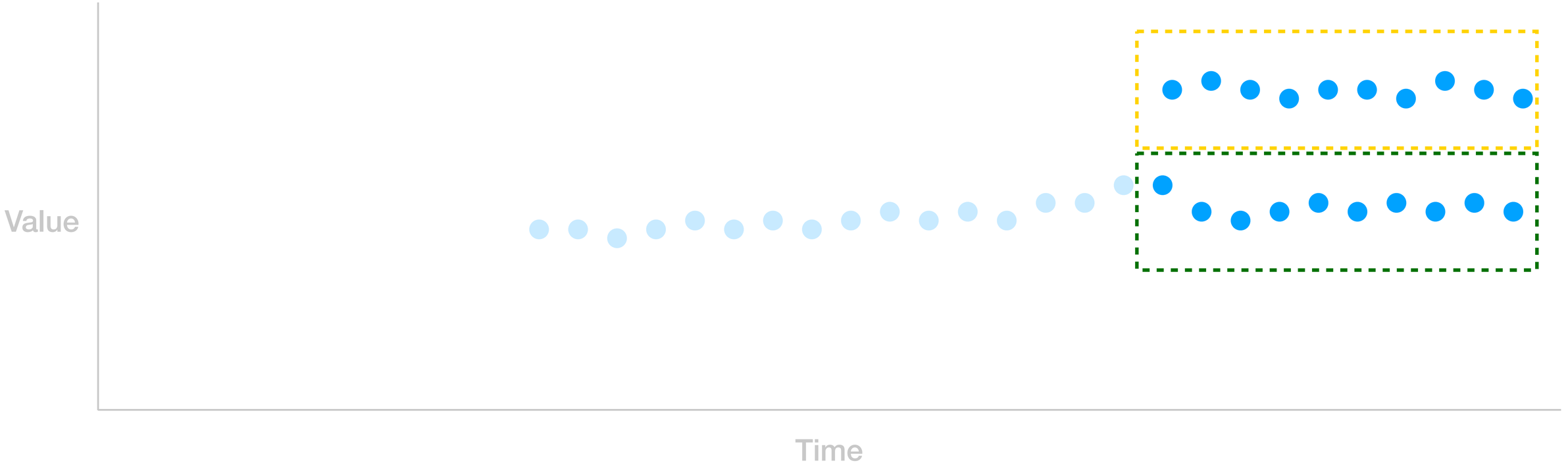
[2] E.g., check out https://pyod.readthedocs.io/en/latest/

Measuring distribution change



3. Compare the windows using a distance metric

Measuring distribution change



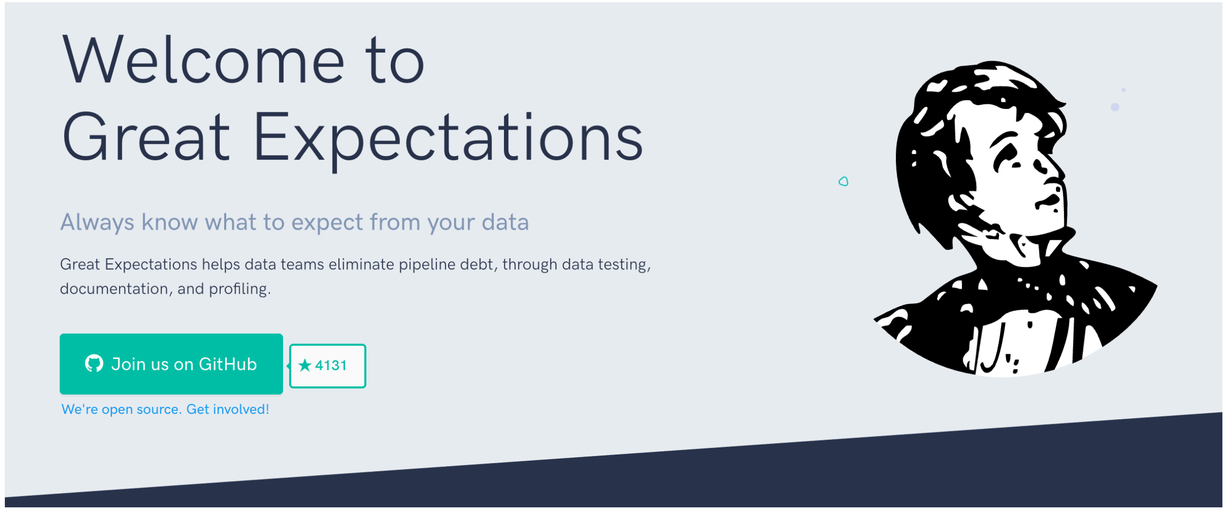
3. Compare the windows using a distance metric

Distance metrics (1D data)

- Rule-based distance metrics (aka, data quality)
- Statistical distance metrics
  - KL divergence
  - Kolmogorov-Smirnov statistic
  - D_1 distance
  - Others

Rule-based distance metrics (i.e., data quality)

- Min, max, mean within acceptable range
- Enough data points
- Not too many missing values
- Col A > Col B
- Etc



Do this!

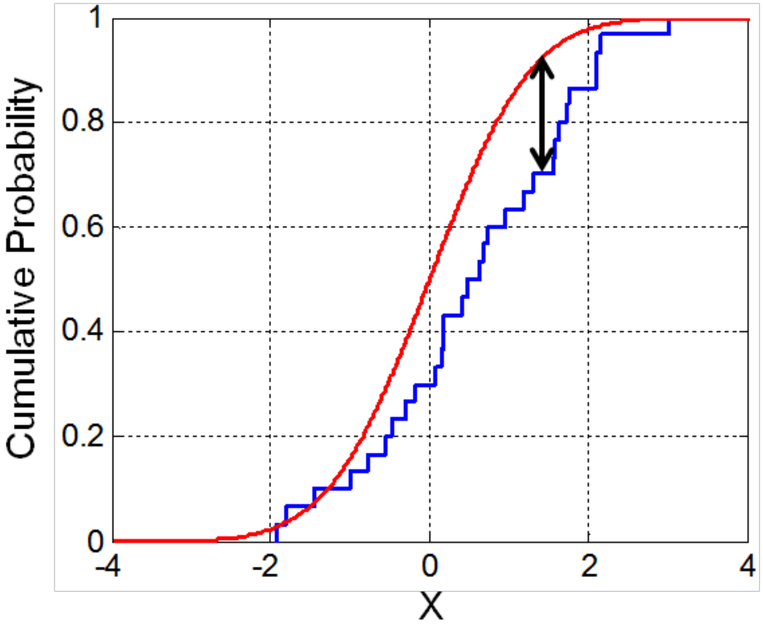
Kullback-Leibler divergence

- $D_{KL}(P \mid Q) = \mathbb{E}_{x \sim P} \log \left(P(x)/Q(x) \right)$
- Sensitive to what happens on the tails of the distribution
- Not interpretable
- What if the P and Q don't have exactly the same range?

