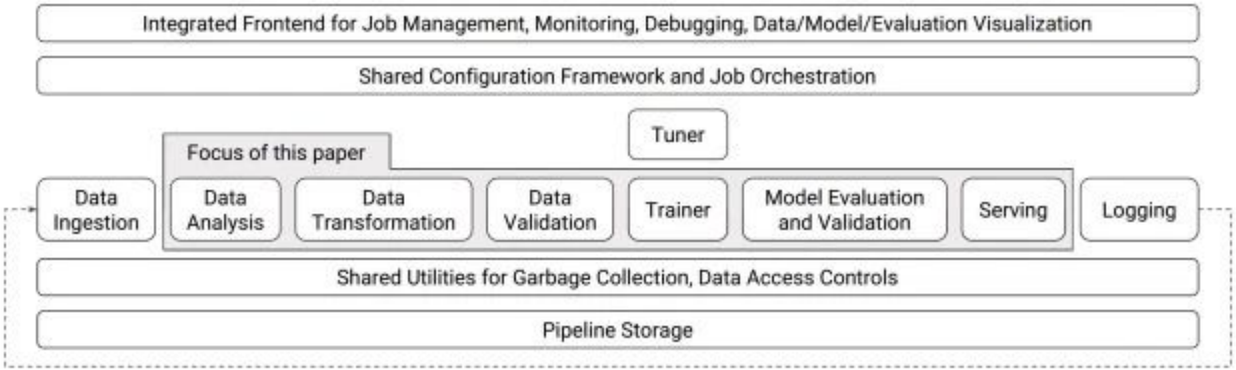
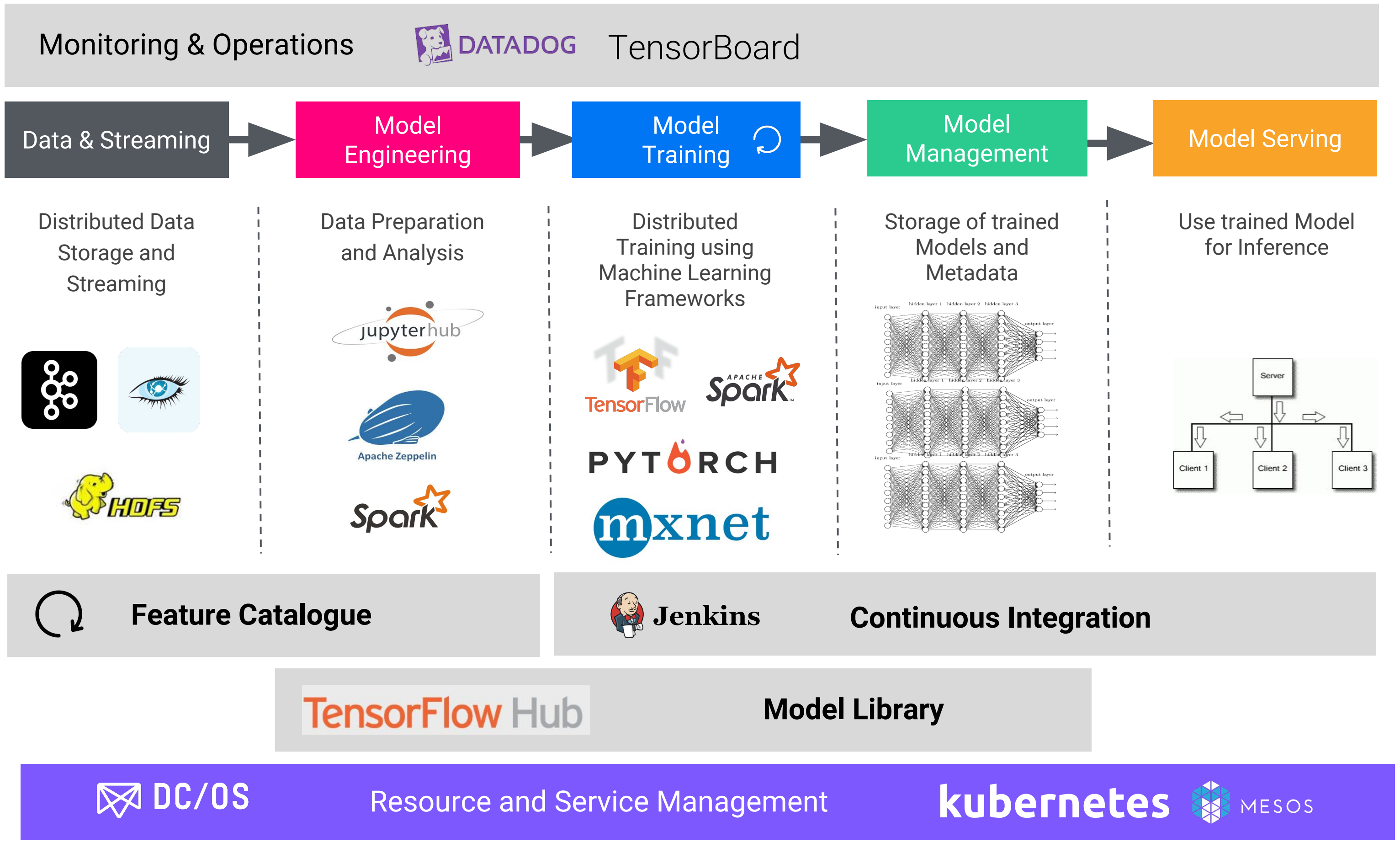
@joerg_schad

Kubeflow++
Building an Open Source Data Science Platform



MESOSPHERE

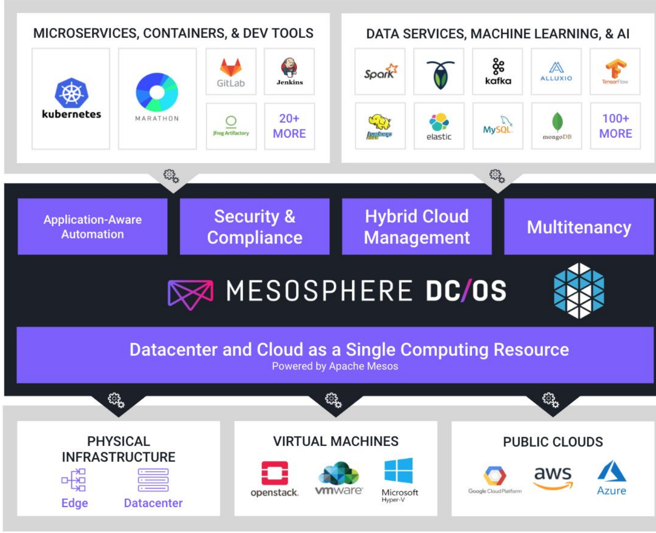

Kubeflow

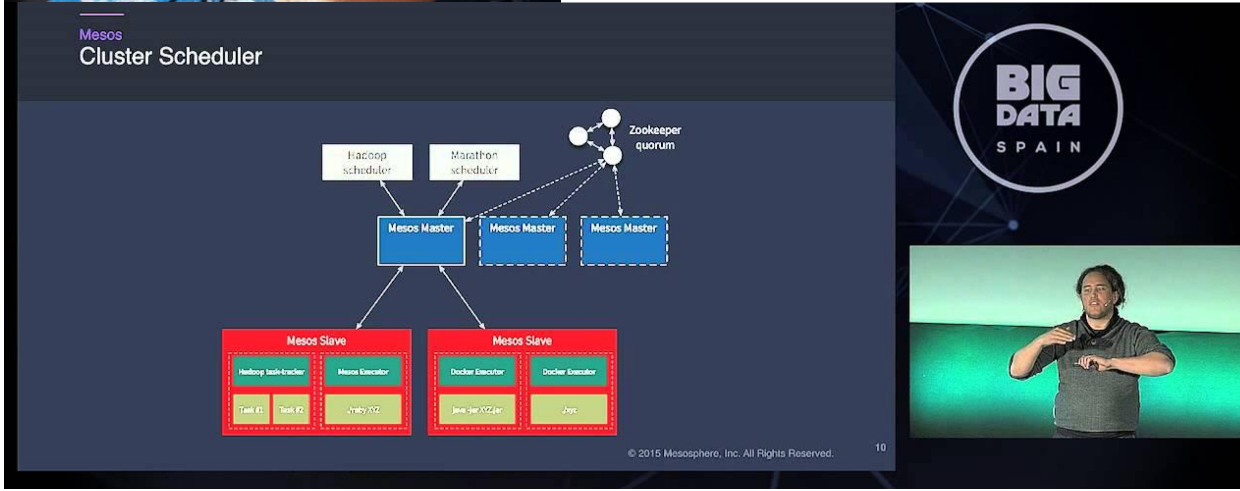

Monitoring & Operations
DATADOG
TensorBoard
Data & Streaming
Model Engineering
Model Training
Model Management
Model Serving
Distributed Data Storage and Streaming
Data Preparation and Analysis
Distributed Training using Machine Learning Frameworks
Storage of trained Models and Metadata
Use trained Model for Inference
HDFS
jupyterhub
Apache Zeppelin
Spark
TensorFlow
APACHE Spark
PYTORCH
mxnet
Server
Client 1
Client 2
Client 3
Feature Catalogue
Jenkins
Continuous Integration
TensorFlow Hub
Model Library
DC/OS
Resource and Service Management
kubernetes
MESOS

Jörg Schad

Technical Lead/Engineer
Deep Learning

- Core Mesos developer at Mesosphere
- Twitter: @joerg_schad





MESOSPHERE

Why is machine learning taking off?


