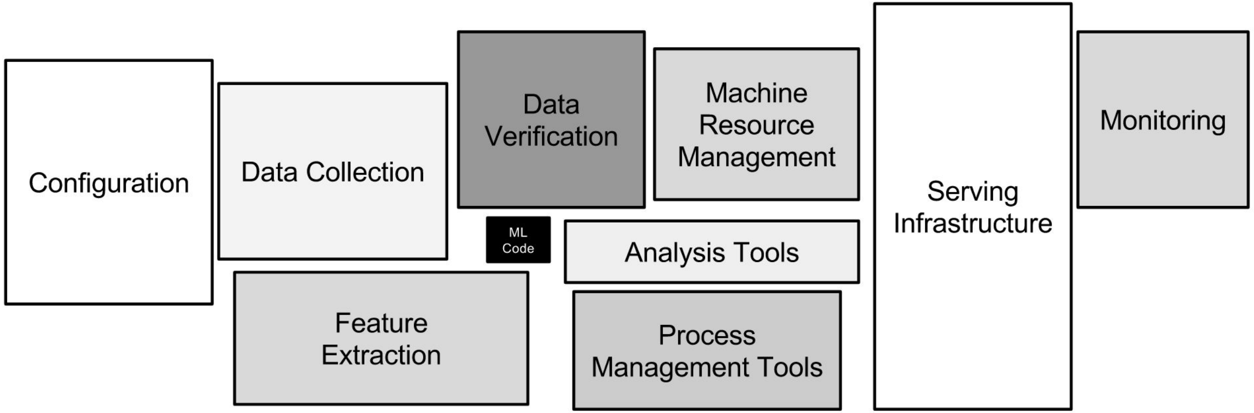
@joerg_schad #DataSciencePrinciples

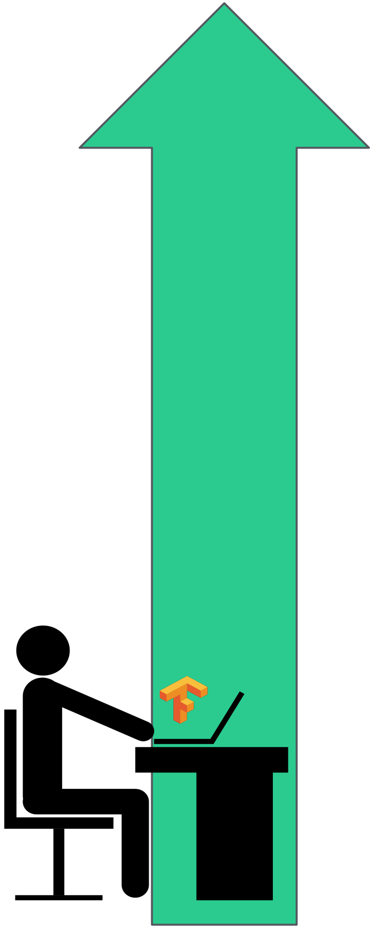
Building an Open Source Data Science Platform



https://goo.gl/ZnrR98

DC/OS

Agenda



Distributed TF, Horovod, Rendezvous Architecture, Serving

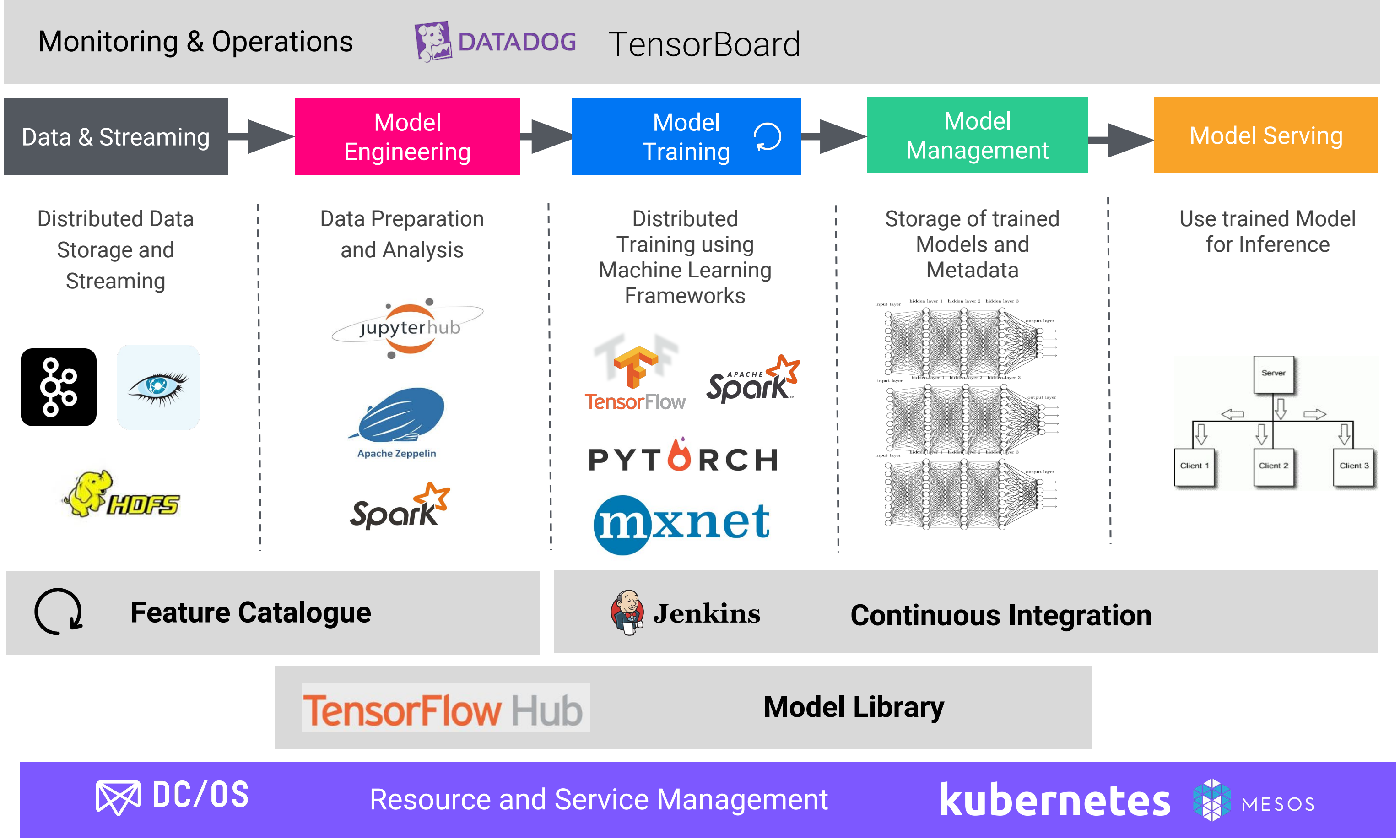
Model Optimization, Feature Store, Hyperparameter Opt

MLFlow, Jenkins

Spark, KubeFlow, TF Serving

TensorFlow, Jupyter

What is Machine Learning?


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

Challenge...

Sunday, October 21 • 13:30 - 17:15

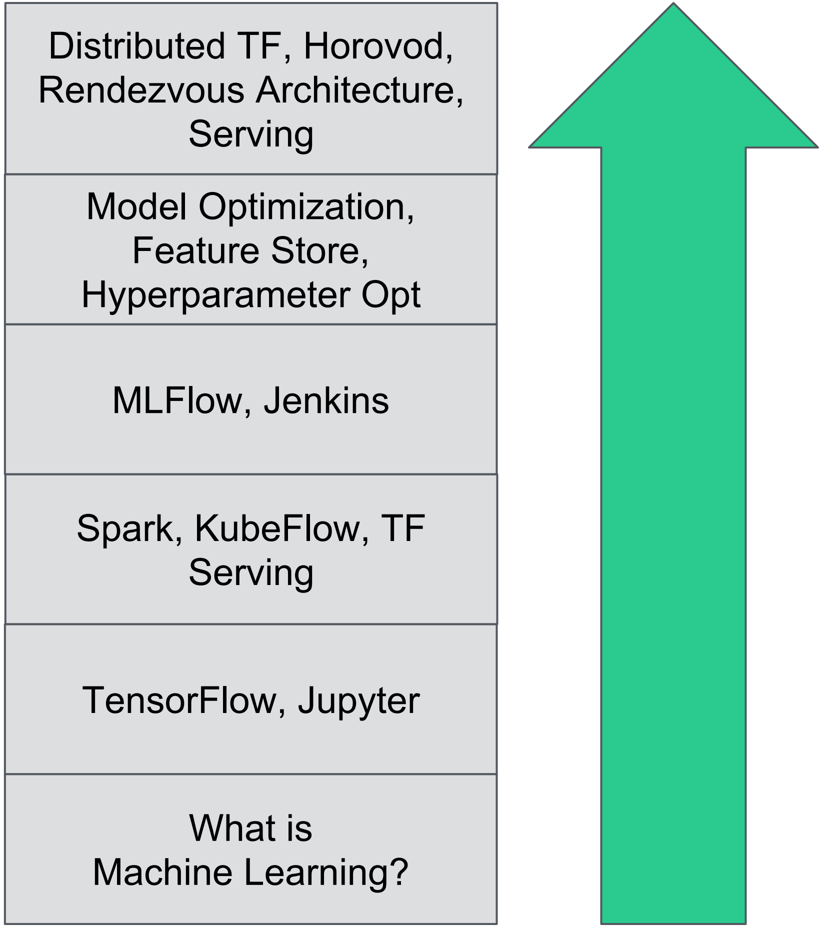
Workshop: Building and Operating an OSS Data Science Platform - Jörg Schad, Mesosphere

Click here to remove from My Sched.

Who are you?

What is your ML experience?

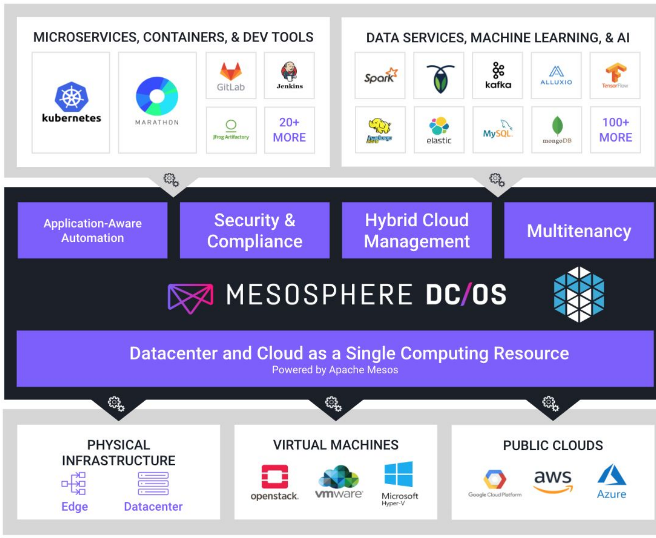
What do you expect to learn today?