Friends don't let friends monitor the KL

Kolmogorov-Smirnov Statistic

- $K = \sup_{t \in (0,1)} \left| B(t) \right|$
- Max distance between CDFs
- Easy to interpret
- Used widely in practice



K-S = “Oh Yes”

D1 distance

- $$d_1(p, q) = \sum_{i=1}^{n} \left| p_i - q_i \right|$$

- Sum of distances between CDFs

- Used at Google [1]

- Isn't this a bit hacky?

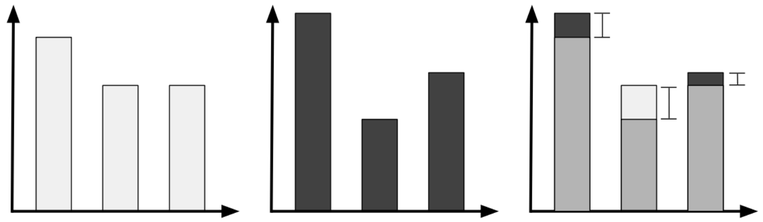


Figure 8: Two distributions, white and black are compared. When overlaying them, the difference can be "seen". The sum of the magnitude of these visible differences is the d_1 distance.

If google does it….

[1] Data Validation for Machine Learning (Breck et al) https://mlsys.org/Conferences/2019/doc/2019/167.pdf

Other distance metrics?

- Earth mover distance
- Population stability index
- Etc

Worthy of some research.

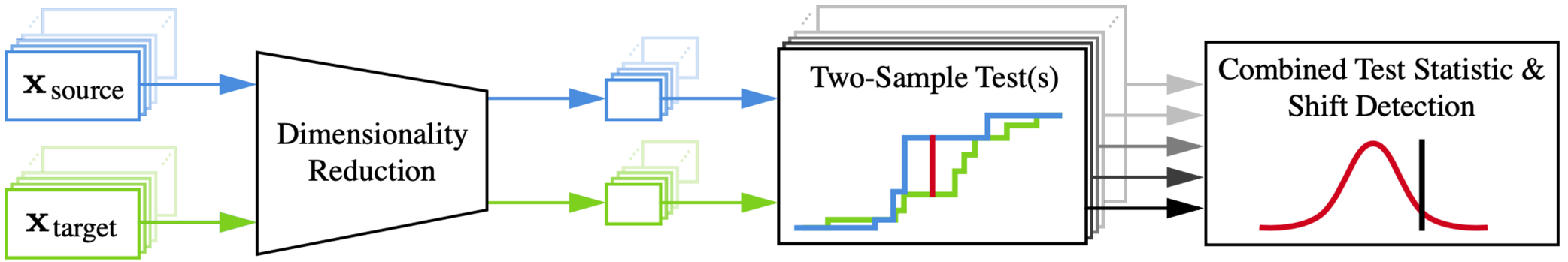
Request for research: how do drifts according to different distance metrics affect model performance differently?

Distance metrics (>1D data)

- Open problem
- Maximum Mean Discrepancy MMD$(\mathcal{F}, p, q) = ||\mu_p - \mu_q||^2_{\mathcal{F}}$
- Do a bunch of 1D comparisons
  - Doesn't capture cross-correlation
  - Multiple hypothesis testing [1]
- Prioritize some features for 1D comparisons
  - Feature importance?
- Projections

[1] Look into corrections like the Bonferroni correction (https://en.wikipedia.org/wiki/Bonferroni_correction)

Detecting high-dimensional data drift using projections



- Analytical projections, e.g.,
  - Mean pixel value
  - Length of sentence
  - (feature_1 - feature_2) / feature_3
- Random projections (e.g., linear)
- Statistical projections, e.g.,
  - Autoencoder or other density model
  - PCA or T-SNE

Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift

Questions?

Outline

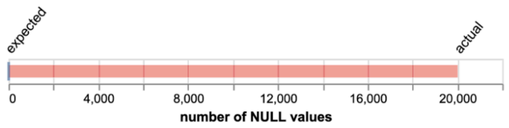
- What should I monitor?
- How do I measure if it's changed?
- **How do I tell if that change is bad?**
- Tools for monitoring
- Monitoring and your broader ML system

Setting thresholds on test values

- Statistical tests (e.g., KS-**Test**)
  - Typically, return a p-value for likelihood that the data distributions are **not the same**
  - When you have a lot of data, you will get very small p-values for small shifts

1. Fixed rule

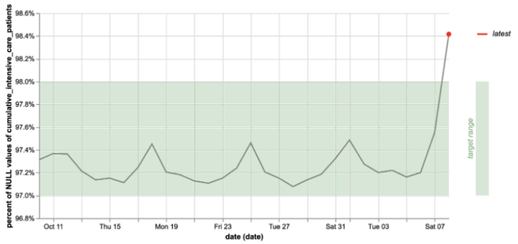
There are never any NULL values



Alert whenever a condition is not perfectly satisfied

2. Specified range

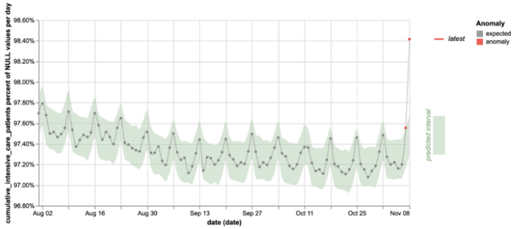
The NULL value % is within 97% and 98%



Alert whenever a value falls outside of a predetermined range specified by the user

3. Predicted range

The NULL value % is not anomalous



Alert whenever a value falls outside of the predicted range from an automated model

4. Unsupervised detection

This pattern is new and concerning

Alert whenever important data changes in an unusual way using unsupervised machine learning algorithms