BOSCH

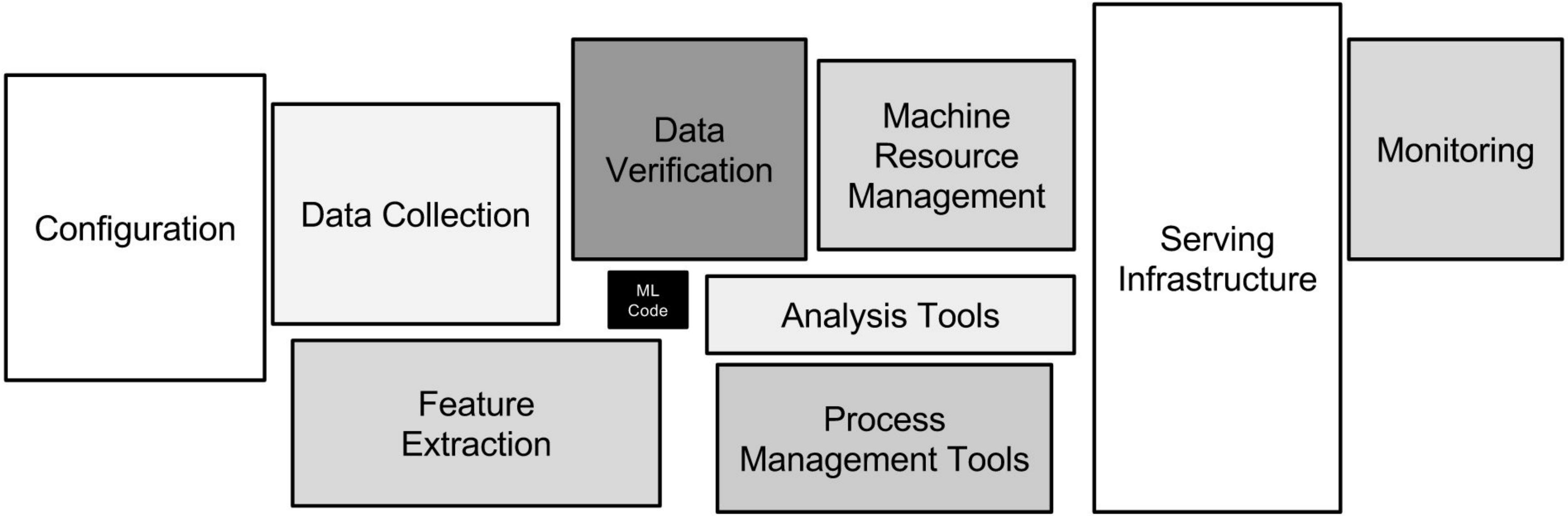
DEEPBACH: A STEERABLE MODEL FOR BACH CHORALES GENERATION

What you want to do be doing

Get Data → Write intelligent machine learning code → Train Model → Run Model

Repeat

What you're actually doing



Sculley, D., Holt, G., Golovin, D. et al. Hidden Technical Debt in Machine Learning Systems

Kubeflow

1. Data Preparation & Model Engineering

2. Model Training

3. Monitoring

4. Debugging

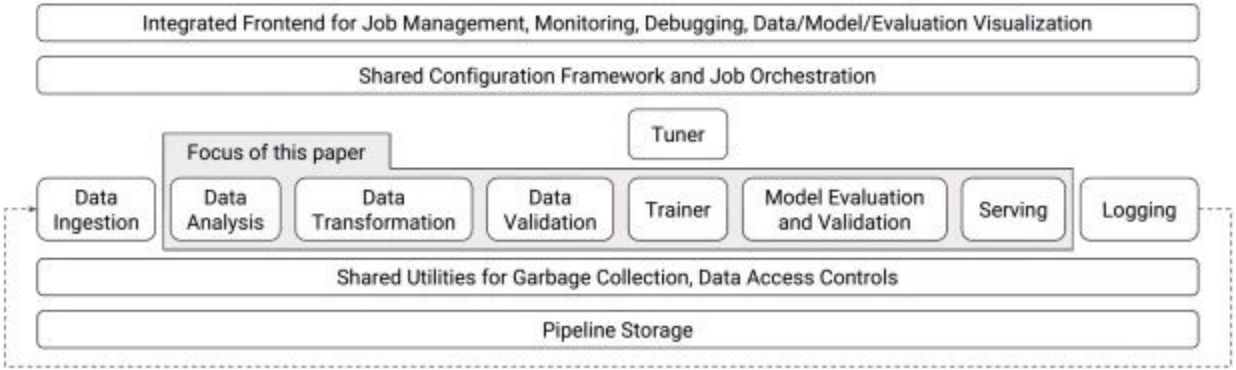
5. Model Serving

Kubeflow

The Kubeflow project is dedicated to making deployments of machine learning (ML) workflows on Kubernetes simple, portable and scalable.

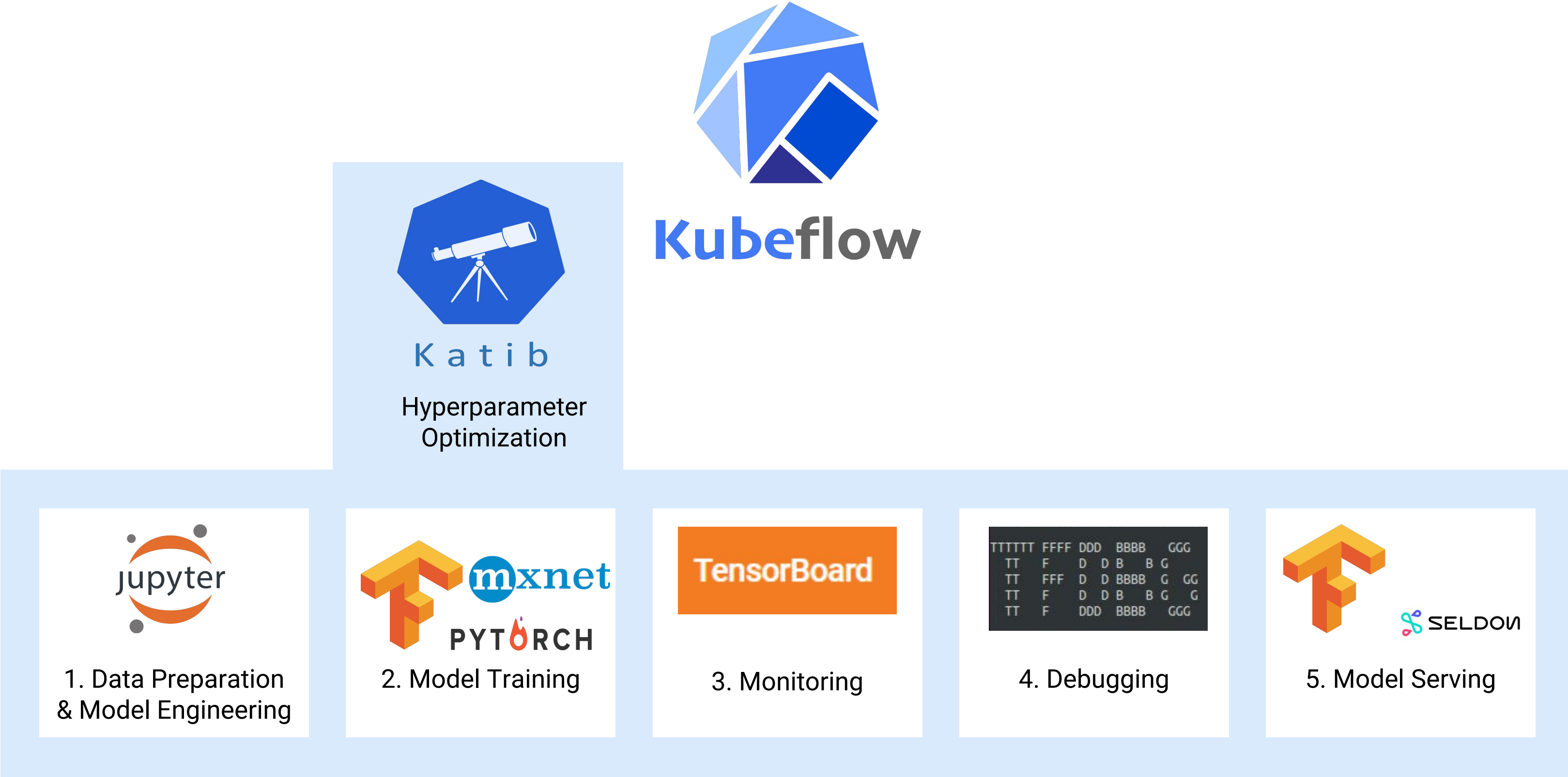
https://www.kubeflow.org/docs/about/kubeflow/