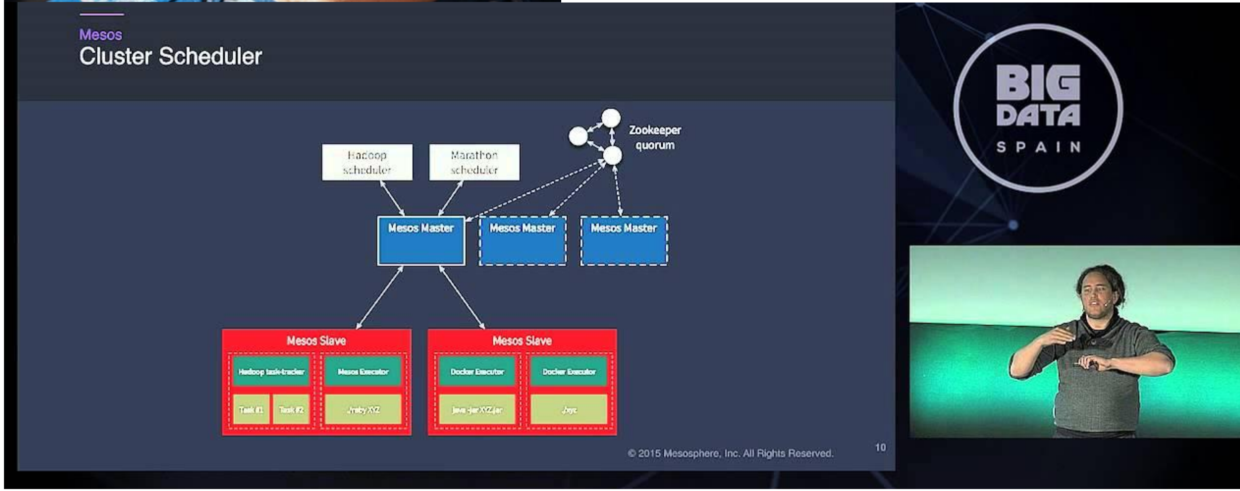


Jörg Schad

Technical Lead/Engineer
Deep Learning

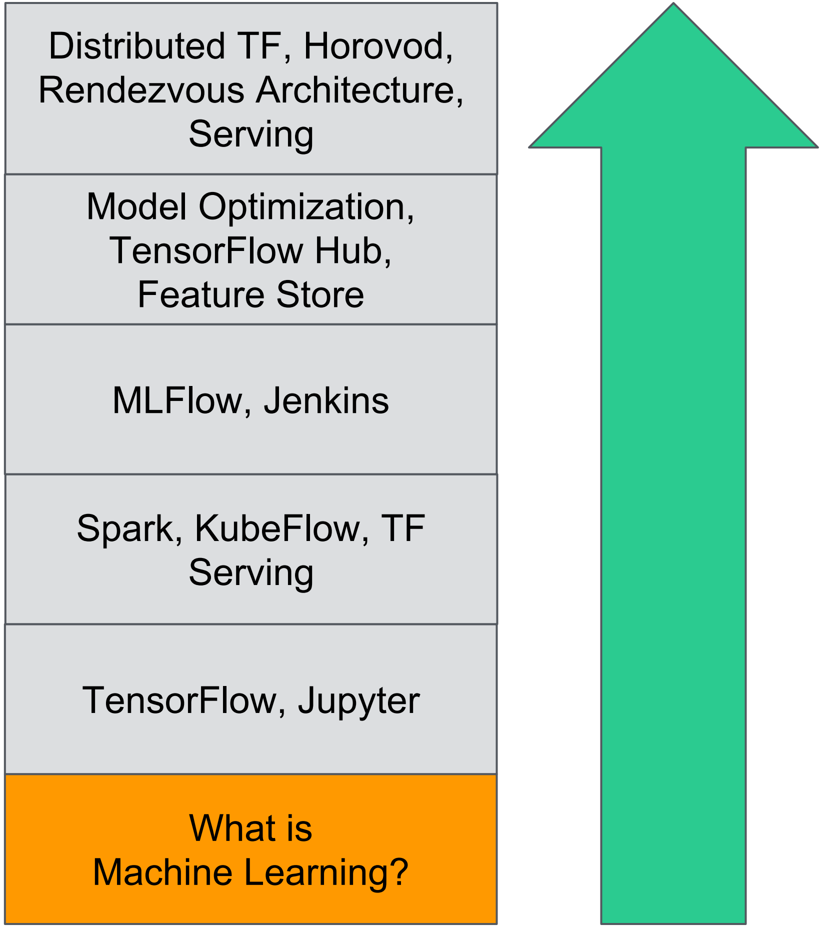
- Core Mesos developer at Mesosphere
- Twitter: @joerg_schad





Machine Learning

- Why do we care
- What is Machine Learning
- Data Science Principles
- Different Personas



Why is machine learning taking off?


BOSCH

AtomNet: A Deep Convolutional Neural Network for Bioactivity Prediction in Structure-based Drug Discovery

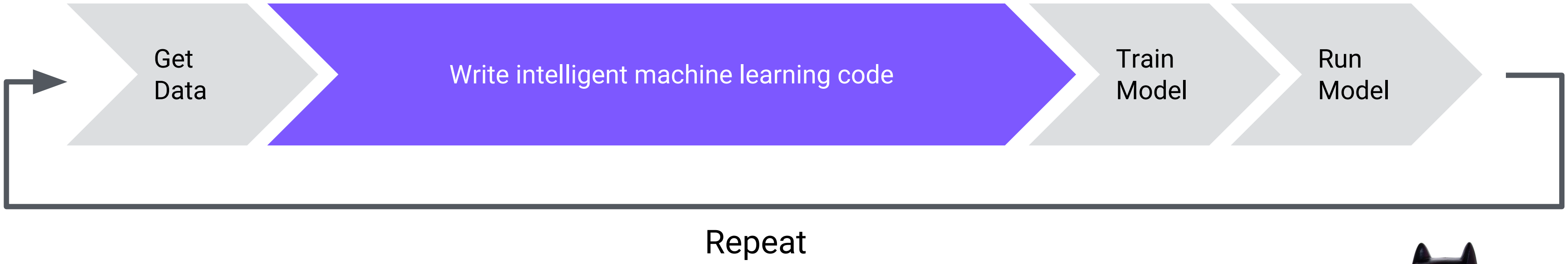
Izhar Wallach
Atomwise, Inc.
izhar@atomwise.com

Michael Dzamba
Atomwise, Inc.
misko@atomwise.com

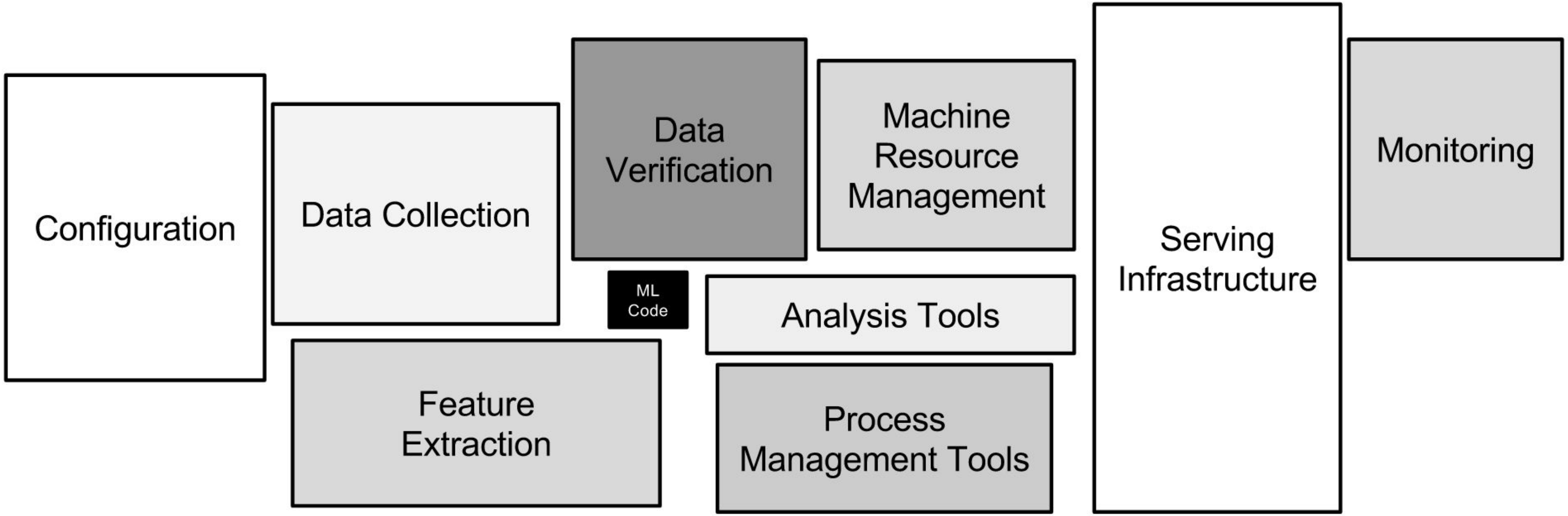
Abraham Heifets
Atomwise, Inc.
abe@atomwise.com

DEEPBACH: A STEERABLE MODEL FOR BACH CHORALES GENERATION

What you want to be doing



What you're actually doing



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

Data Science Principles

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

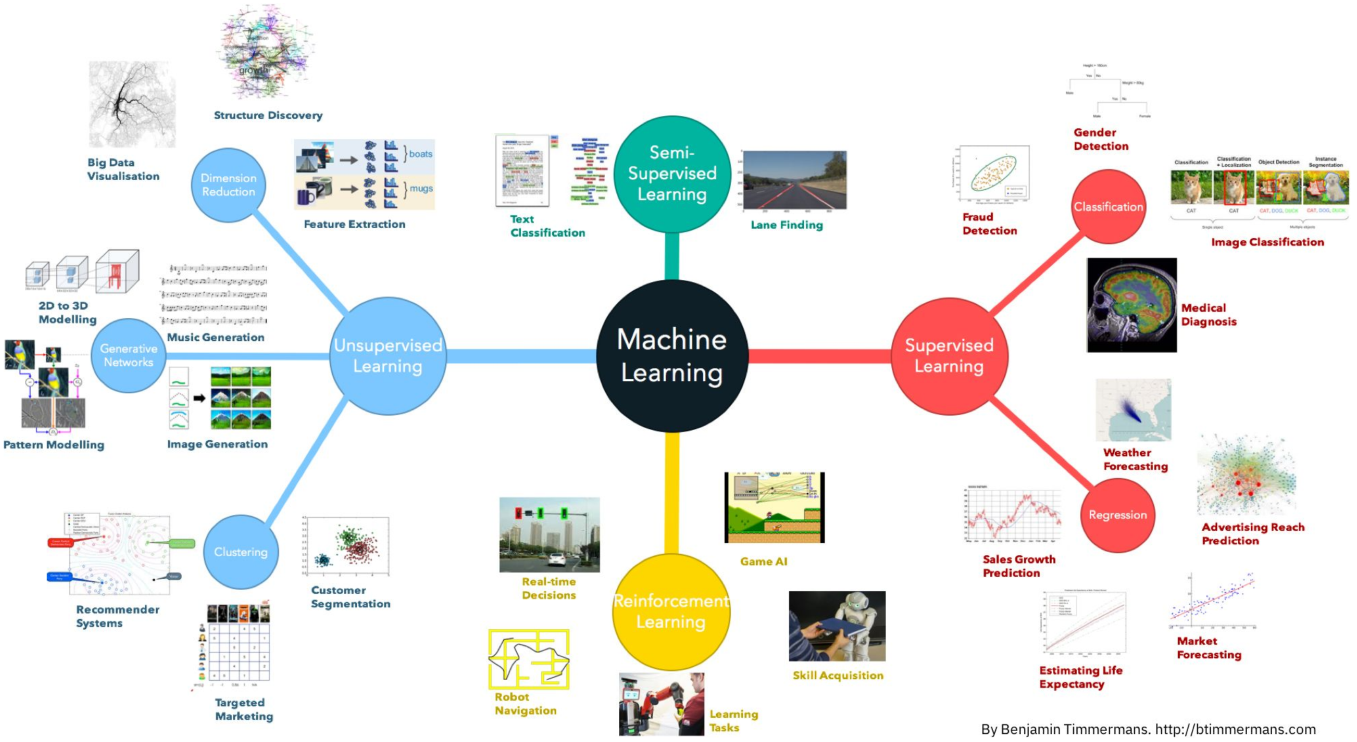
Challenge: Requirements Engineering

- 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?
- Required Model Freshness?

* Can I actually use ...