Used most in practice

https://blog.anomalo.com/dynamic-data-testing-f831435dba90

Request for research

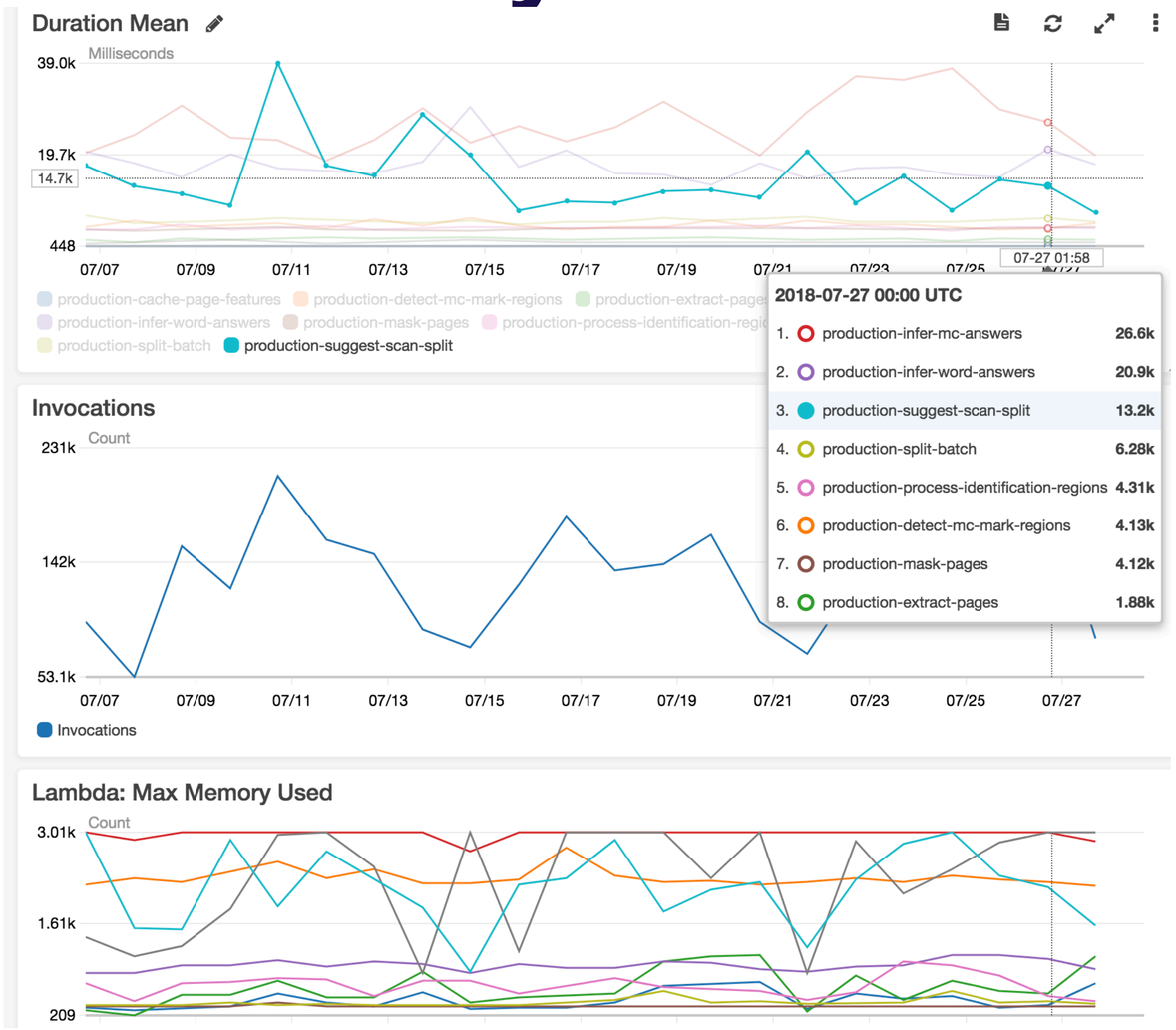
- Approximating model performance changes given different distribution changes

Questions?

Outline

- What should I monitor?
- How do I measure if it's changed?
- How do I tell if that change is bad?
- **Tools for monitoring**
- Monitoring and your broader ML system

System monitoring tools



Pingdom Server Monitor APP 17:13

Alert - % Memory Used met or exceeded 80%, increasing to 81% for 10 minutes at 05:10PM

- Alarms for when things go wrong, and records for tuning things.
- Cloud providers have decent monitoring solutions.
- Anything that can be logged can be monitored (i.e. data skew)
- Will explore in lab

System monitoring tools

Data quality tools

great_expectations

Anomalo

MC MONTE CARLO

ML monitoring

Questions?

Outline

- What should I monitor?
- How do I measure if it's changed?
- How do I tell if that change is bad?
- Tools for monitoring
- **Monitoring and your broader ML system**

Monitoring is more central to ML than to traditional software

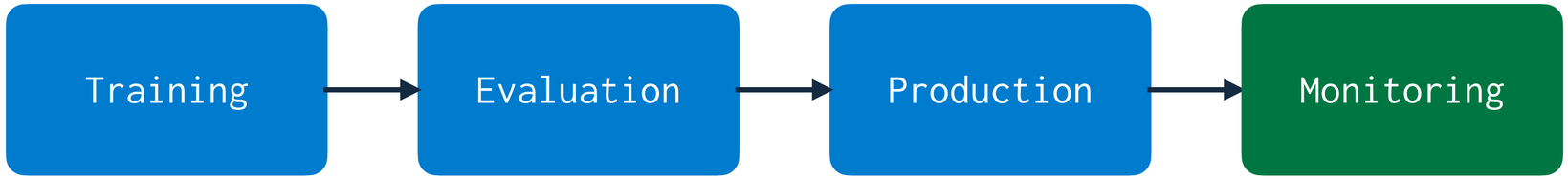
In traditional software…

- Bugs are usually obvious and/or cause loud failures
- The data you monitor (system metrics, etc) is *mostly valuable to detect & diagnose problems*