TFX: A TensorFlow-Based Production-Scale Machine Learning Platform

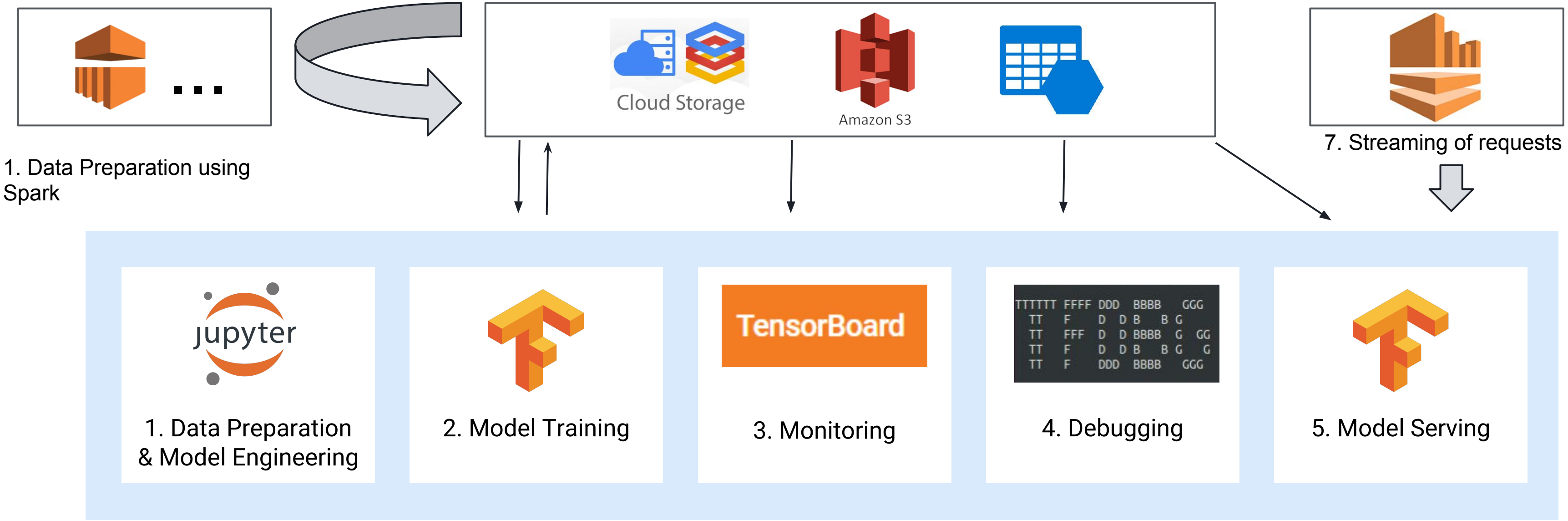


http://www.kdd.org/kdd2017/papers/view/tfx-a-tensorflow-based-production-scale-machine-learning-platform

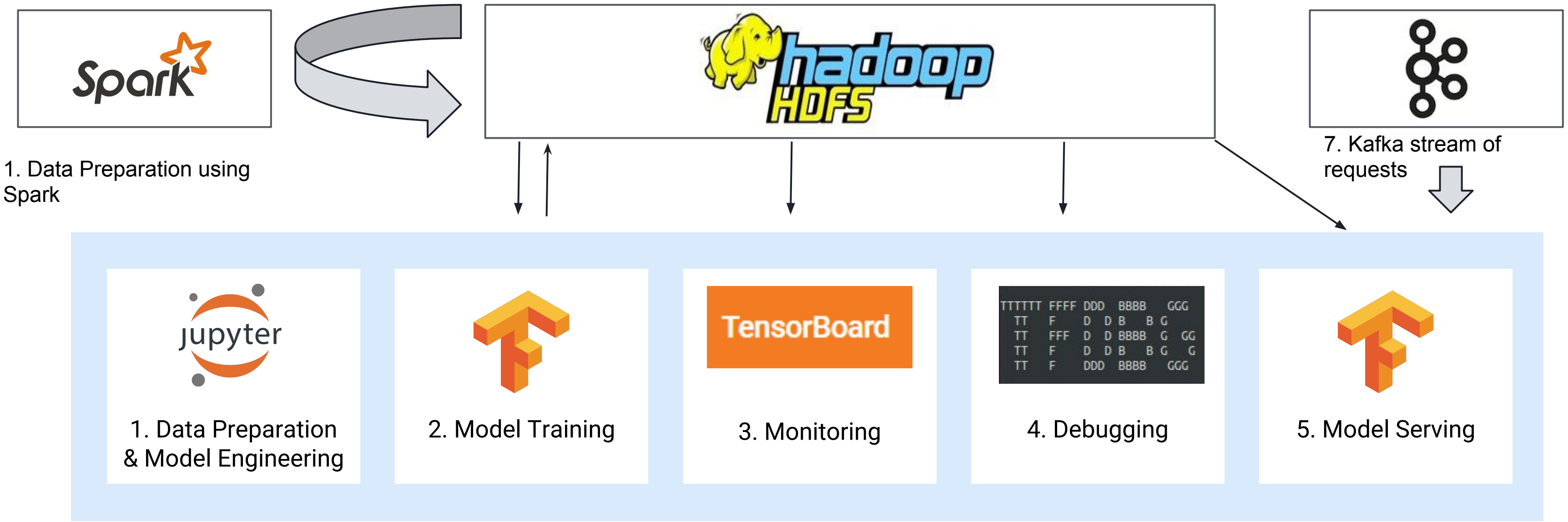
https://www.youtube.com/watch?v=fPTwLVCq00U


Kubeflow
Katib
Hyperparameter Optimization
jupyter
1. Data Preparation & Model Engineering
mxnet
PYTORCH
2. Model Training
TensorBoard
3. Monitoring
4. Debugging
SELDON
5. Model Serving

Public Cloud Pipeline



DIY Open Source Pipeline



Monitoring & Operations DATADOG TensorBoard

Data & Streaming → Model Engineering → Model Training → Model Management → Model Serving