Machine Learning

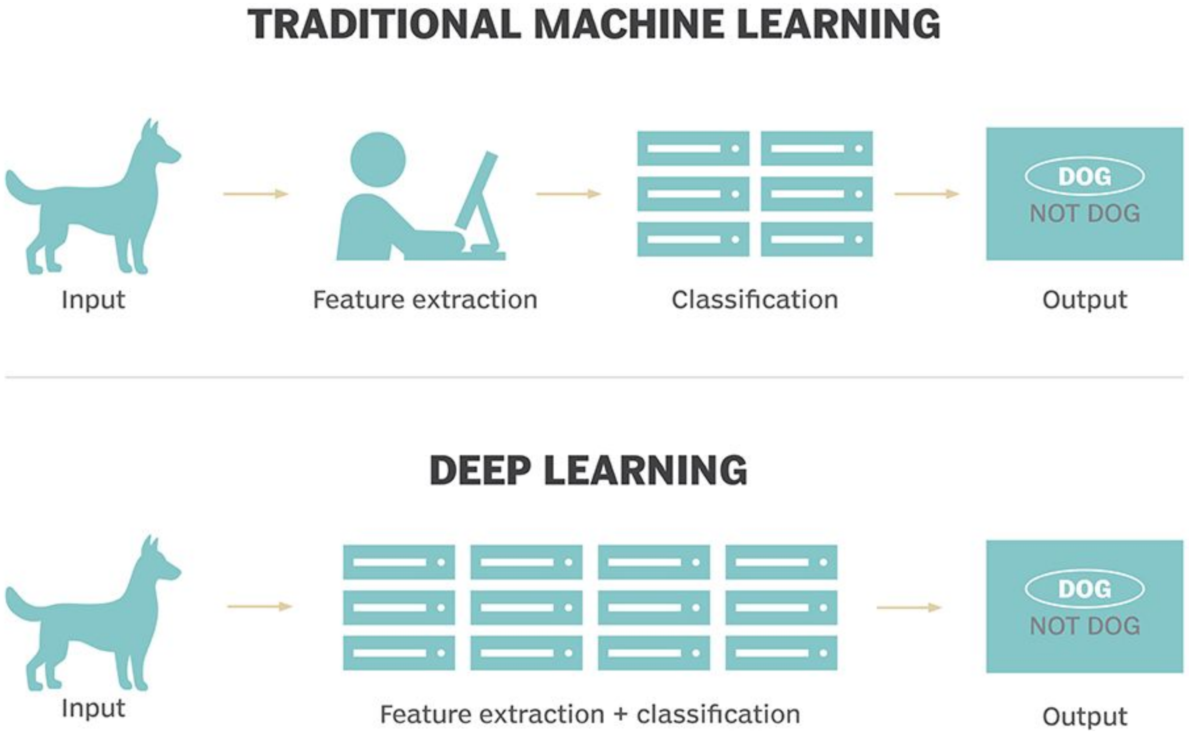


http://btimmermans.com/2017/12/11/machine-learning-overview/

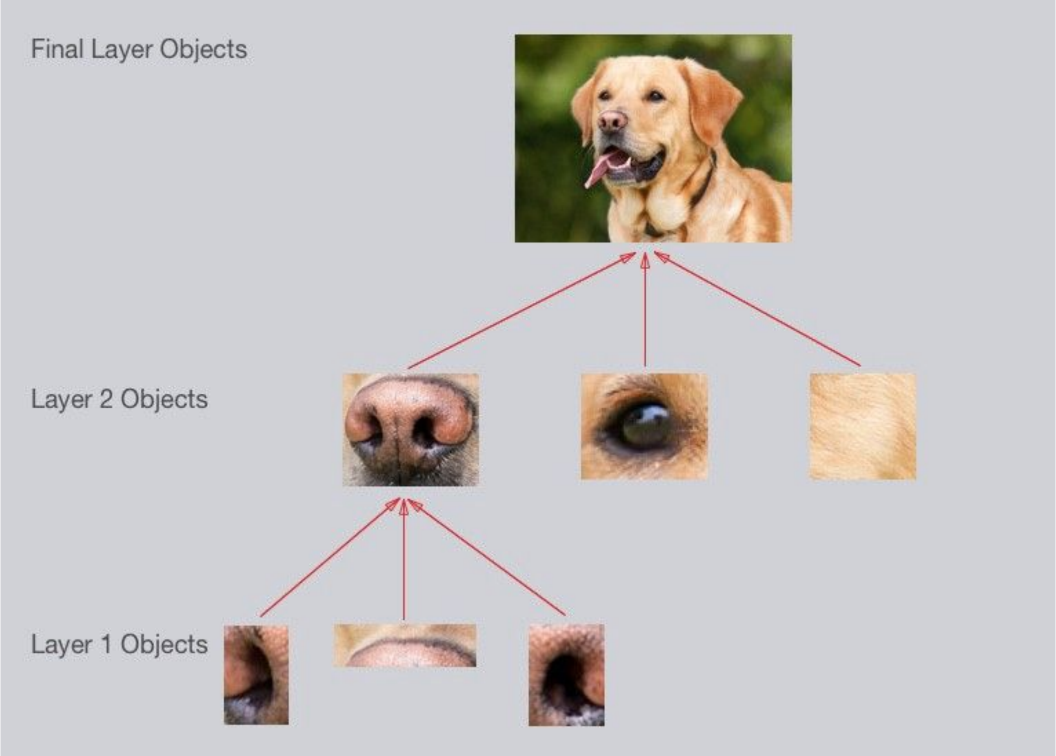
Fast.ai


False Positive Rate
False Negative Rate
66.3%
7.0%
47.5%
0.0%
Money
Could this computer save yo
Layer 5
Multi-label classification
haze primary
agriculture clear primary water
char 4 using chars 1, 2 & 3
Hidden
→Output
→Hidden
char 4 output: vocab size
FC3: # activations
char 3 input
FC2: # activations
char 2 input
FC1: # activations
char 1 input: vocab size

Deep Learning: The Promise



Deep Learning: Insights



https://arxiv.org/pdf/1506.06579.pdf

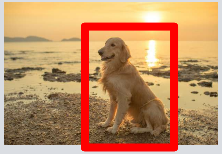
Deep Learning: The Process

Step 1: Training
(In Data Center - Over Hours/Days/Weeks)

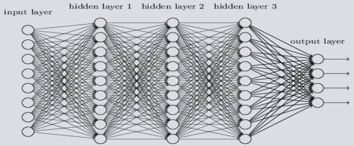
Input:
Lots of Labeled Data

Deep neural network model

Output:
Trained Model

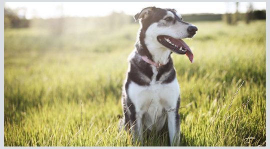


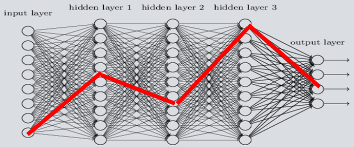
Dog

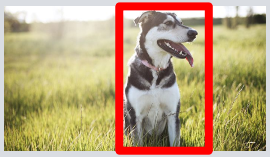


Trained Model

Step 2: Inference
(Endpoint or Data Center - Instantaneous)







97% Dog
3%
Panda

New Input from Camera or Sensor

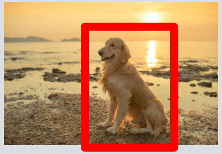
Trained Model

Output:
Classification

Deep Learning: The Process

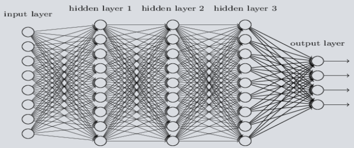
Step 1: Training
(In Data Center - Over Hours/Days/Weeks)

Input:
Lots of Labeled
Data



Dog

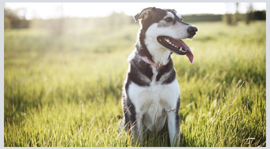
Deep neural
network model



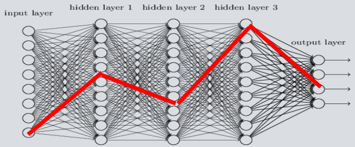
Output:
Trained Model

Trained Model

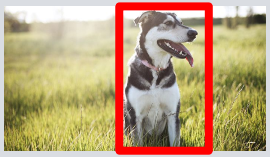
Step 2: Inference
(Endpoint or Data Center - Instantaneous)



New Input from
Camera or
Sensor



Trained Model



97% Dog
3%
Panda

Output:
Classification

Challenge: Persona(s)

The Rise of the *DataOps Engineer*

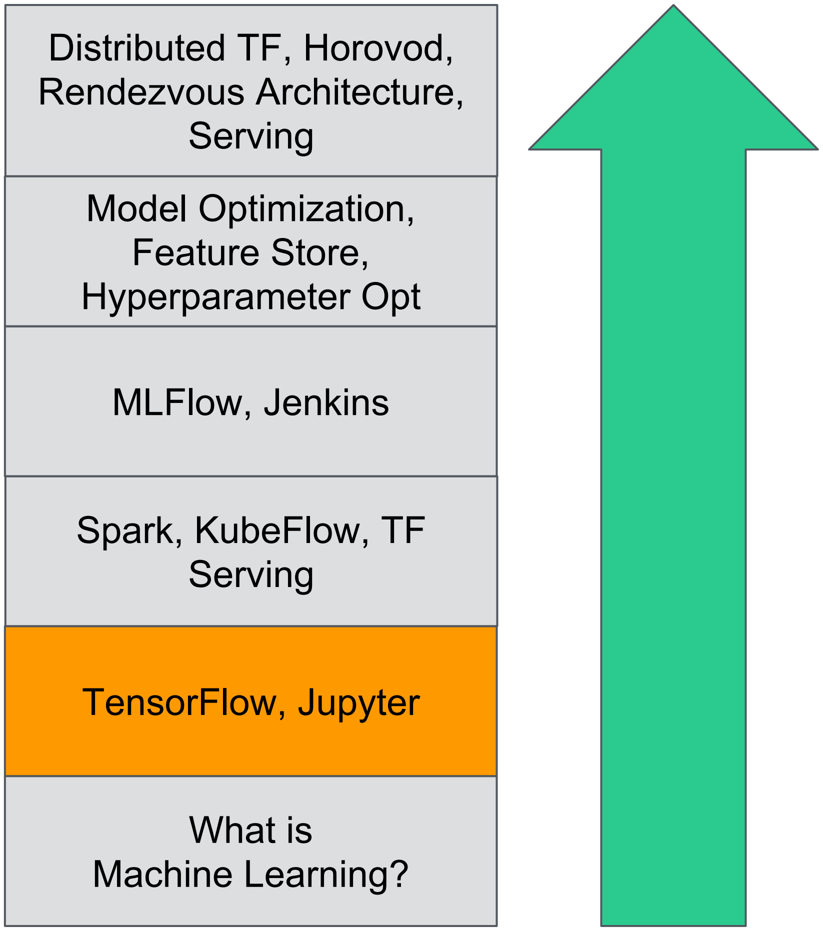
Combines two key skills:

- Data science
- Distributed systems engineering

The equivalent of *DevOps* for *Data Science*

TensorFlow & Jupyter

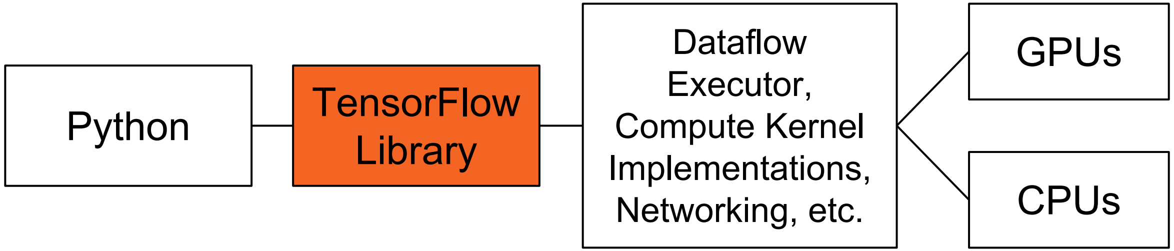
- First Hands-On Machine Learning
- **Open Source Technologies**
  - TensorFlow
  - Jupyter
- **Labs**
  - Mnist with Google Colab
  - Deploy and use Jupyter