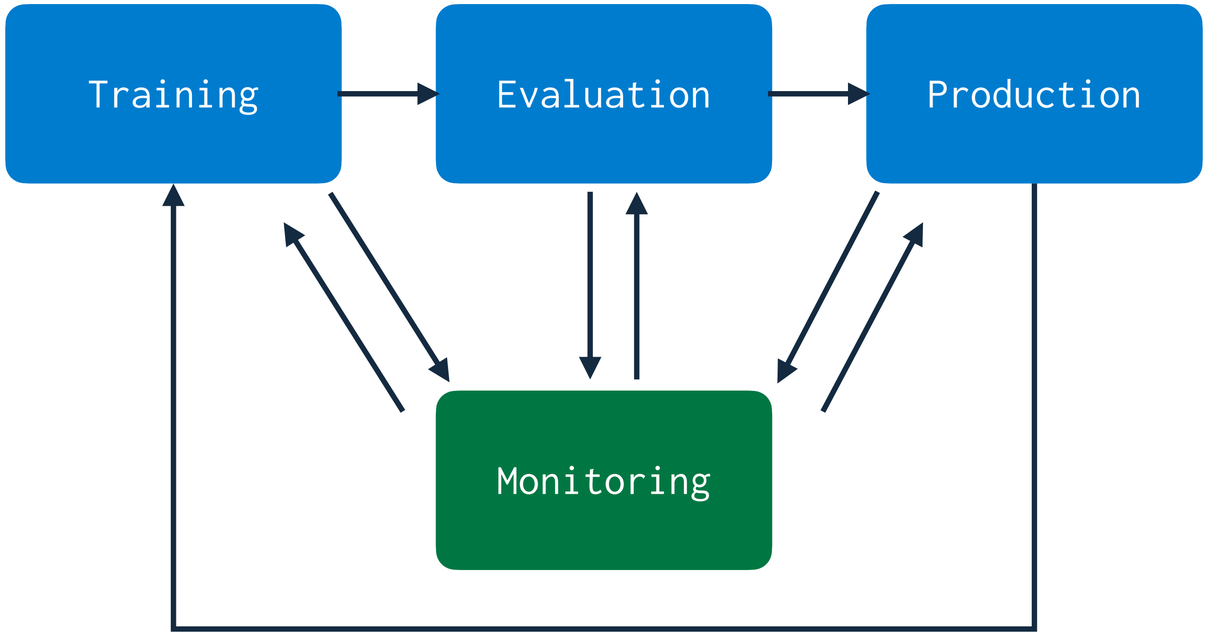
… in ML

- Bugs usually cause silent depredations in performance
- The data you monitor (system metrics, etc) is *literally the code used to produce the next model*