Data & Streaming

Distributed Data Storage and Streaming

HDFS

Model Engineering

Data Preparation and Analysis

jupyterhub

Apache Zeppelin

Spark

Model Training

Distributed Training using Machine Learning Frameworks

TensorFlow

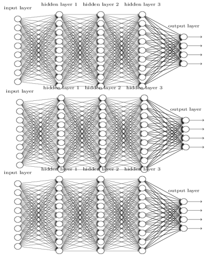
APACHE Spark

PYTORCH

mxnet

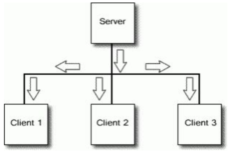
Model Management

Storage of trained Models and Metadata



Model Serving

Use trained Model for Inference



Feature Catalogue

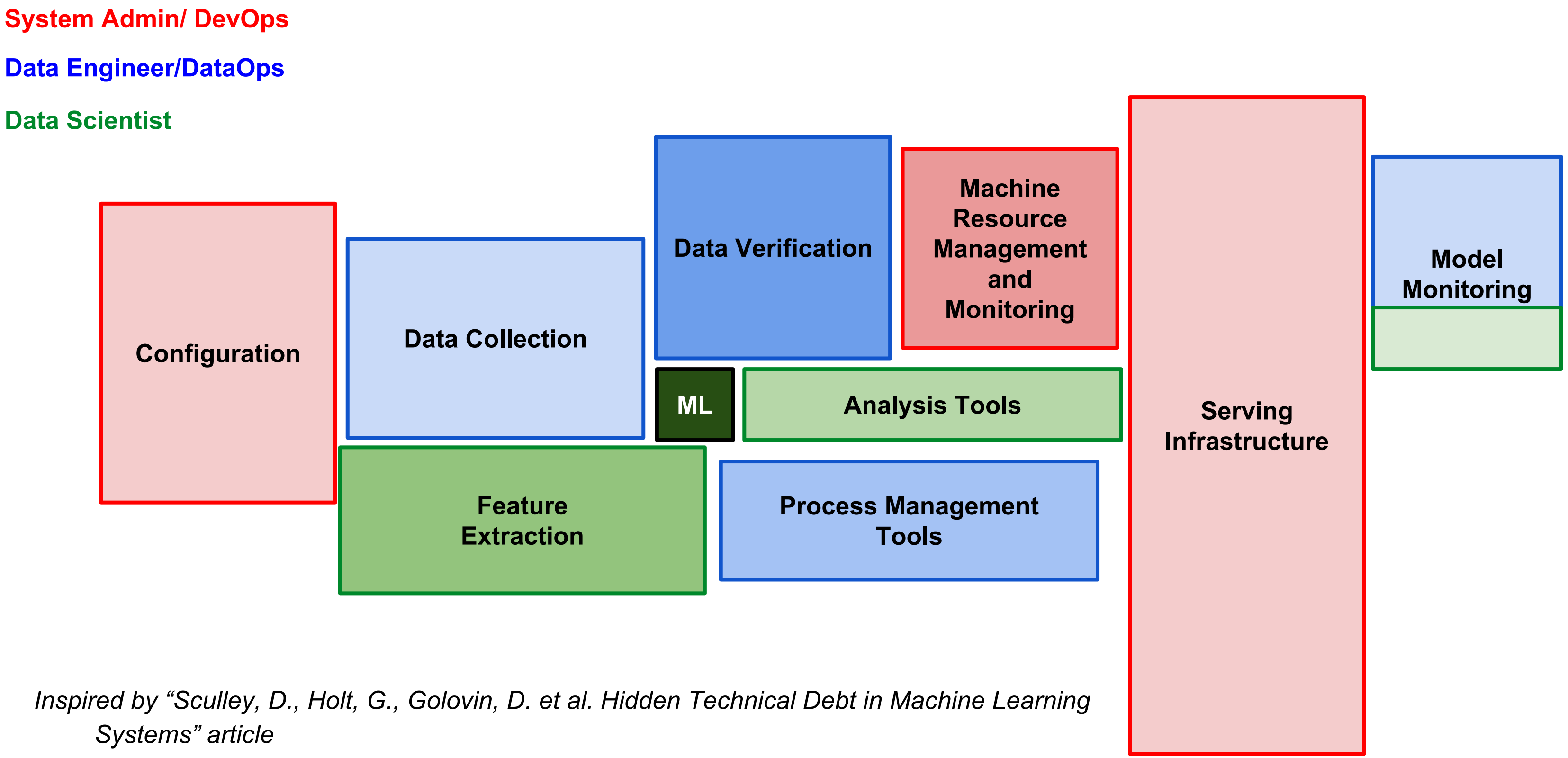
Jenkins **Continuous Integration**

TensorFlow Hub **Model Library**

DC/OS Resource and Service Management kubernetes MESOS

Challenge: Persona(s)

Division of Labor



Inspired by "Sculley, D., Holt, G., Golovin, D. et al. Hidden Technical Debt in Machine Learning Systems" article

The Rise of the *DataOps Engineer*

Combines two key skills:

- Data science
- Distributed systems engineering

The equivalent of *DevOps* for *Data Science*

SOFTWARE ENGINEERING

Report on a conference sponsored by the

NATO SCIENCE COMMITTEE

Garmisch, Germany, 7th to 11th October 1968

Chairman: Professor Dr. F. L. Bauer

Co-chairmen: Professor L. Bolliet, Dr. H. J. Helms

Editors: Peter Naur and Brian Randell

Software Engineering

The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software

IEEE Standard Glossary of Software Engineering Terminology

Do we need Data Science Engineering Principles?

Do

ering


Ian Goodfellow
@goodfellow_ian
Follow
Replying to @rctatman
I have a different controversial opinion: ML development is a different kind of software development and has a different set of best practices.
7:59 PM - 18 Sep 2018

Challenge: Requirements Engineering

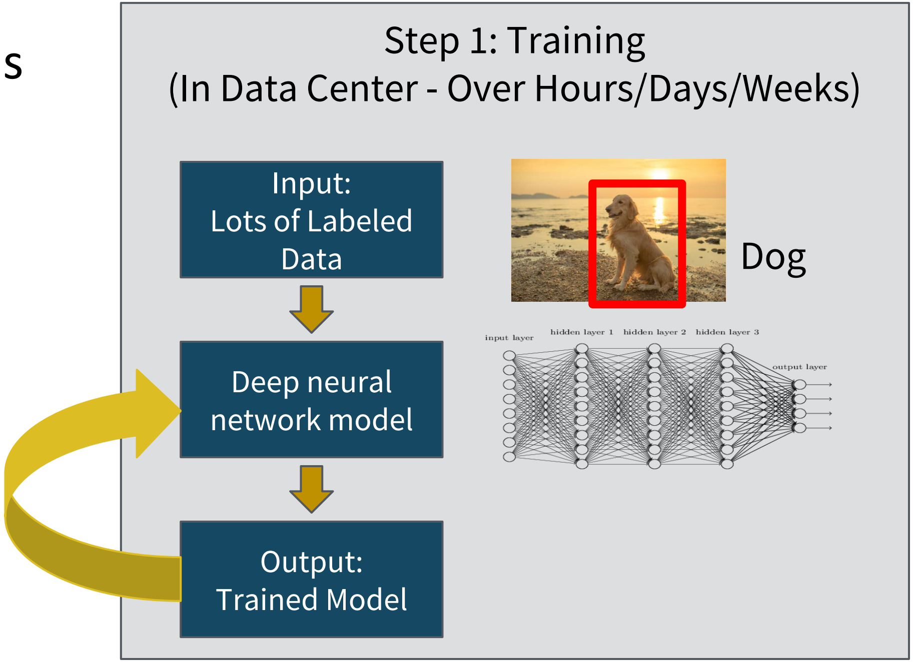
- Do I need Machine Learning? *
- Do I need {Neural Networks, Regression,...}*

- What dataset(s)?
  - Quality?
- What target/serving environment?
- What model architecture?
- Pre-trained model available?
- How many training resources?

* Can I actually use ...

Challenge: Reproducible Builds

- Many adhocs model/training runs
- Regulatory Requirements
- Dependencies
- CI/CD
- Git



MFlow

mlflow

Tracking

Record and query experiments: code, data, config, results

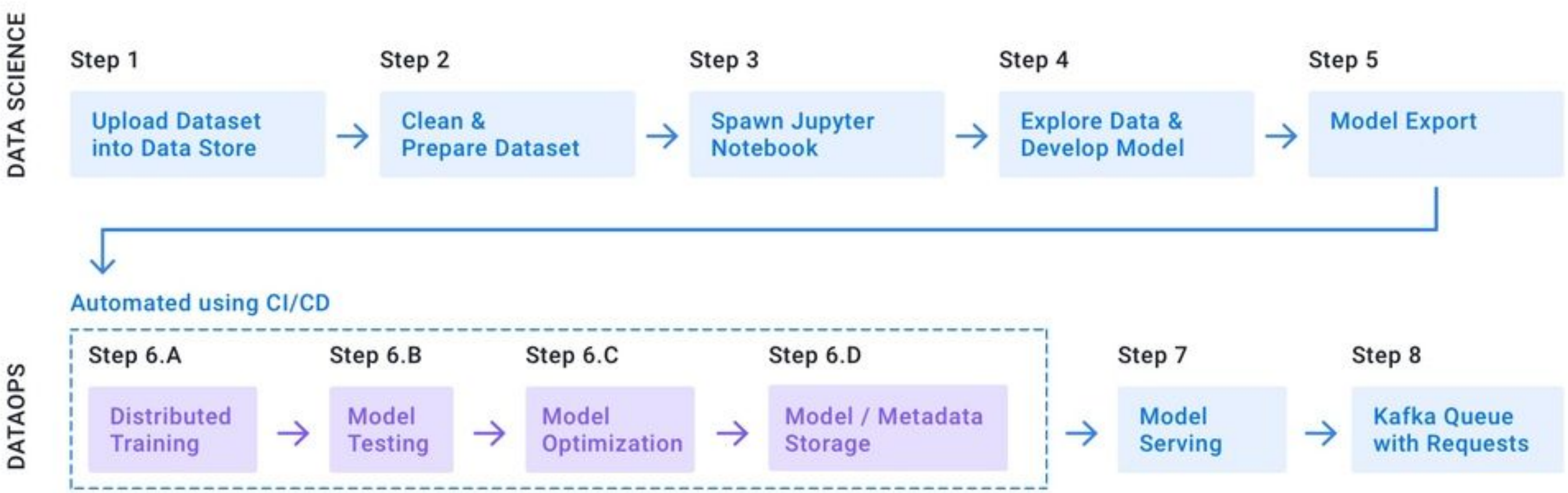
Projects

Packaging format for reproducible runs on any platform

Models

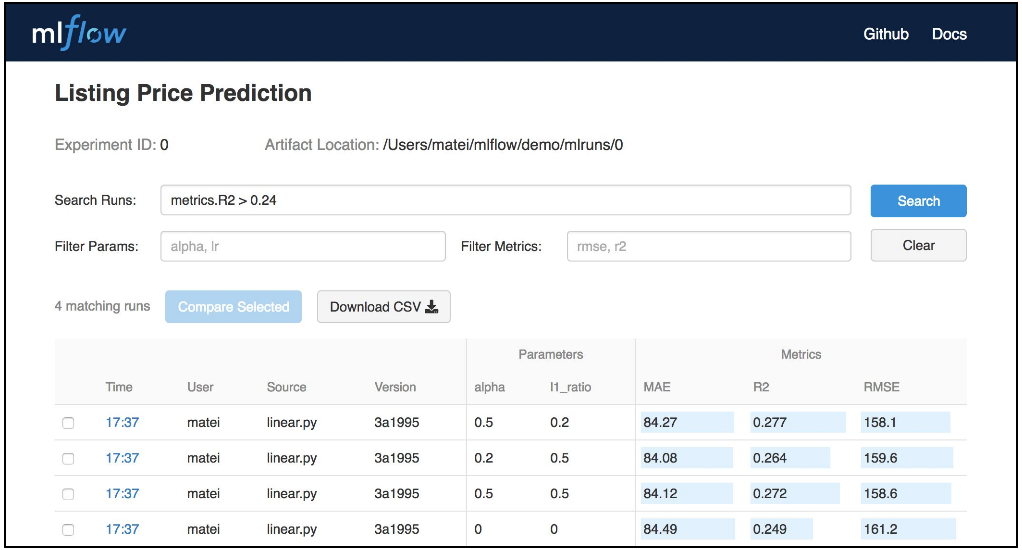
General format for sending models to diverse deploy tools