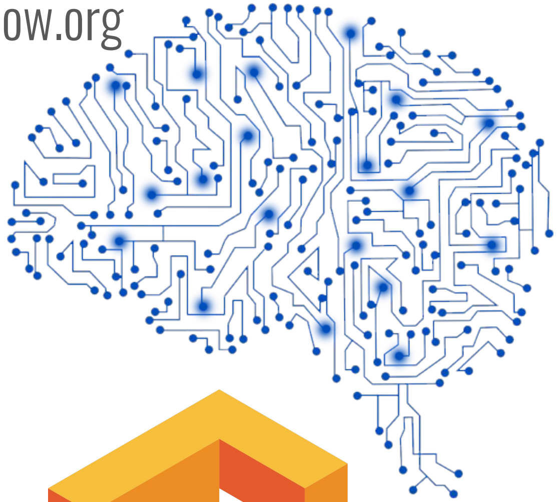


TensorFlow Overview

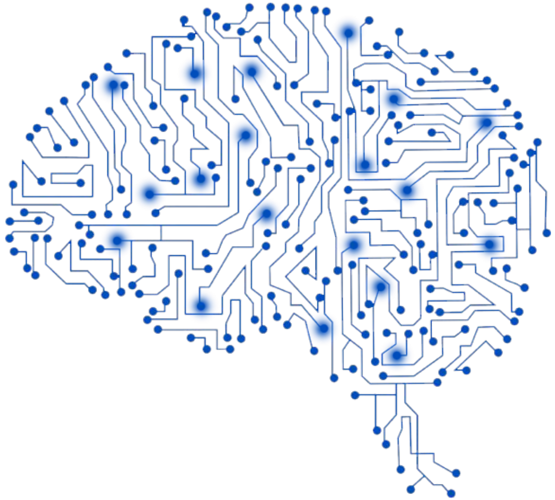
"An open-source **software library** for Machine Intelligence" - tensorflow.org

- Tensorflow is a **software library** that makes it easy for developers to construct artificial neural networks to analyze their data of interest





ML Frameworks Overview



Alternatives

```
import tensorflow as tf
import numpy as np
X = tf.placeholder("float")
Y = tf.placeholder("float")
W = tf.Variable(np.random.random(), name="weight")
pred = tf.multiply(X, W)
cost = tf.reduce_sum(tf.pow(pred-Y, 2))
optimizer = tf.train.GradientDescentOptimizer(0.01).minimize(cost)
init = tf.global_variables_initializer()
with tf.Session() as sess:
    sess.run(init)
    for t in range(10000):
        x = np.array(np.random.random()).reshape((1, 1, 1, 1))
        y = x * 3
        (_, c) = sess.run([optimizer, cost], feed_dict={X: x, Y: y})
        print c
```

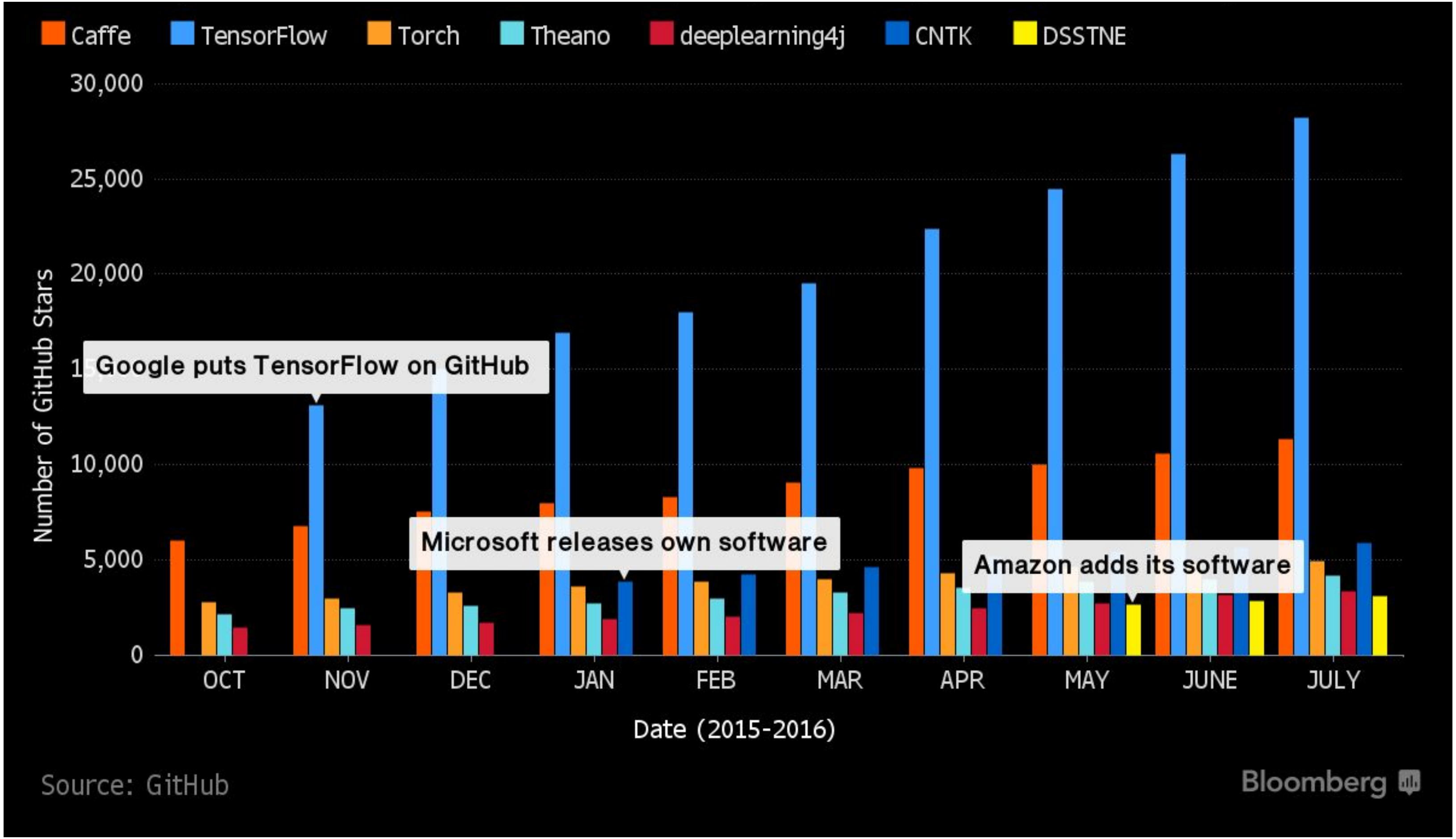
```
import numpy as np
import torch
from torch.autograd import Variable
model = torch.nn.Linear(1, 1)
loss_fn = torch.nn.MSELoss(size_average=False)
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)
for t in range(10000):
    x = Variable(torch.from_numpy(np.random.random((1,1)).astype(np.float32)))
    y = x * 3
    y_pred = model(x)
    loss = loss_fn(y_pred, y)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    print loss.data[0]
```

Alternatives

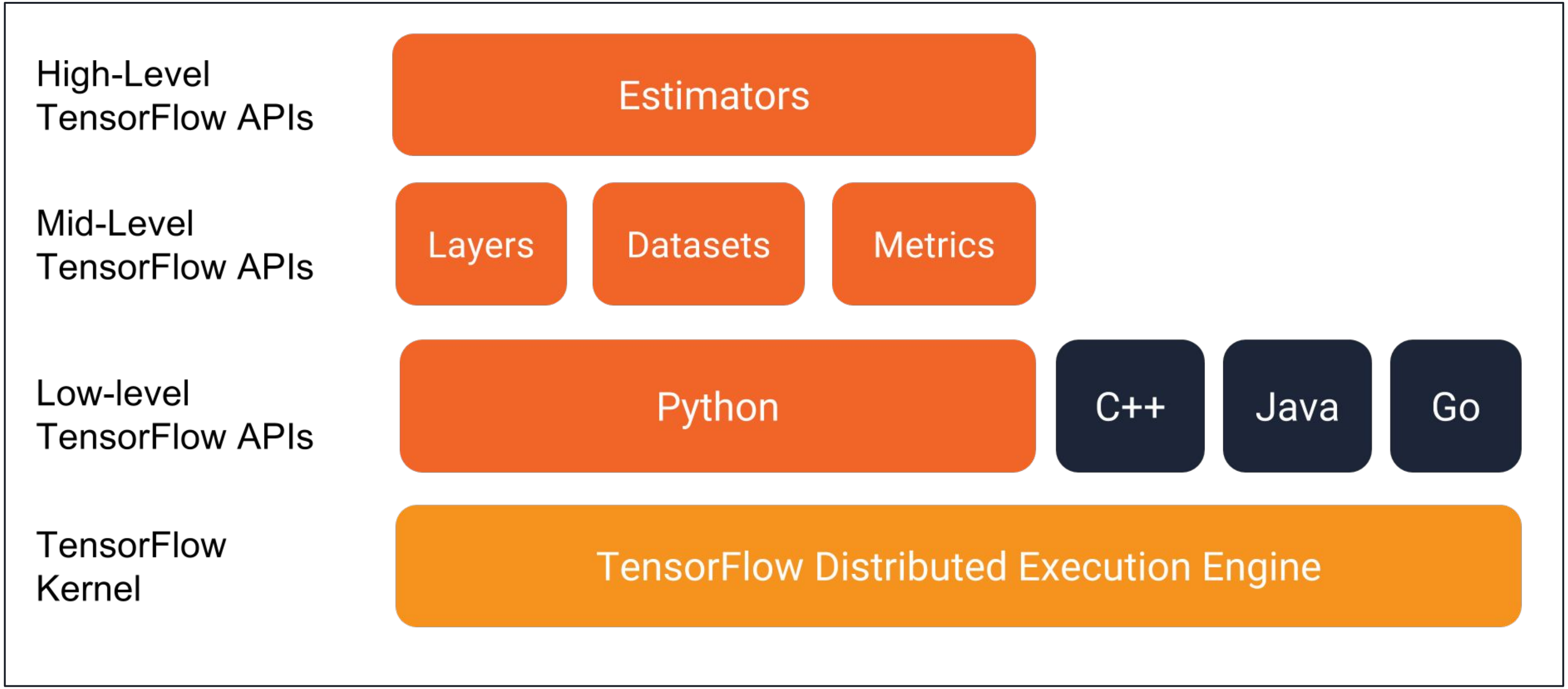
```python
import tensorflow as tf
import numpy as np
X = tf.placeholder("float")
Y = tf.placeholder("float")
W = tf.Variable(np.random.random(), name="weight")
pred = tf.multiply(X, W)
cost = tf.reduce_sum(tf.pow(pred-Y, 2))
optimizer = tf.train.GradientDescentOptimizer
init = tf.global_variables_initializer()
with tf.Session() as sess:
    sess.run(init)
    for t in range(10000):
        x = np.array(np.random.random()).reshape((1, 1, 1, 1))
        y = x * 3
        (_, c) = sess.run([optimizer, cost], feed_dict={X: x, Y: y})
        print c
```

```python
tf.enable_eager_execution()
```

```python
import numpy as np
import torch
from torch.autograd import Variable
model = torch.nn.Linear(1, 1)
loss_fn = torch.nn.MSELoss(size_average=False)
ptim.SGD(model.parameters(), lr=0.01)
0):
rch.from_numpy(np.random.random((1,1)).astype(np.float32)))

x)
    loss = loss_fn(y_pred, y)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    print loss.data[0]
```


Caffe
TensorFlow
Torch
Theano
deeplearning4j
CNTK
DSSTNE
30,000
25,000
20,000
10,000
5,000
0
Number of GitHub Stars
Google puts TensorFlow on GitHub
Microsoft releases own software
Amazon adds its software
OCT
NOV
DEC
JAN
FEB
MAR
APR
MAY
JUNE
JULY
Date (2015-2016)
Source: GitHub
Bloomberg

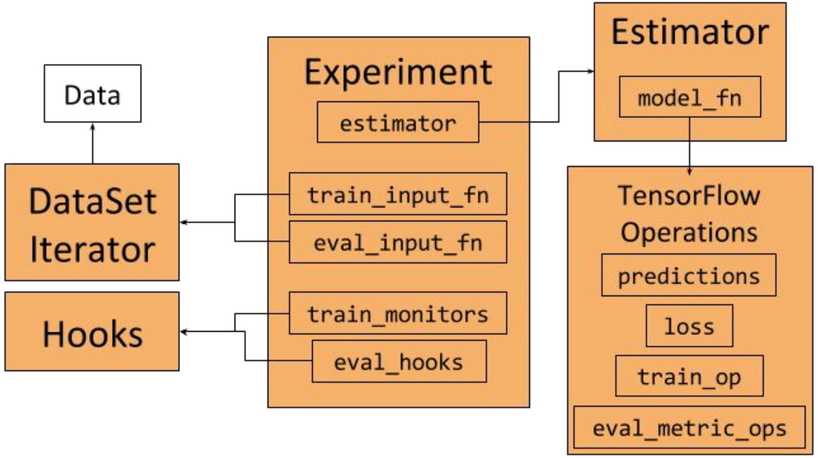
Challenge: Writing Distributed Model Functions