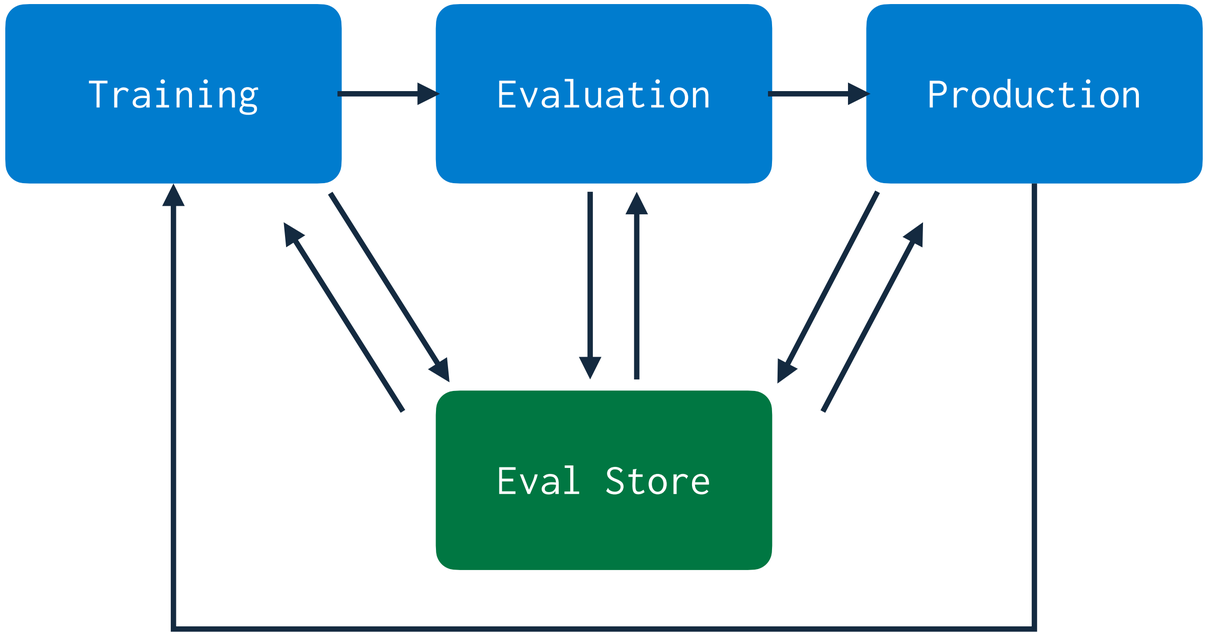
A traditional view of monitoring



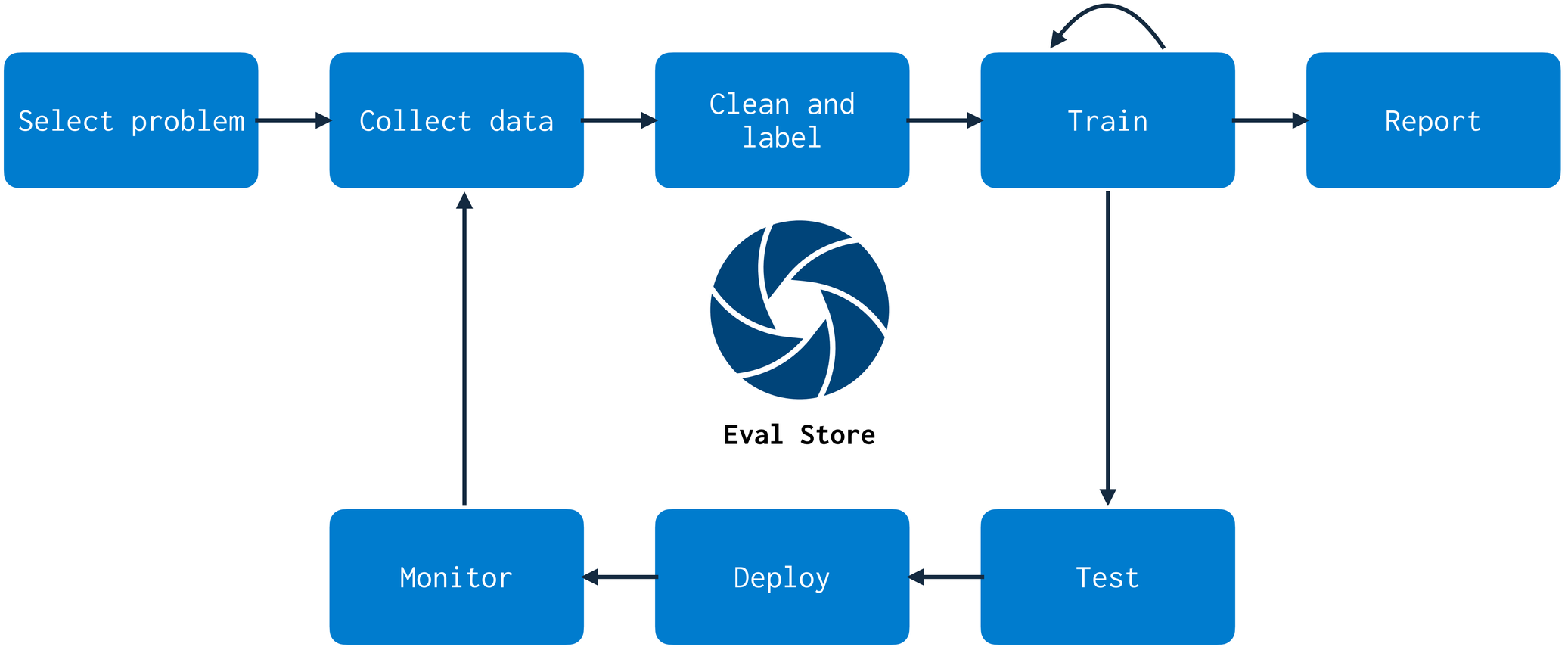
The reality of monitoring in ML



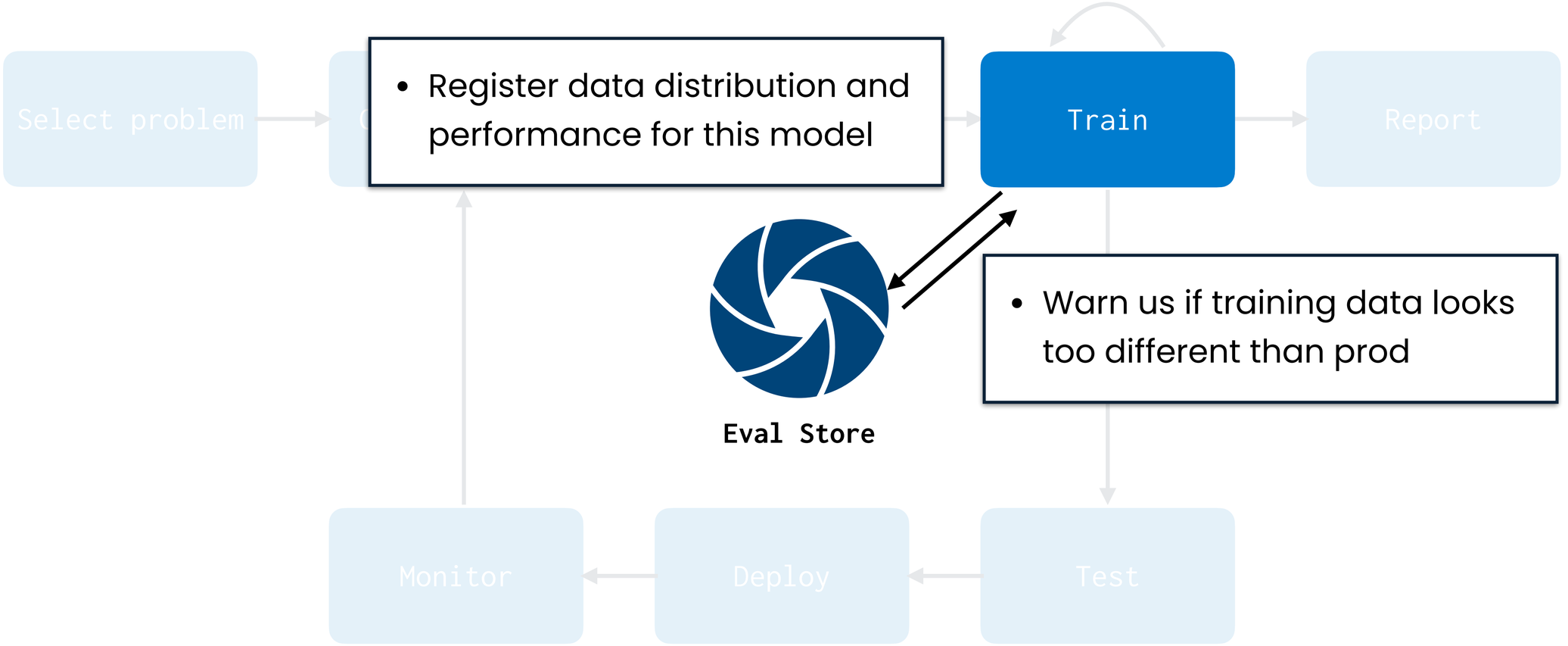
The reality of monitoring in ML



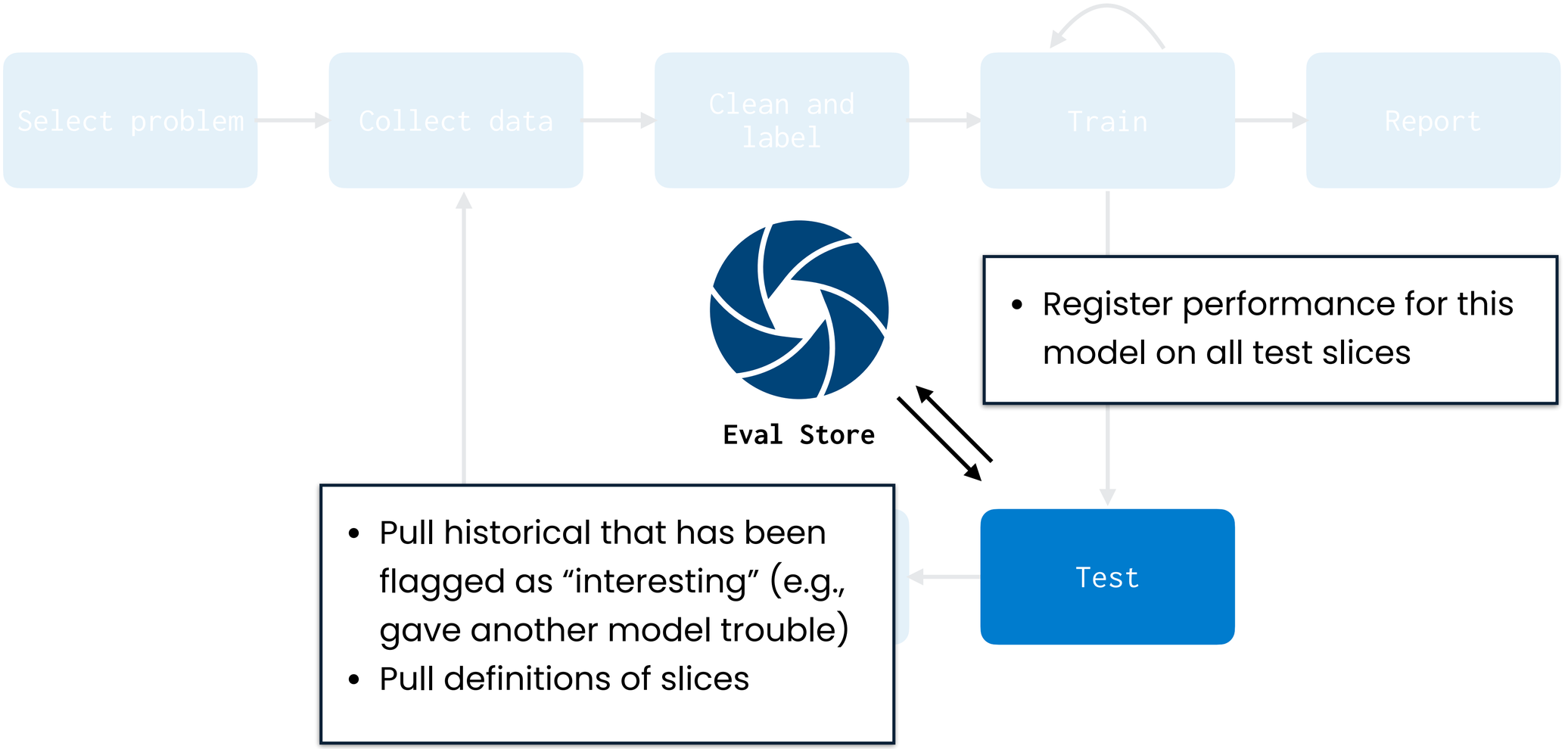
Closing the data flywheel