Challenge: Automation & CI/CD



MFlow Tracking

```python
import mlflow


# Log parameters (key-value pairs)
mlflow.log_param("num_dimensions", 8)
mlflow.log_param("regularization", 0.1)


# Log a metric;
mlflow.log_metric("accuracy", 0.1)
...
mlflow.log_metric("accuracy", 0.45)


# Log artifacts (output files)
mlflow.log_artifact("roc.png")
mlflow.log_artifact("model.pkl")
```



MFlow Project

```yaml
name: My Project
conda_env: conda.yaml
entry_points:
  main:
    parameters:
      data_file: path
      regularization: {type: float, default:
0.1}
    command: "python train.py -r
{regularization} {data_file}"
  validate:
    parameters:
      data_file: path
    command: "python validate.py {data_file}"
```

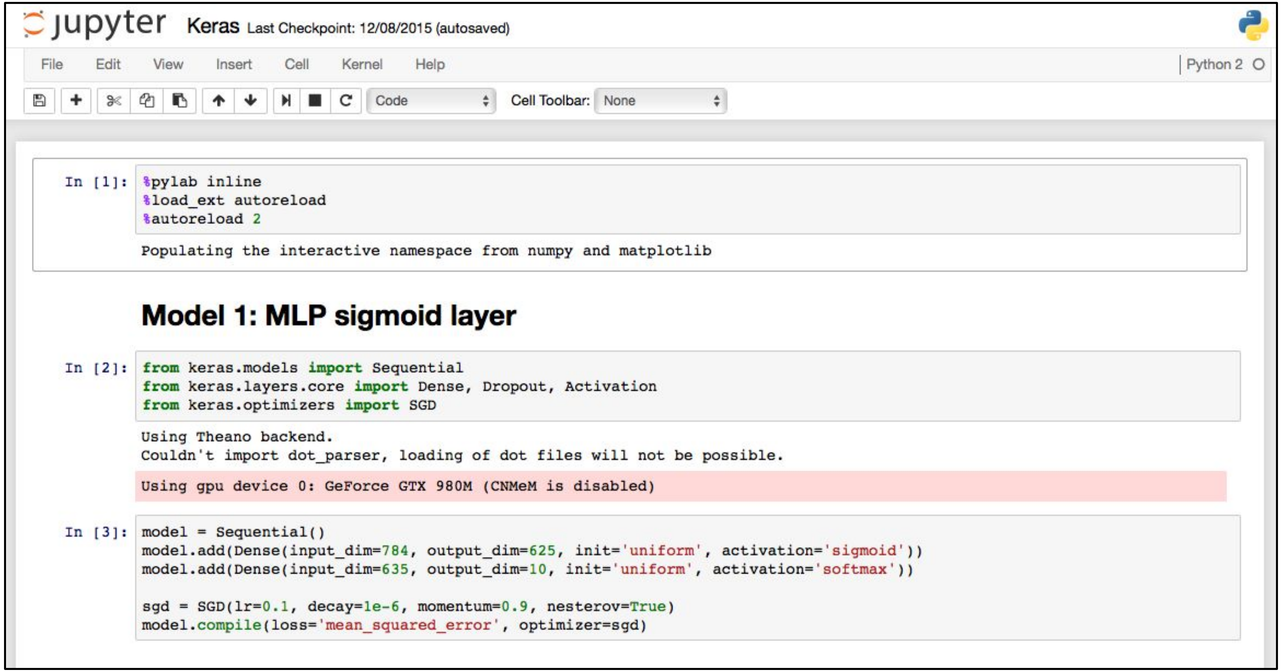
```
$mlflow run example/project -P alpha=0.5
$mlflow run git@github.com:databricks/mlflow-example.git
```

MFlow Model

```yaml
time_created: 2018-02-21T13:21:34.12
flavors:
  sklearn:
    sklearn_version: 0.19.1
    pickled_model: model.pkl
  python_function:
    loader_module: mlflow.sklearn
    pickled_model: model.pkl
```

```
$mlflow run example/project -P alpha=0.5
$mlflow run git@github.com:databricks/mlflow-example.git
```

Challenge: Data Science IDE



Challenge: Data Quality

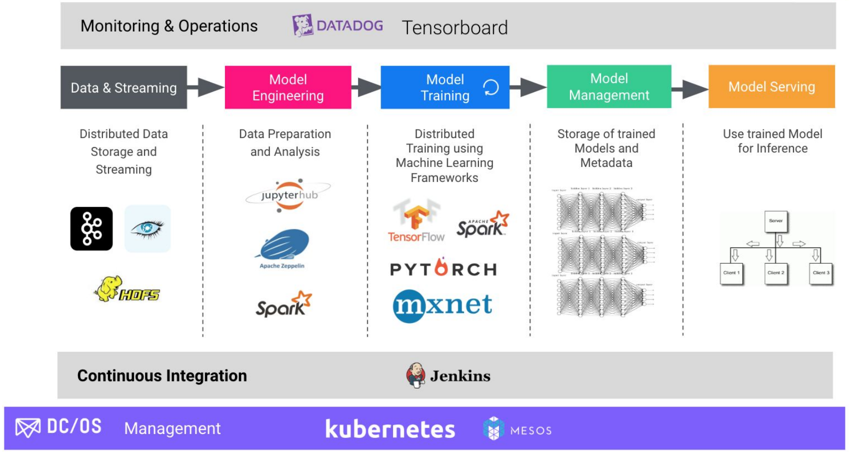
- Data is typically not ready to be consumed by ML job*
  - Data Cleaning
    - Missing/incorrect labels
  - Data Preparation
    - Same Format
    - Same Distribution

* Demo datasets are a fortunate exception :)

Challenge: Data Quality

- Data is typically not ready to be consumed by ML job*
  - Data Cleaning
    - Missing/incorrect labels
  - Data Preparation
    - Same Format
    - Same Distribution

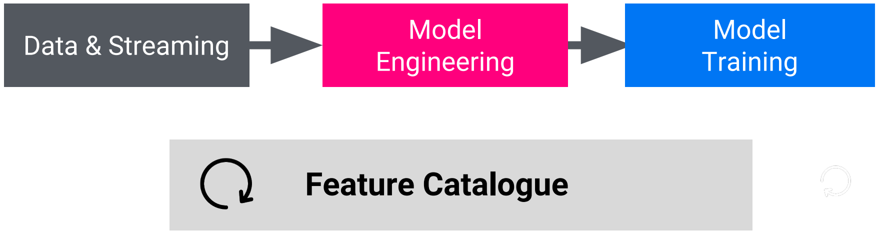
Don't forget about the serving environment!!



* Demo datasets are a fortunate exception :)

Challenge: Data (Preprocessing) Sharing

- Preprocessed Data Sets valuable
  - Sharing
  - Automatic Updating
- Feature Catalogue ≊ Preprocessing Cache + Discovery