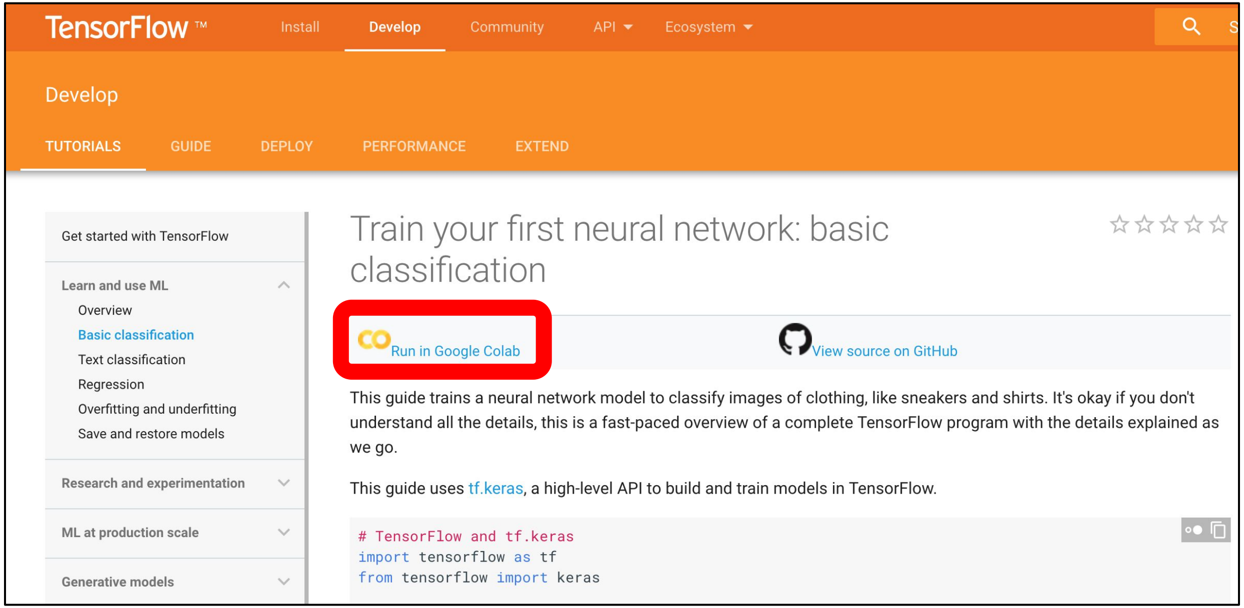
TensorFlow Estimator & Keras APIs

- Prefered APIs

```
return tf.estimator.Estimator(
    model_fn=model_fn,  # First-class function
    params=params,  # HParams
    config=run_config  # RunConfig
)
```



Lab 0: Fashion Mnist



https://www.tensorflow.org/tutorials/keras/basic_classification

Lab 1: JupyterLab

DataOps: Setup Environment

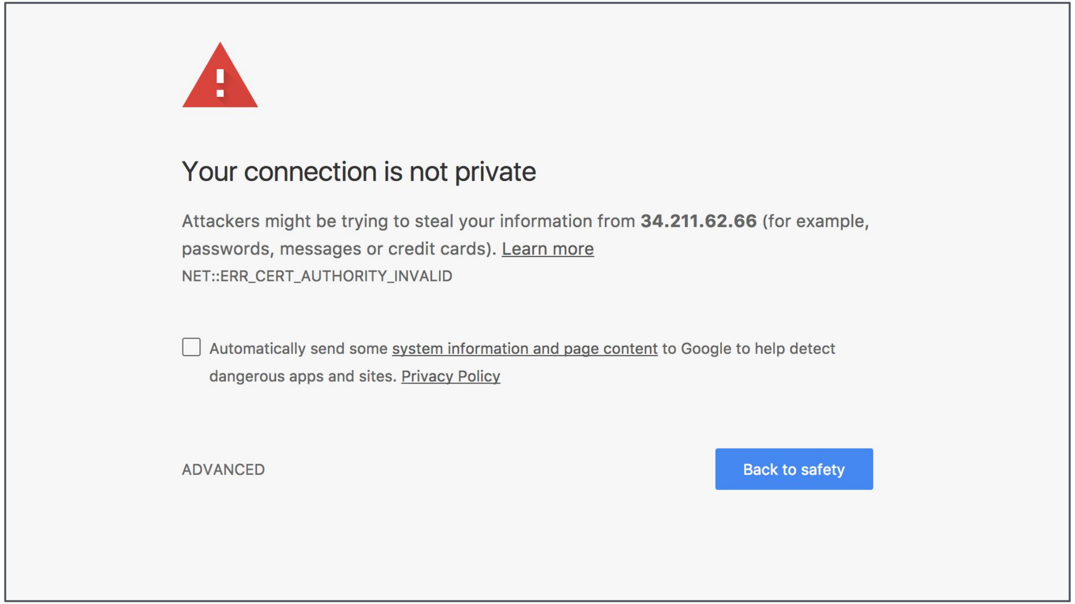
Data Scientist: Use distributed resources

1. Install HDFS
2. Install Marathon-LB (Proxy)
3. Install Jupyter

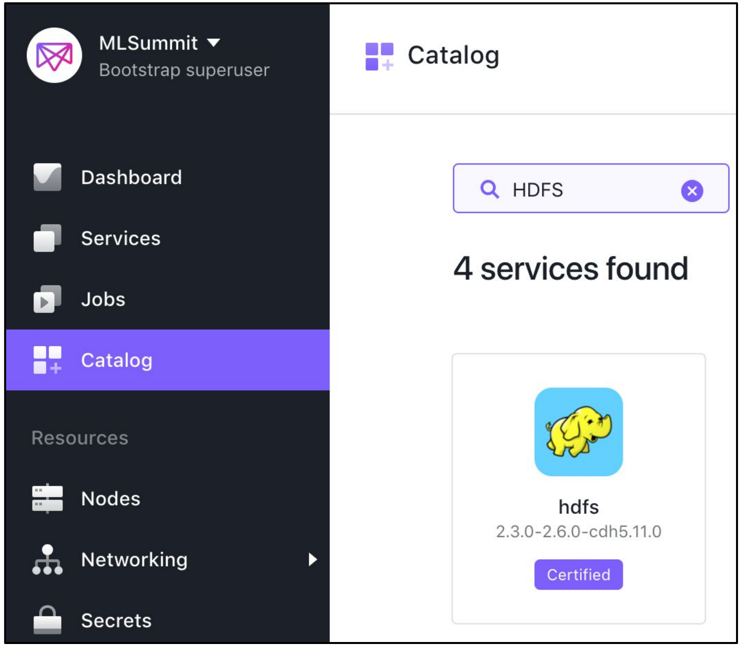
https://github.com/dcos/demos/tree/master/jupyterlab/1.11

Lab 1: Connect to Cluster

USER: bootstrapuser
Password: deleteme

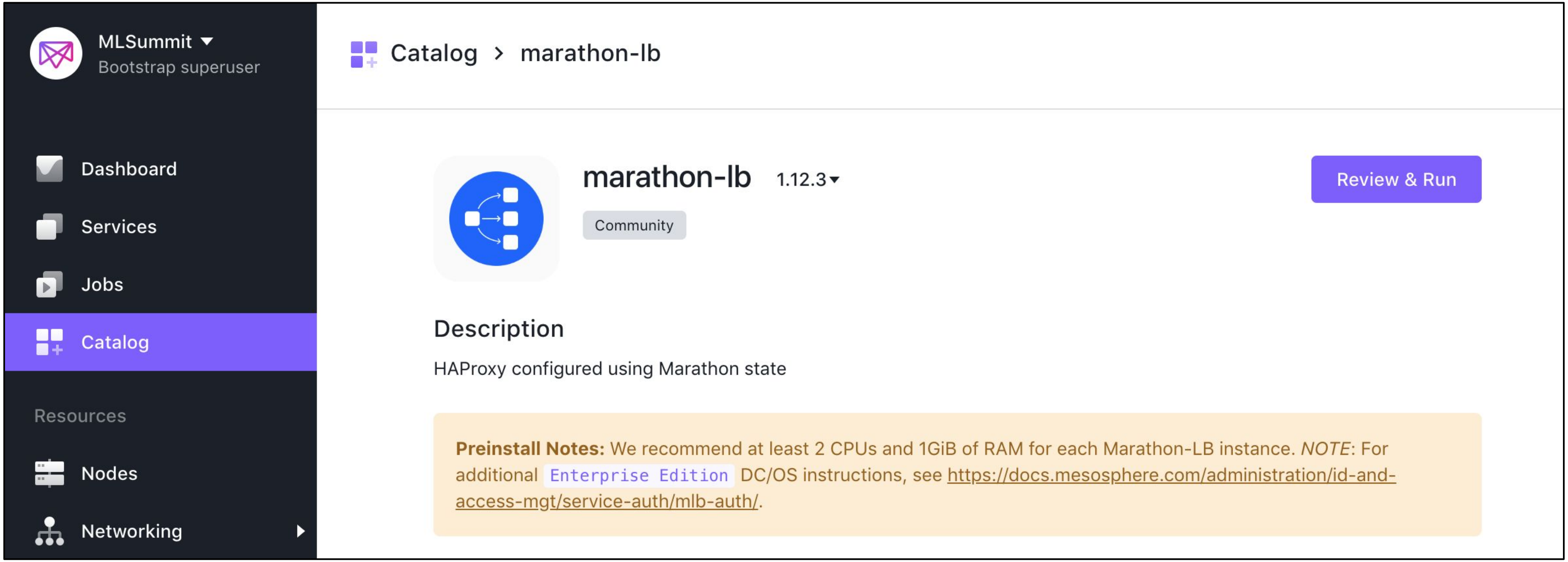


Lab 1: HDFS



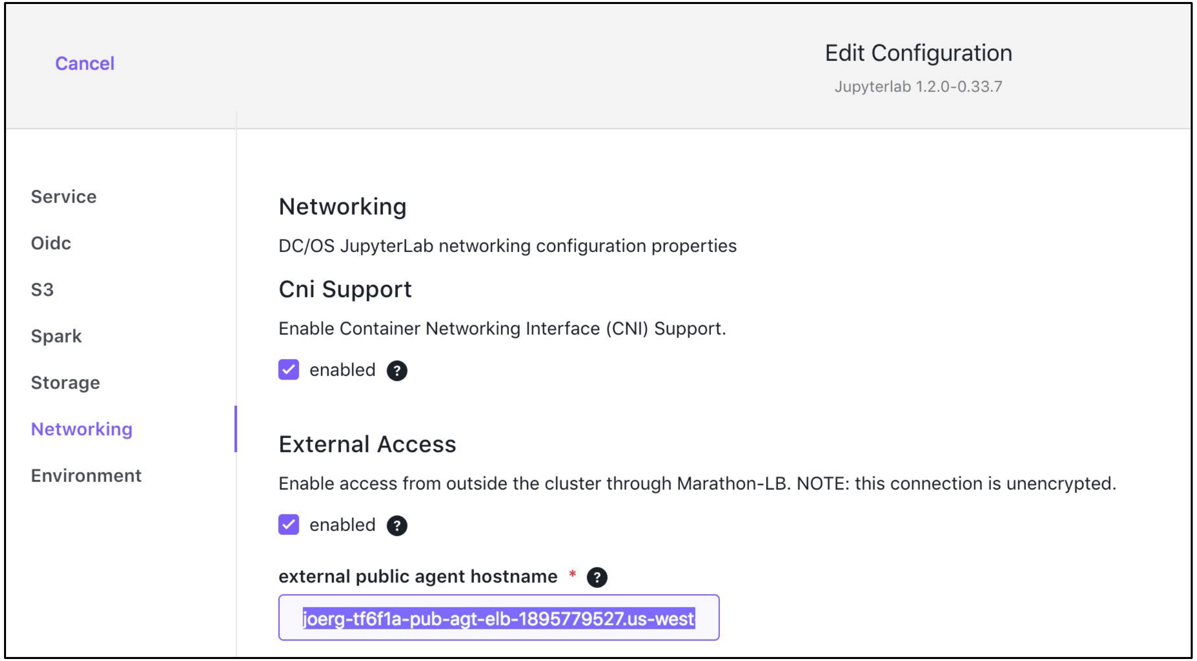
https://github.com/dcos/demos/tree/master/jupyterlab/1.11