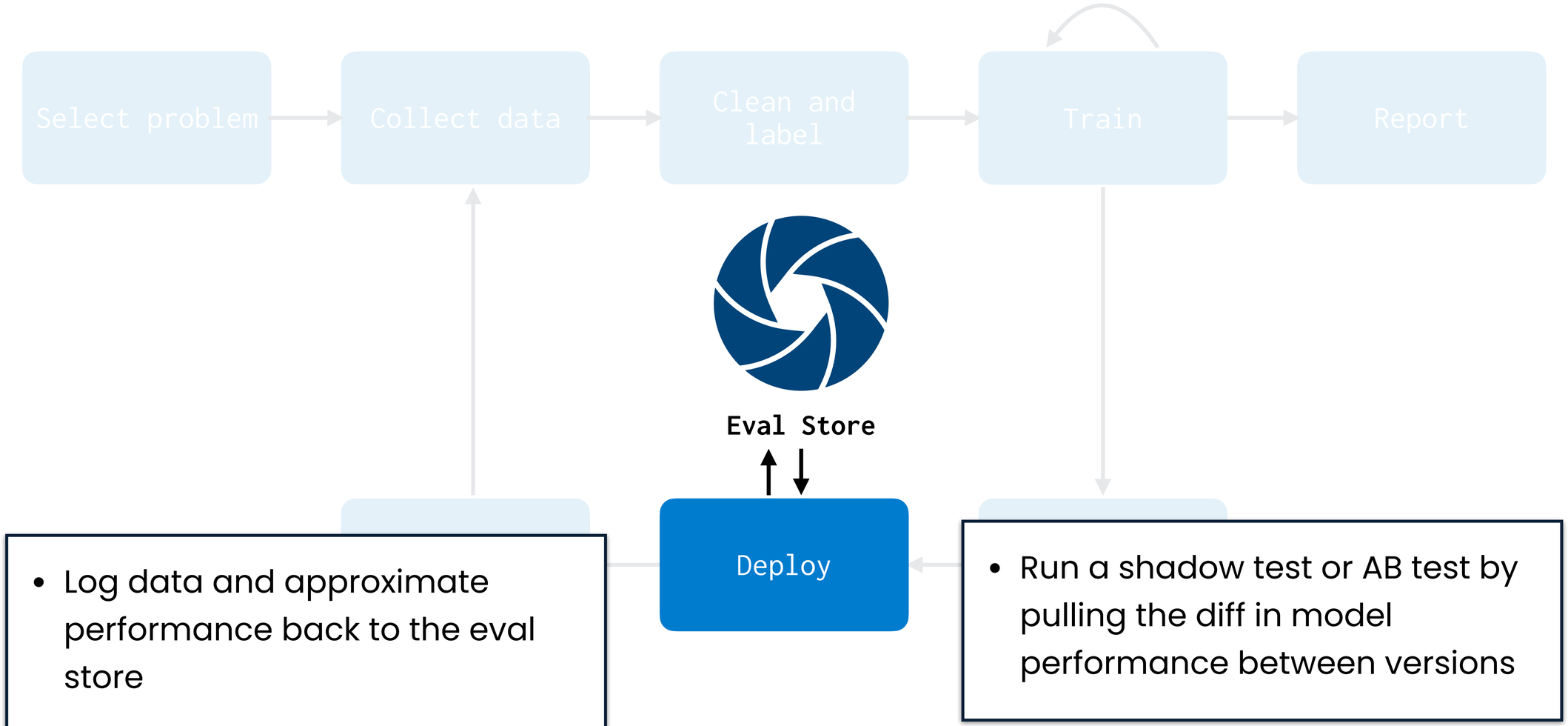
Closing the data flywheel



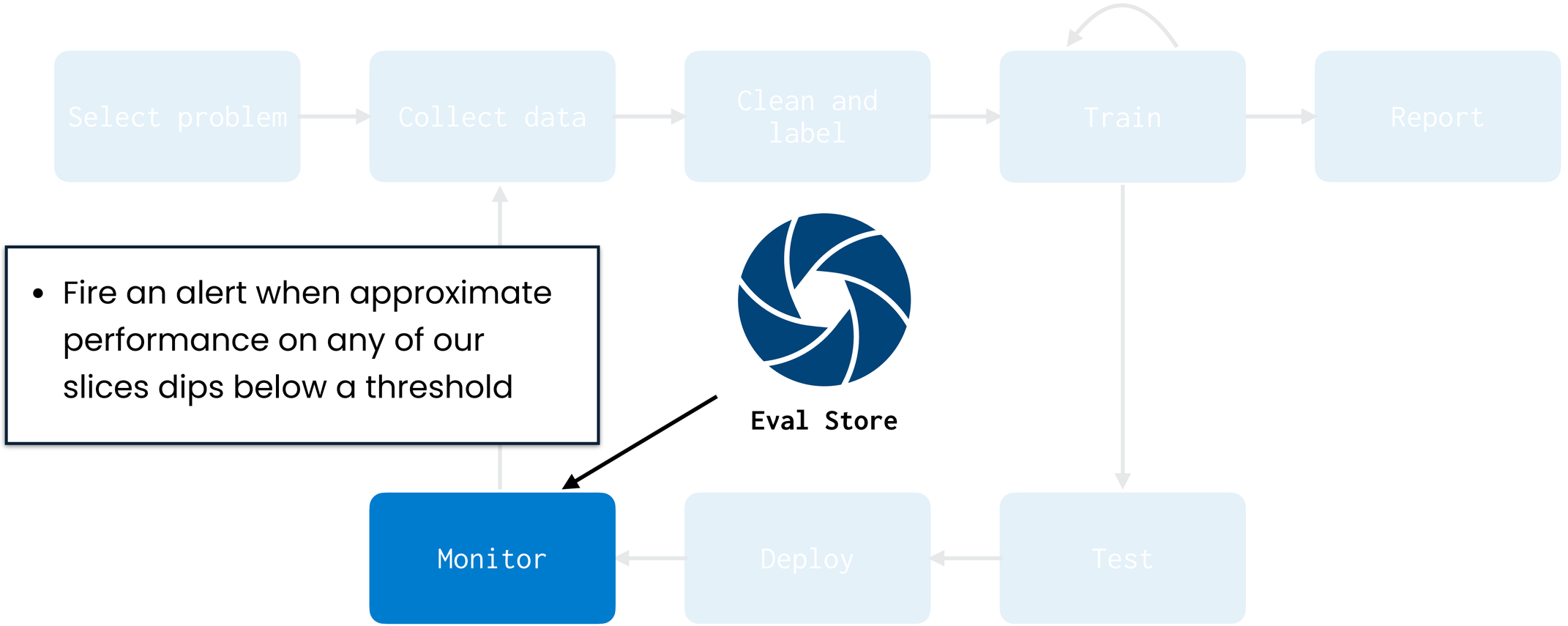
Closing the data flywheel



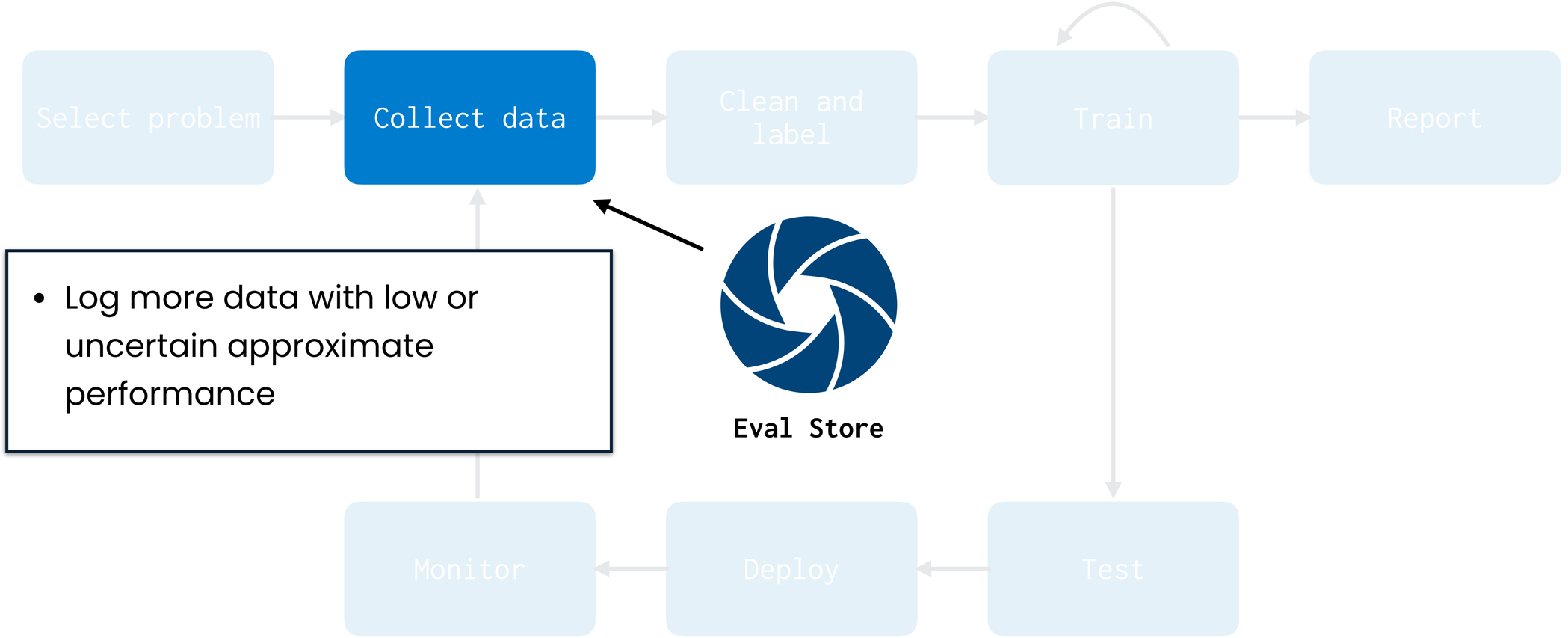
Closing the data flywheel



Closing the data flywheel

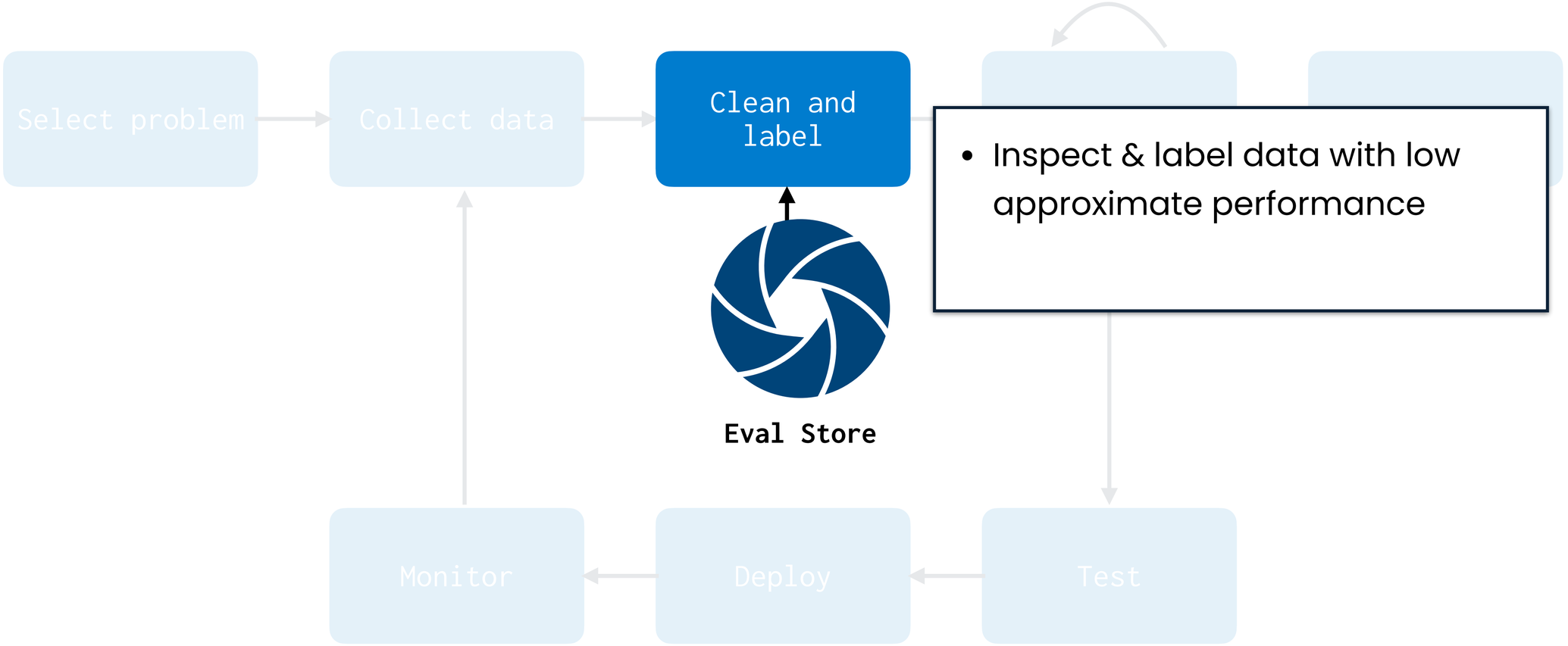