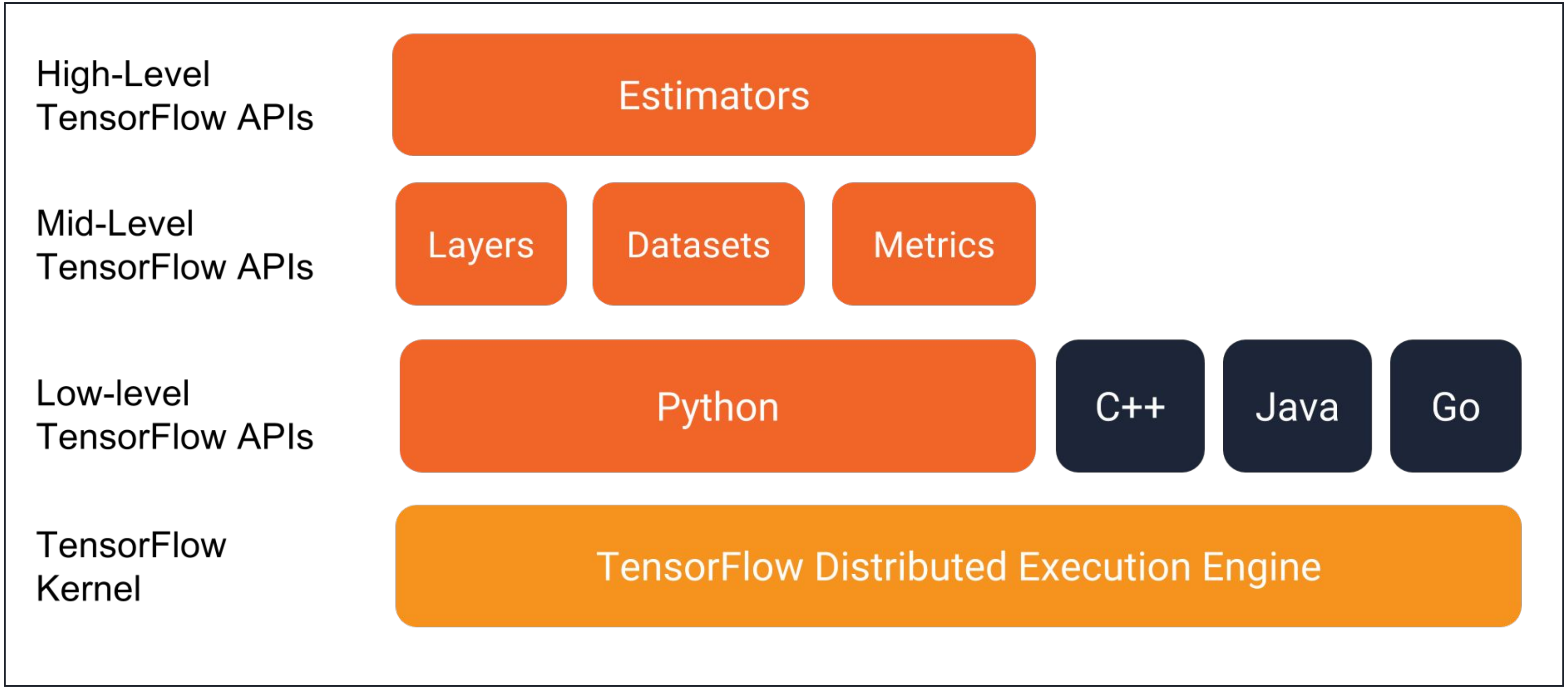
https://eng.uber.com/michelangelo/

Challenge: Model Libraries

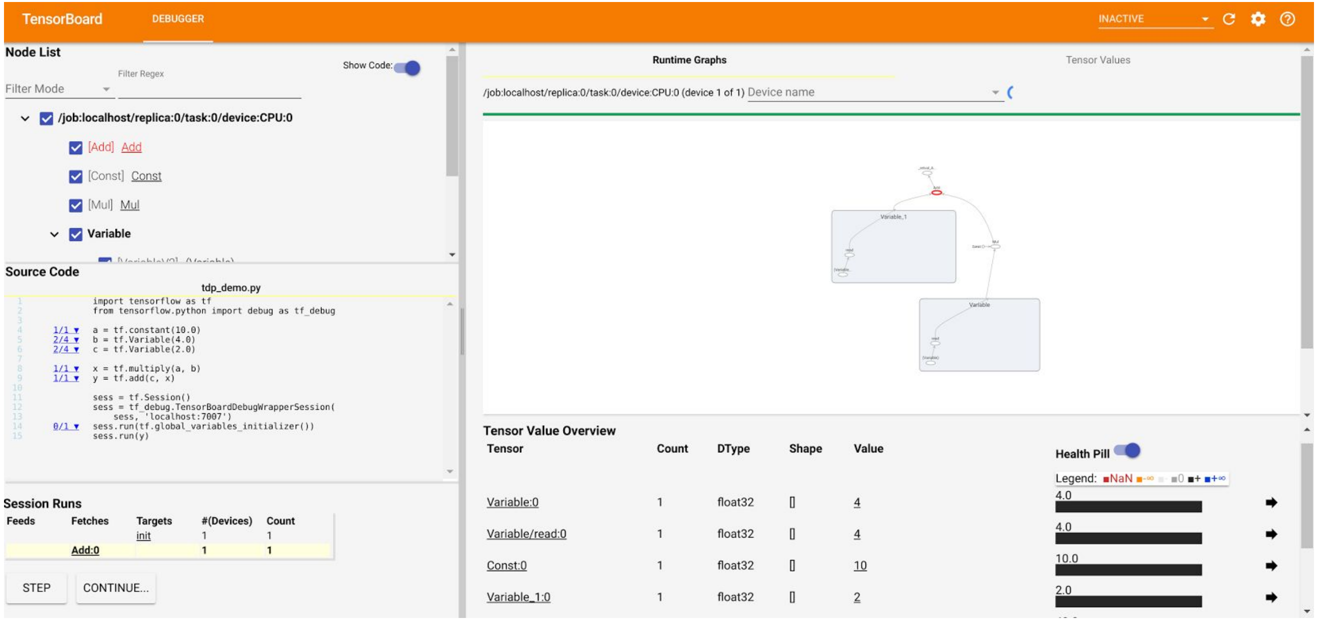
- Existing architectures
- Pretrained models

```
import tensorflow as tf
import tensorflow_hub as hub

with tf.Graph().as_default():
  embed = hub.Module("https://tfhub.dev/google/nnlm-en-dim128-with-normalization/1")
  embeddings = embed(["A long sentence.", "single-word", "http://example.com"])

  with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    sess.run(tf.tables_initializer())

    print(sess.run(embeddings))
```

Challenge: Writing Distributed Model Functions



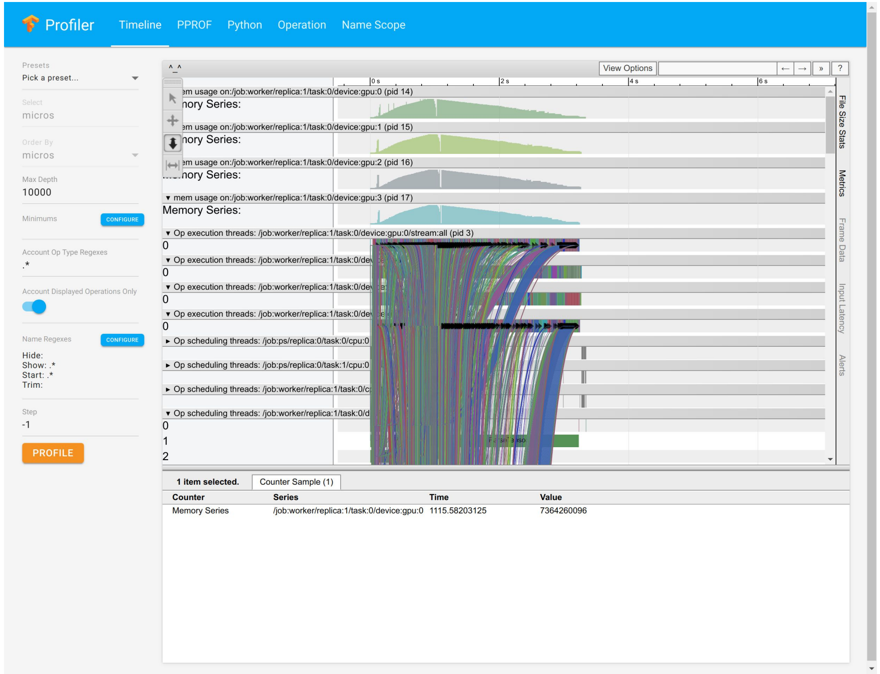
Challenge: Debugging



https://www.tensorflow.org/programmers_guide/debugger

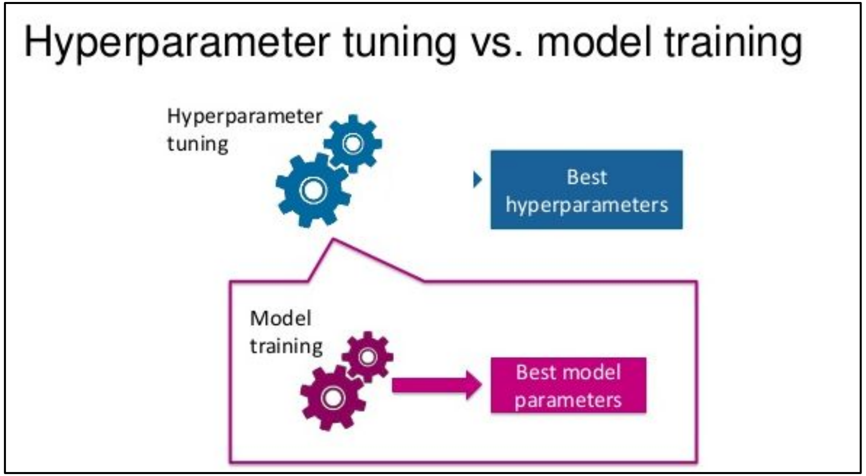
Profiling

- Crucial when using “expensive” devices
- Memory Access Pattern
- “Secret knowledge”
- More is not necessarily better....

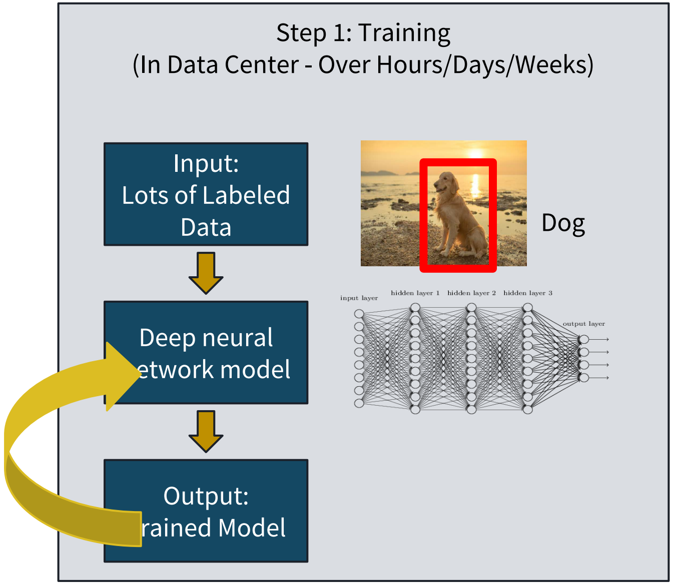


https://www.tensorflow.org/performance/performance_guide

Hyperparameter Optimization



- Networks Shape
- Learning Rate
- ...