Lab 1: Marathon-LB



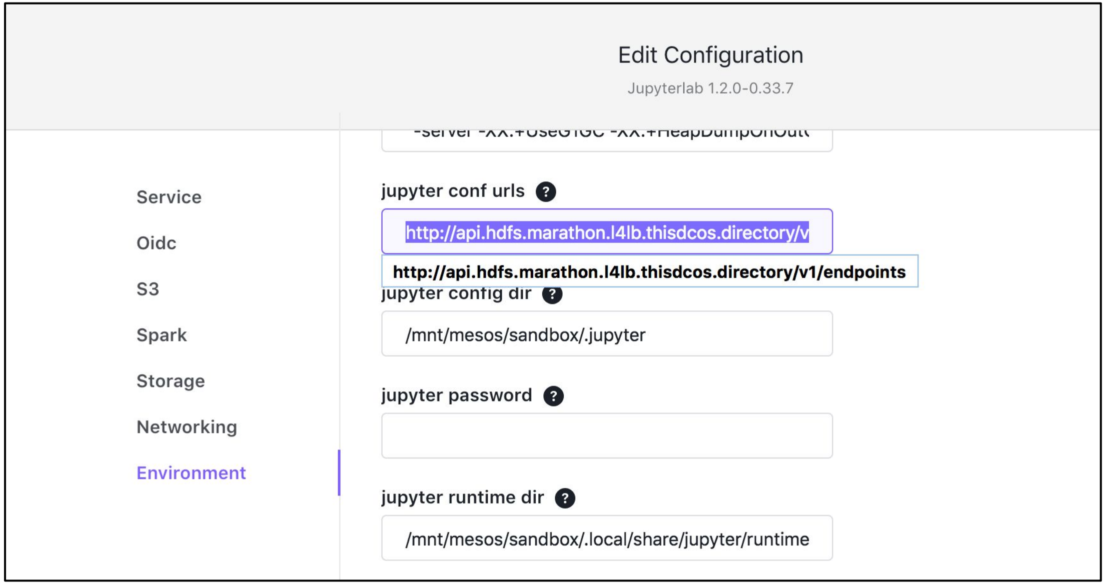
https://github.com/dcos/demos/tree/master/jupyterlab/1.11

Lab 1: Jupyter



https://github.com/dcos/demos/tree/master/jupyterlab/1.11

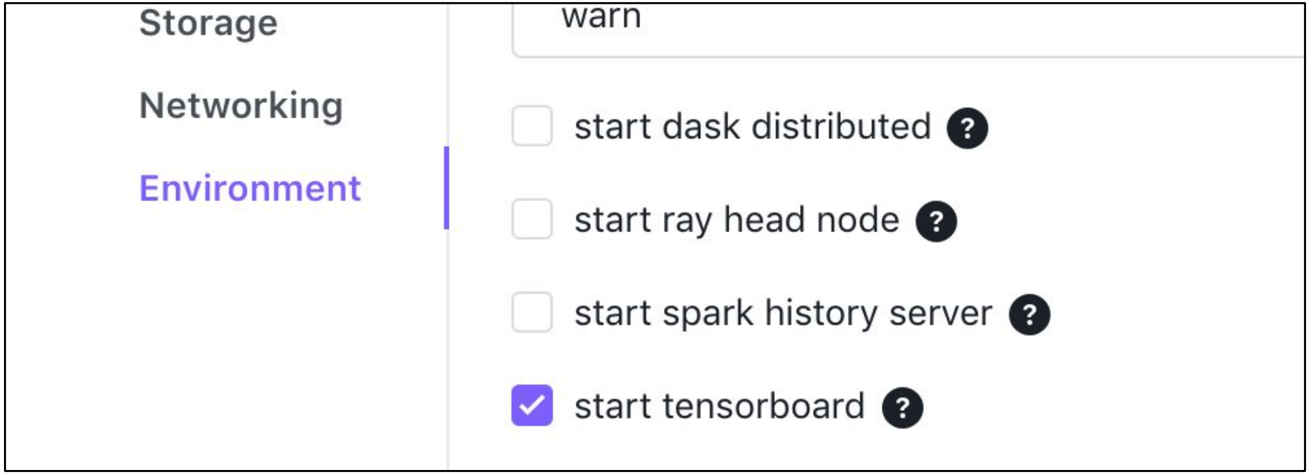
Lab 1: Jupyter



`http://api.hdfs.marathon.l4lb.thisdcos.directory/v1/endpoints`

https://github.com/dcos/demos/tree/master/jupyterlab/1.11

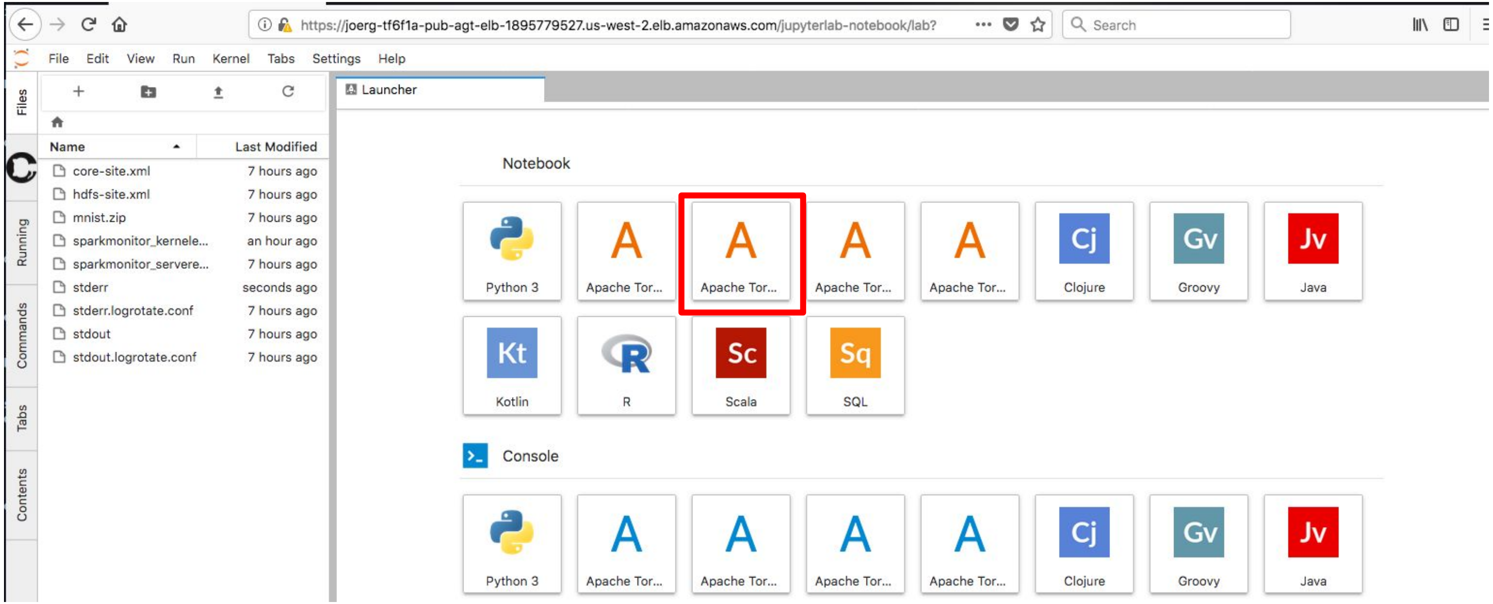
Lab 1: Jupyter



https://github.com/dcos/demos/tree/master/jupyterlab/1.11

<External>/jupyterlab-notebook
PW: jupyter

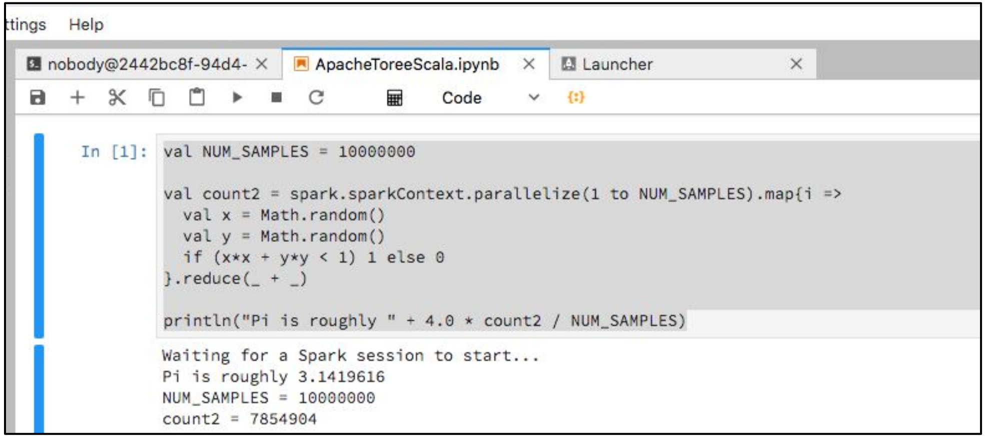
Lab 1: Jupyter



https://github.com/dcos/demos/tree/master/jupyterlab/1.11

Lab 1: Jupyter

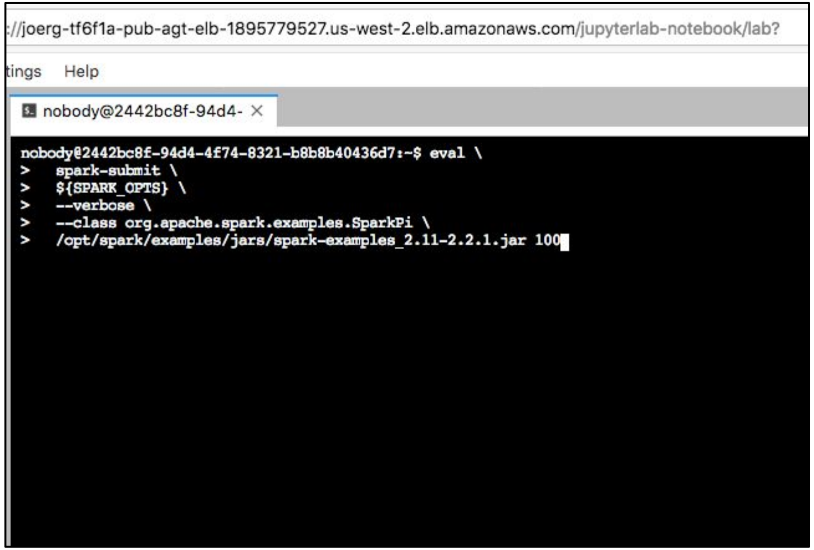
```
val NUM_SAMPLES = 10000000

val count2 =
spark.sparkContext.parallelize(1 to
NUM_SAMPLES).map{i =>
  val x = Math.random()
  val y = Math.random()
  if (x*x + y*y < 1) 1 else 0
}.reduce(_ + _)

println("Pi is roughly " + 4.0 *
count2 / NUM_SAMPLES)
```