Closing the data flywheel

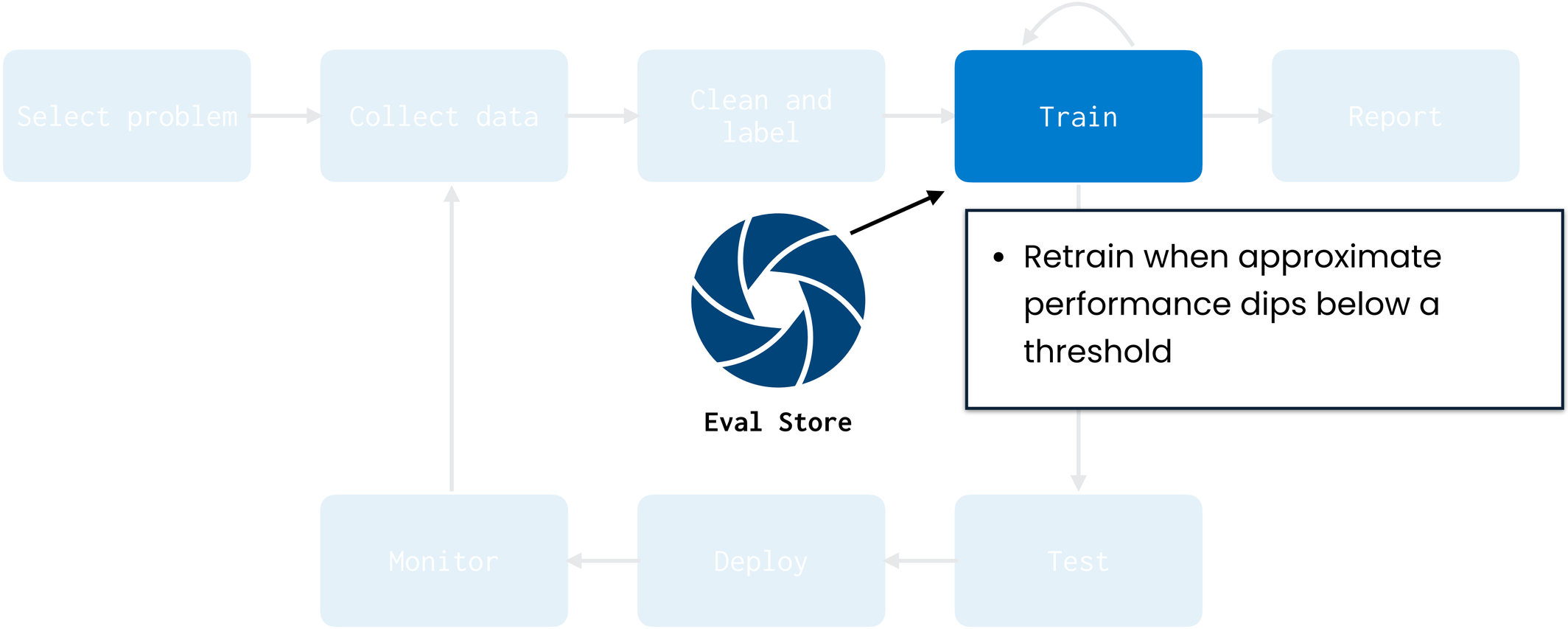


- Log more data with low or uncertain approximate performance

Closing the data flywheel



Closing the data flywheel



Questions?

Monitoring ML models: takeaways

- You should be monitoring your models
- Start by looking at data quality metrics and system metrics because it's easiest
- In a perfect world, your testing and monitoring systems should be linked, and should help you close the data flywheel
- Lots of activity in tooling — watch this space!
- Big open research questions still to be answered

Thank you!