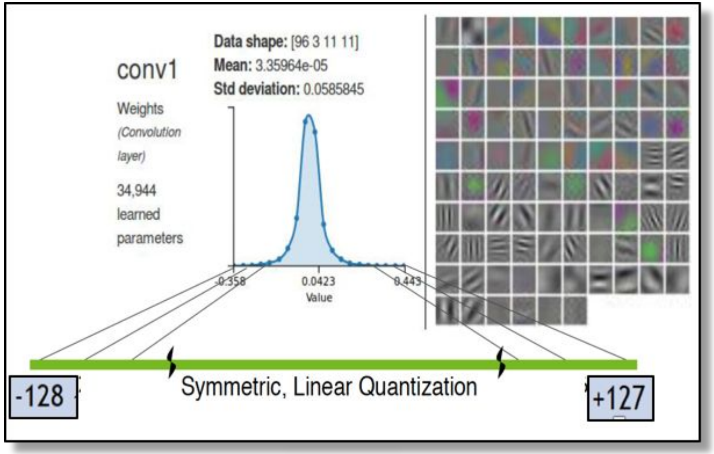


https://towardsdatascience.com/understanding-hyperparameters-and-its-optimisation-techniques-f0debba07568

Model Optimization

```
transform_graph \
  --in_graph=unoptimized_cpu_graph.pb \  ← Original Graph
  --out_graph=optimized_cpu_graph.pb \   ← Transformed Graph
  --inputs='x_observed:0' \              ← Feed (Input)
  --outputs='Add:0' \                    ← Fetch (Output)
  --transforms='                         ← List of Transforms
       strip_unused_nodes
       remove_nodes(op=Identity, op=CheckNumerics)
       fold_constants(ignore_errors=true)
       fold_batch_norms
       fold_old_batch_norms
       quantize_weights
       quantize_nodes'
```

Model Optimization

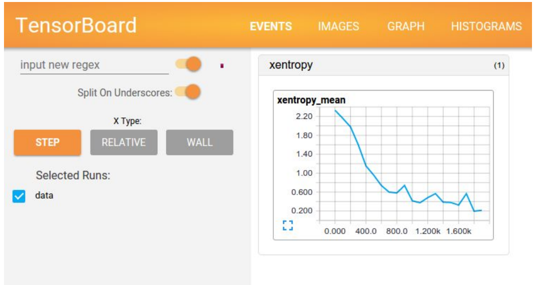


	Dynamic Range	Min Pos Value
FP32	-3.4×10^{38} ~ $+3.4\times10^{38}$	1.4×10^{-45}
FP16	-65504 ~ +65504	5.96×10^{-8}
INT8	-128 ~ +127	1

Challenge: Monitoring

- Understand {...}
- Debug
- Model Quality
  - Accuracy
  - Training Time
  - ...
- Overall Architecture
  - Availability
  - Latencies
  - ...

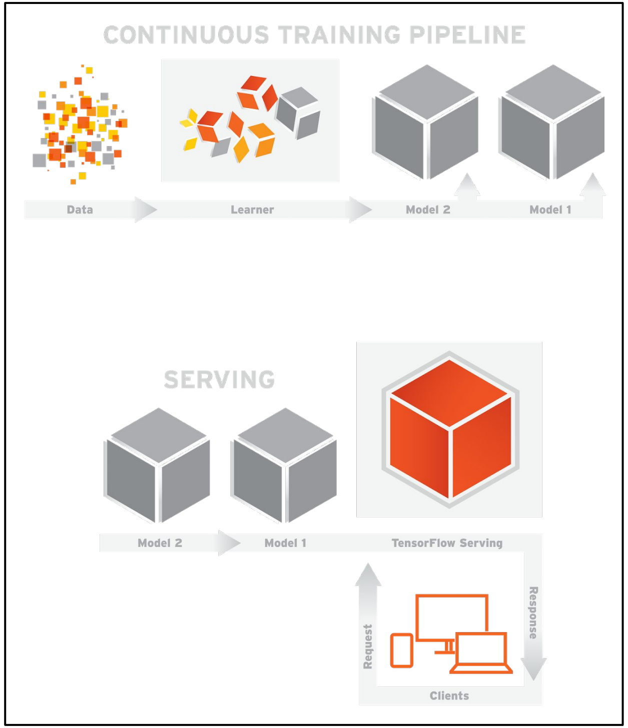
- TensorBoard



- Traditional Cluster Monitoring Tool

Challenge: Serving Environment

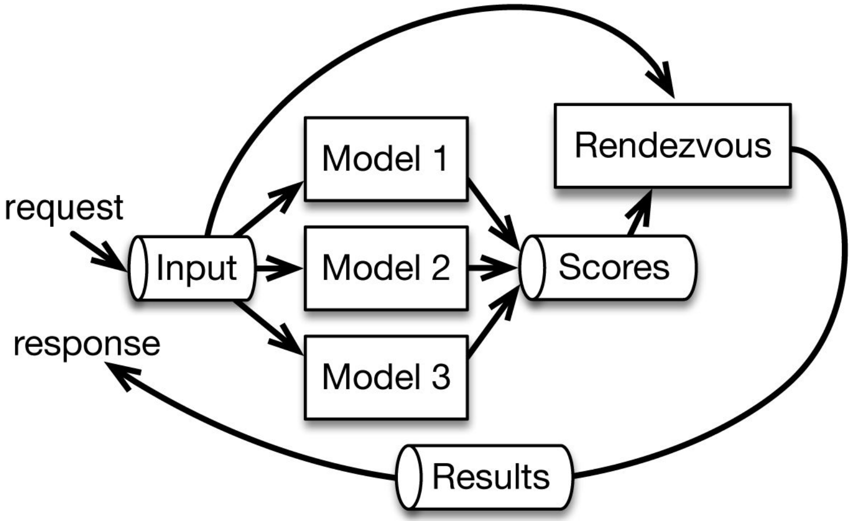
- How to Deploy Models?
  - Zero Downtime
  - Canary
- Multiple Models?
  - Testing



https://ai.googleblog.com/2016/02/running-your-models-in-production-with.html

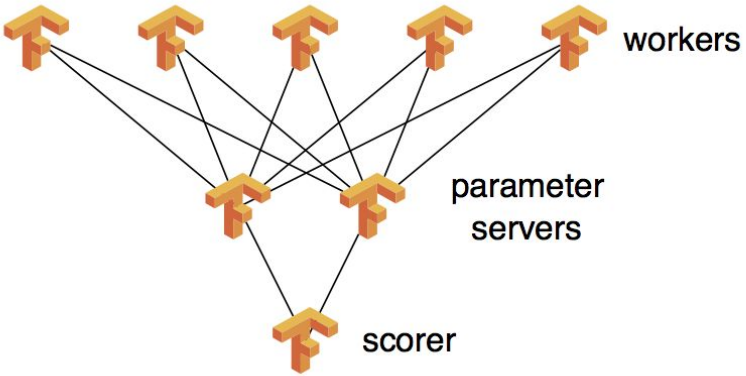
Challenge: Serving Environment

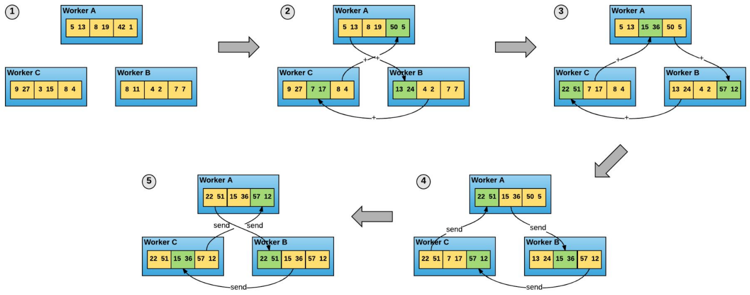
- How to Deploy Models?
  - Zero Downtime
  - Canary
- Multiple Models?
  - Testing