https://github.com/dcos/demos/tree/master/jupyterlab/1.11

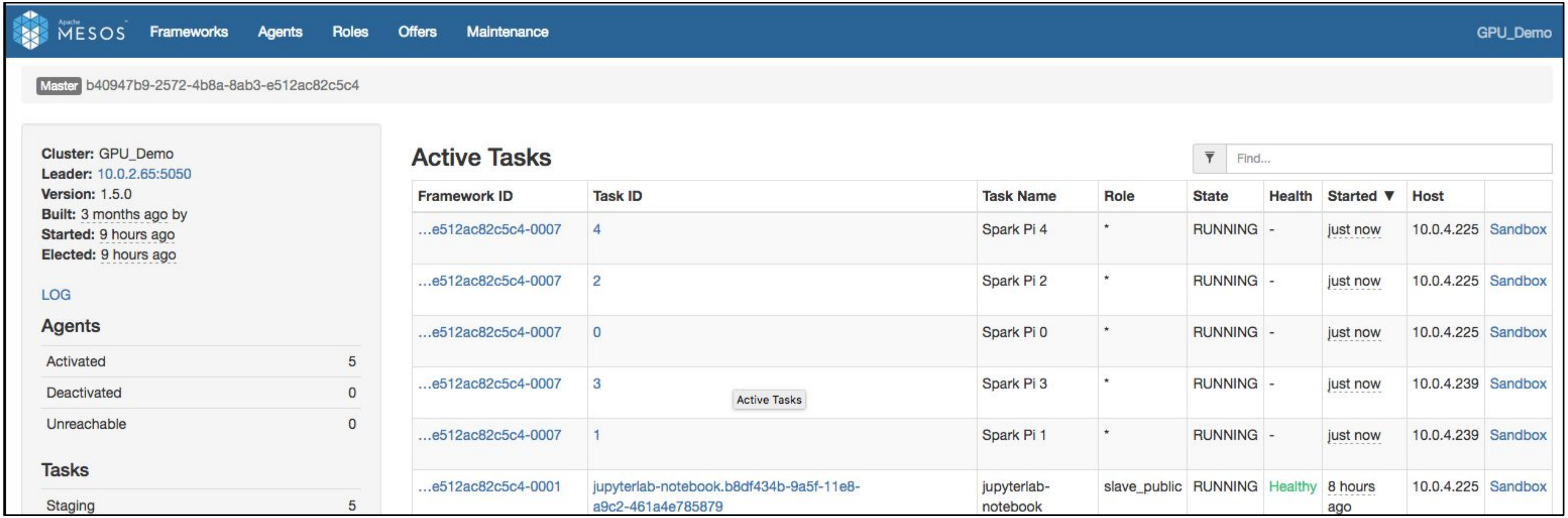
Lab 1: Jupyter

```
eval \
  spark-submit \
  ${SPARK_OPTS} \
  --verbose \
  --class
org.apache.spark.examples.SparkPi \

/opt/spark/examples/jars/spark-example
s_2.11-2.2.1.jar 100
```



https://github.com/dcos/demos/tree/master/jupyterlab/1.11

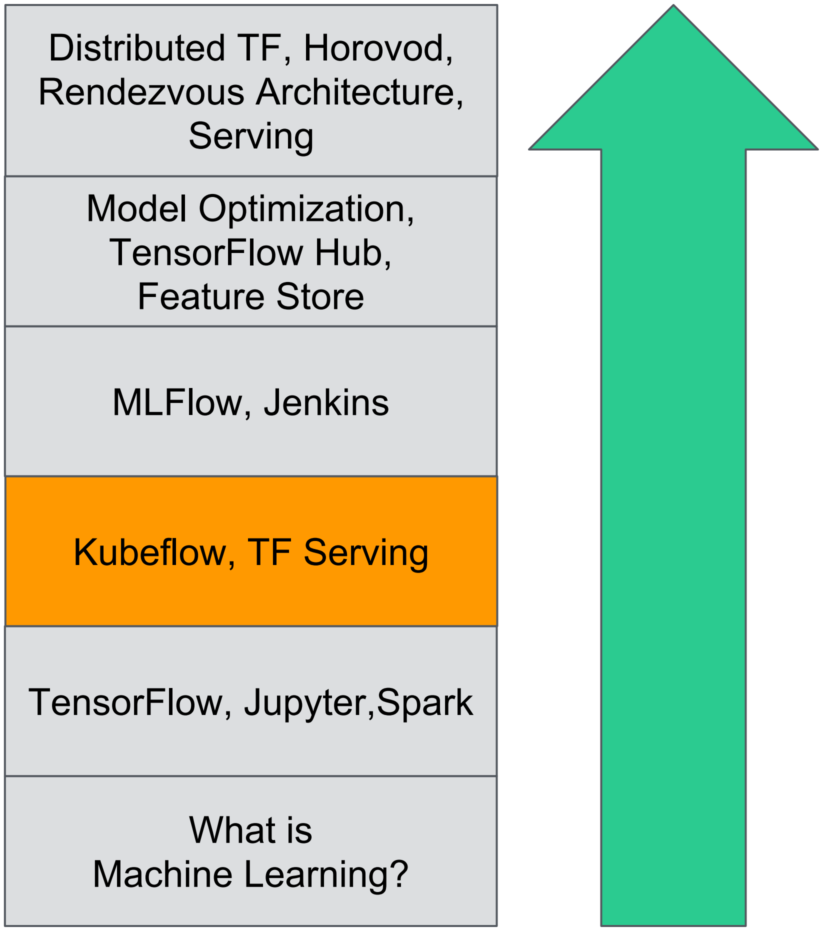
Lab 1: Jupyter



https://github.com/dcos/demos/tree/master/jupyterlab/1.11

First Pipeline

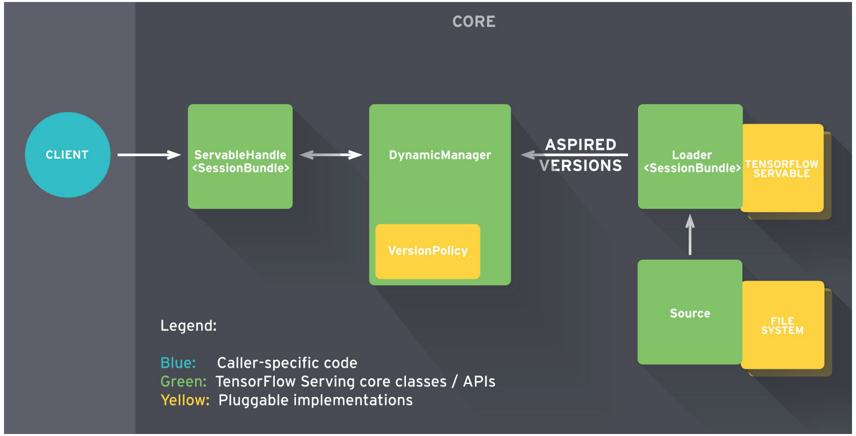
- Pipeline overview
    - KubeFlow
    - TFX
    - Michelangelo
- **Open Source Technologies**
    - Kubeflow
    - TF Serving
- **Labs**
    - [opt] Serving
    - [opt] KubeFlow



Challenge: Serving

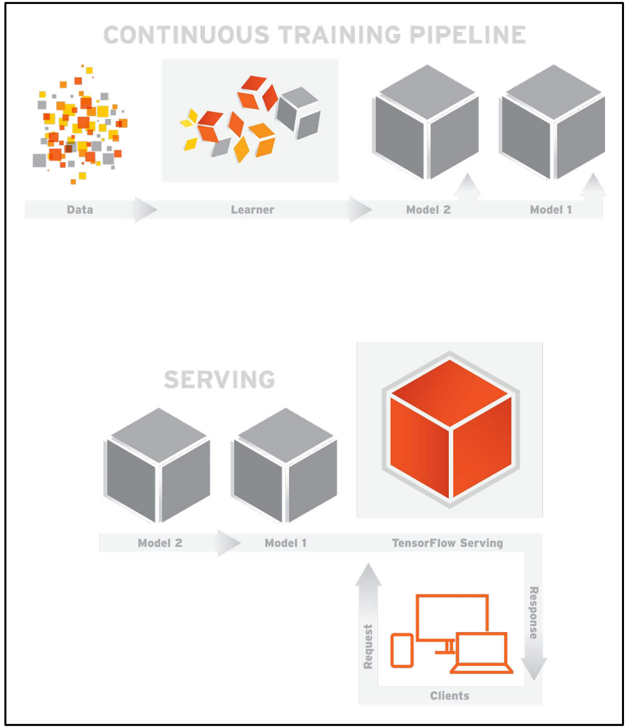
- How to Deploy Models?
  - Zero Downtime
  - Canary
- Multiple Models?
  - Testing

- TensorFlow Serving



Challenge: Serving Environment

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



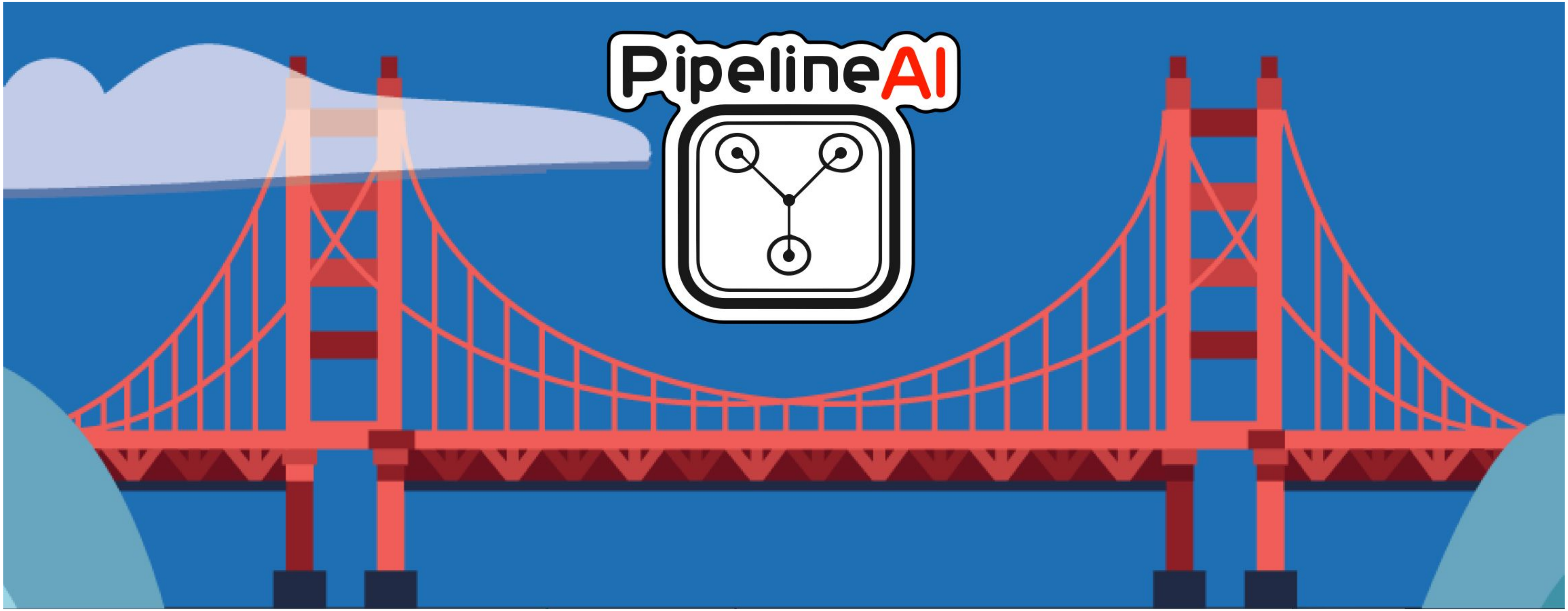
https://ai.googleblog.com/2016/02/running-your-models-in-production-with.html

Lab: (Optional) TensorFlow Serving

```
# Download the TensorFlow Serving Docker image and repo

docker pull tensorflow/serving

git clone https://github.com/tensorflow/serving

# Location of demo models

TESTDATA="$(pwd)/serving/tensorflow_serving/servables/tensorflow/testdata"

# Start TensorFlow Serving container and open the REST API port

docker run -t --rm -p 8501:8501 \

   -v "$TESTDATA/saved_model_half_plus_two_cpu:/models/half_plus_two" \

   -e MODEL_NAME=half_plus_two \

   tensorflow/serving &

# Query the model using the predict API

curl -d '{"instances": [1.0, 2.0, 5.0]}' \

   -X POST http://localhost:8501/v1/models/half_plus_two:predict

# Returns => { "predictions": [2.5, 3.0, 4.5] }
```

https://www.tensorflow.org/serving/serving_basic
https://www.tensorflow.org/serving/serving_advanced

Pipeline.ai



https://pipeline.ai/

Pipeline.ai

PipelineAI   Train   Evaluate   Compare

```
@log(labels=_labels, logger=_logger)
def predict(request: bytes) -> bytes:
```