https://mapr.com/ebooks/machine-learning-logistics/ch03.html

Challenge: Distributed TensorFlow

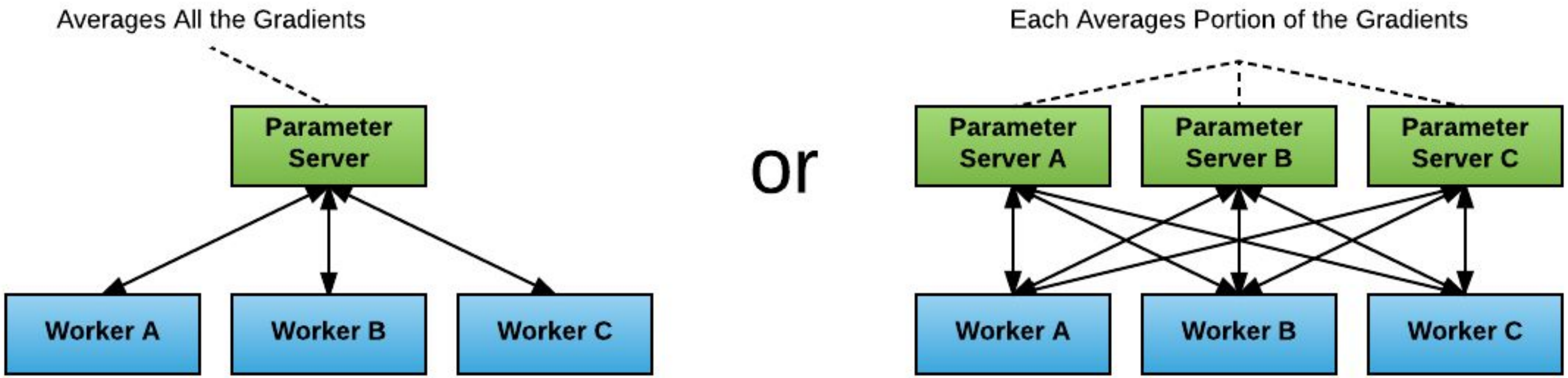




https://eng.uber.com/horovod/

https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/distribute

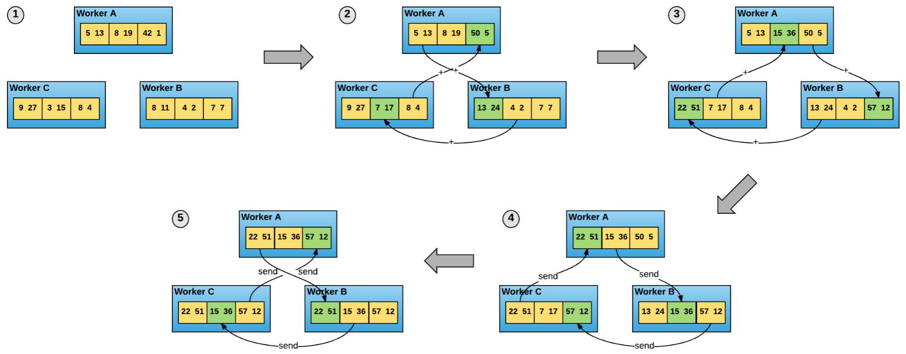
Challenge: Distributed TensorFlow



https://eng.uber.com/horovod/

Horovod

- [All-Reduce] to update Parameter
  - Bandwidth Optimal
- Uber Horovod is MPI based
  - Difficult to set up
  - Other Spark based implementations
- Wait for TensorFlow 2.0 ;)



https://eng.uber.com/horovod/

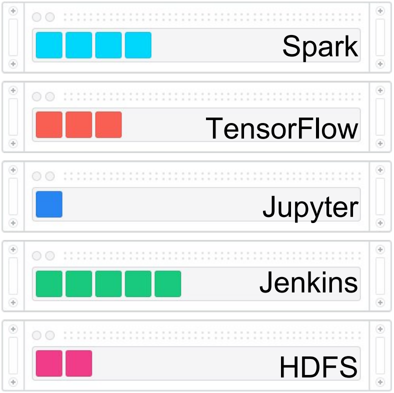
TF Distribution Strategy

- `MirroredStrategy`: This does in-graph replication with synchronous training on many GPUs on one machine. Essentially, we create copies of all variables in the model's layers on each device. We then use all-reduce to combine gradients across the devices before applying them to the variables to keep them in sync.
- `CollectiveAllReduceStrategy`: This is a version of `MirroredStrategy` for multi-working training. It uses a collective op to do all-reduce. This supports between-graph communication and synchronization, and delegates the specifics of the all-reduce implementation to the runtime (as opposed to encoding it in the graph). This allows it to perform optimizations like batching and switch between plugins that support different hardware or algorithms. In the future, this strategy will implement fault-tolerance to allow training to continue when there is worker failure.
- `ParameterServerStrategy`: This strategy supports using parameter servers either for multi-GPU local training or asynchronous multi-machine training. When used to train locally, variables are not mirrored, instead they placed on the CPU and operations are replicated across all local GPUs. In a multi-machine setting, some are designated as workers and some as parameter servers. Each variable is placed on one parameter server. Computation operations are replicated across all GPUs of the workers.

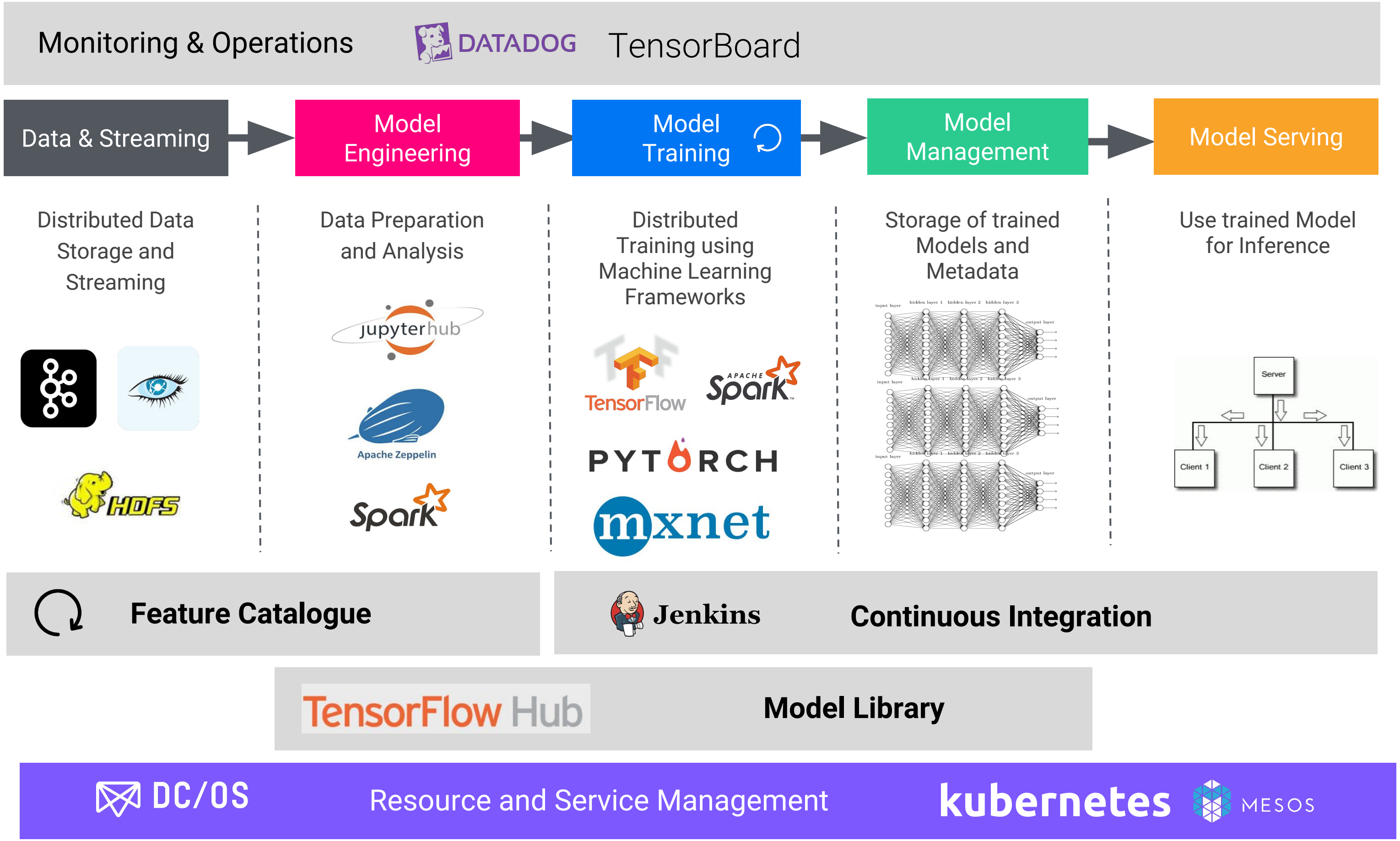
https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/distribute

Challenge: Resource and Service Management

- Different Distributed Systems
  - Deployment
  - Updates
  - Failure Recovery
  - Scaling
- Resource Efficiency
  - Multiple VM per Service?



Typical Datacenter
siloed, over-provisioned servers,
low utilization


Monitoring & Operations
DATADOG
TensorBoard
Data & Streaming
Model Engineering
Model Training
Model Management
Model Serving
Distributed Data Storage and Streaming
Data Preparation and Analysis
Distributed Training using Machine Learning Frameworks
Storage of trained Models and Metadata
Use trained Model for Inference
HDFS
jupyterhub
Apache Zeppelin
Spark
TensorFlow
APACHE Spark
PYTORCH
mxnet
Server
Client 1
Client 2
Client 3
Feature Catalogue
Jenkins
Continuous Integration
TensorFlow Hub
Model Library
DC/OS
Resource and Service Management
kubernetes
MESOS

@mesosphere

THANK YOU!

ANY QUESTIONS?

https://mesosphere.com/resources/building-data-science-platform/


Make it insanely easy
to build and scale
world-changing technology
MESOSPHERE
Swipe/Tap Your Card/Phone
VISA
BOSCH