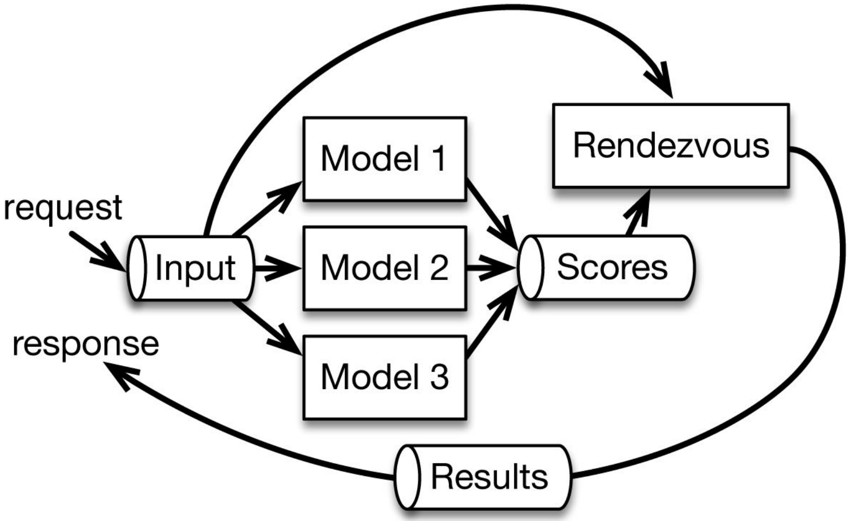
Digit	Confidence
0	0.0022526539396494627
1	2.63791100074684e-10
2	0.4638307988643646
3	0.21909376978874207
4	3.2985670372909226e-07
5	0.29357224702835083
6	0.00019597385835368186
7	5.230629176367074e-05
8	0.020996594801545143
9	5.426473762781825e-06

Digit	Confidence
0	0.0022526539396494627
1	2.63791100074684e-10
2	0.4638307988643646
3	0.21909376978874207
4	3.2985670372909226e-07
5	0.29357224702835083
6	0.00019597385835368186
7	5.230629176367074e-05
8	0.020996594801545143
9	5.426473762781825e-06

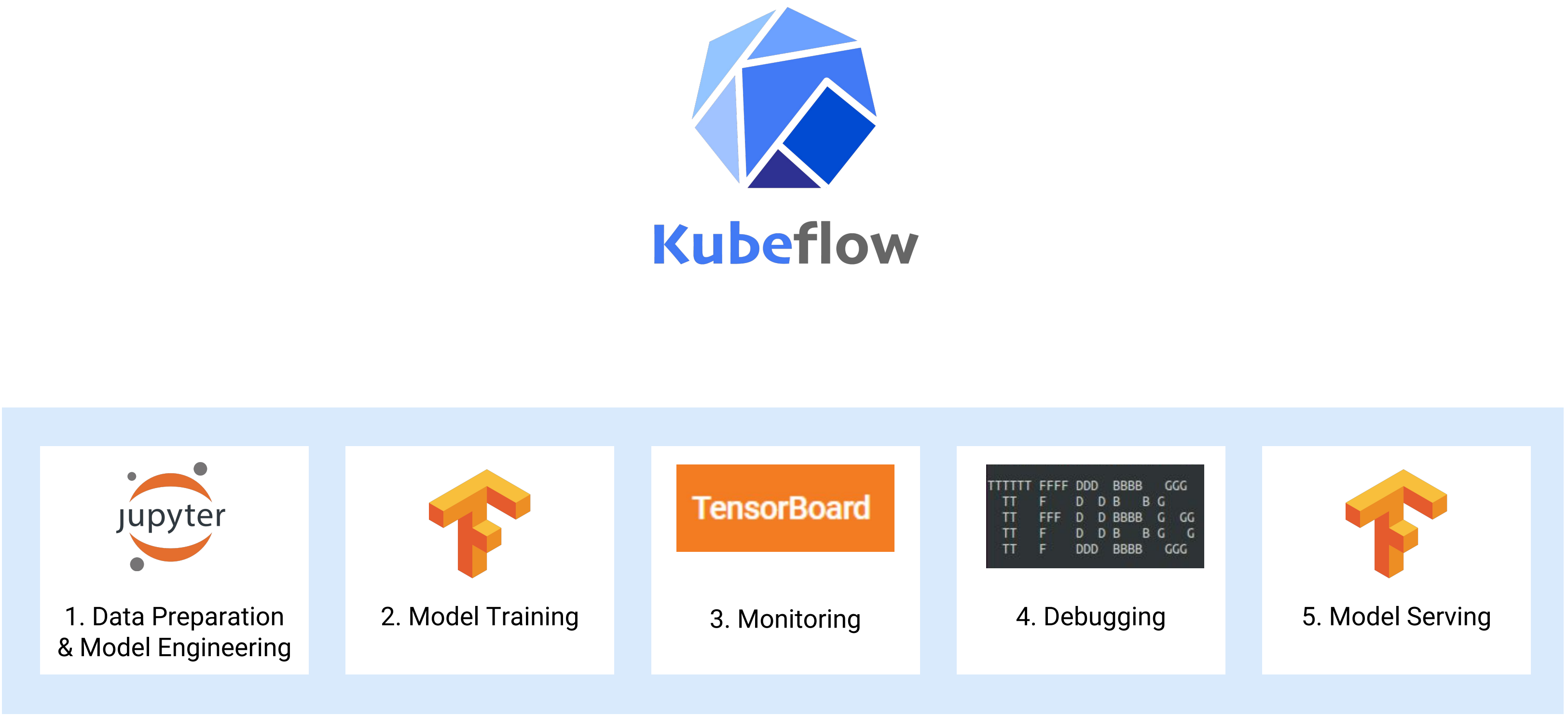
Digit	Confidence
0	0.0022526539396494627
1	2.63791100074684e-10
2	0.4638307988643646
3	0.21909376978874207
4	3.2985670372909226e-07
5	0.29357224702835083
6	0.00019597385835368186
7	5.230629176367074e-05
8	0.020996594801545143
9	5.426473762781825e-06

https://community.cloud.pipeline.ai/admin/app

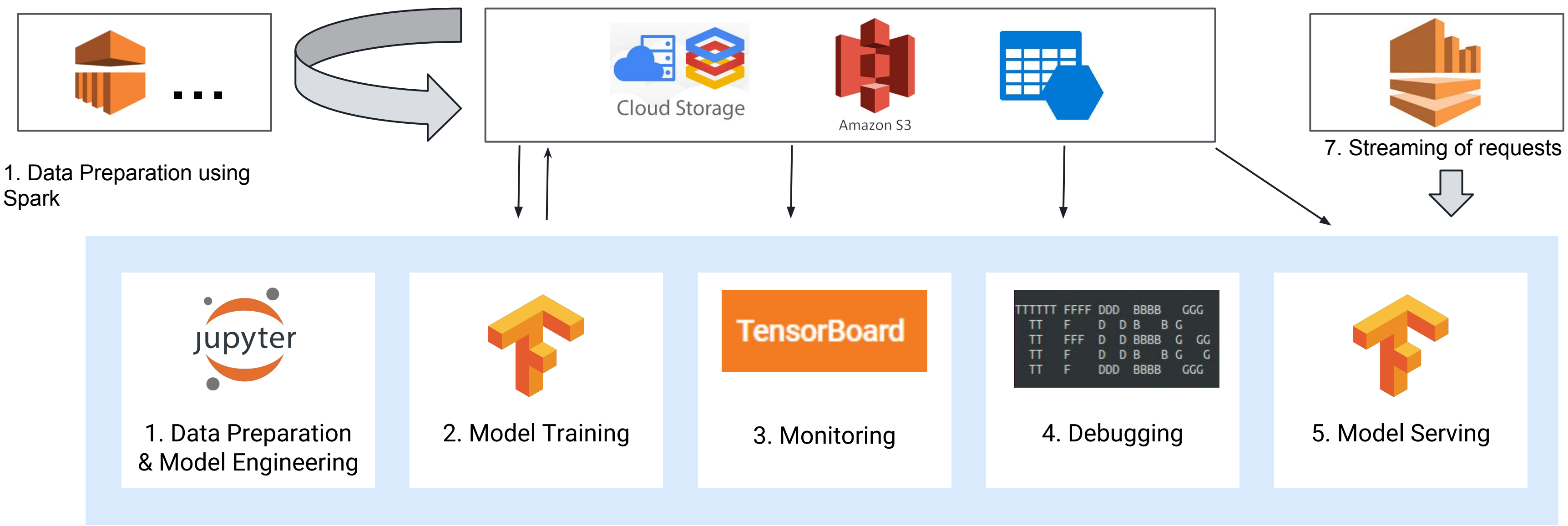
Rendezvous Architecture



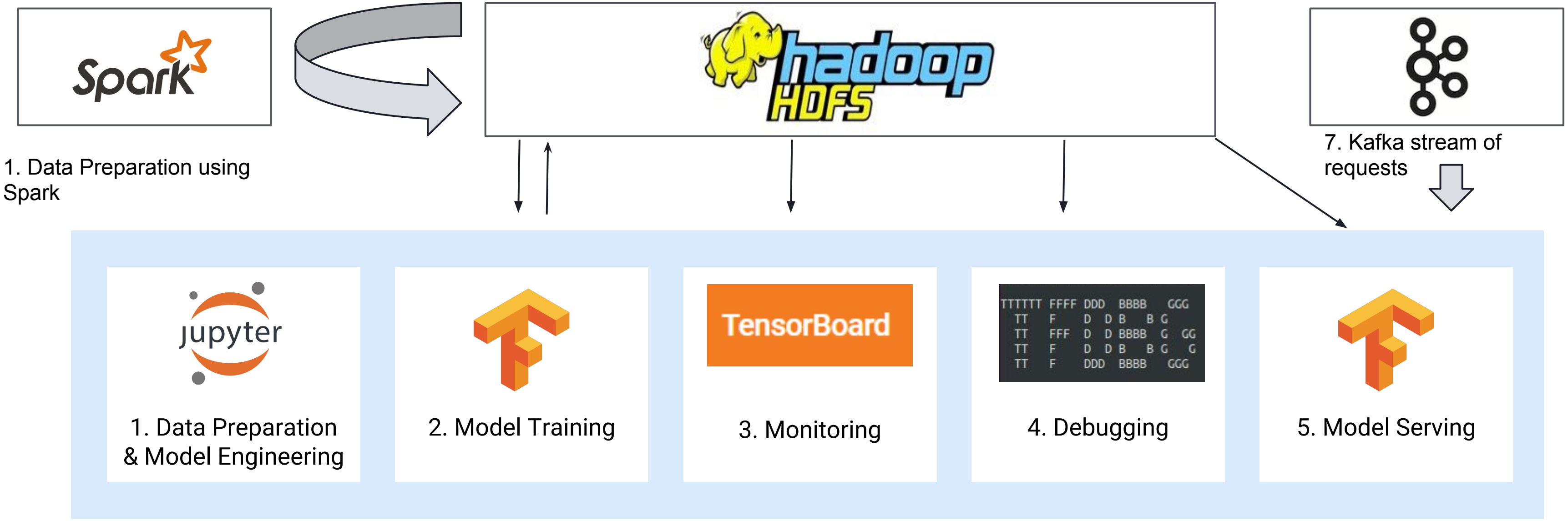
https://mapr.com/ebooks/machine-learning-logistics/ch03.html


Kubeflow
jupyter
1. Data Preparation & Model Engineering
2. Model Training
TensorBoard
3. Monitoring
4. Debugging
5. Model Serving

Public Cloud Pipeline



DIY Open Source Pipeline



Data Science 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

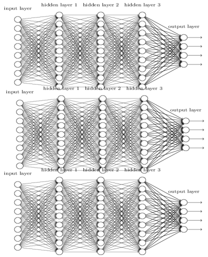
APACHE Spark

PYTORCH

mxnet

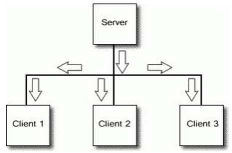
Model Management

Storage of trained Models and Metadata



Model Serving

Use trained Model for Inference



Jenkins **Continuous Integration**

DC/OS Resource and Service Management kubernetes MESOS

DATA SCIENCE

Step 1 → Upload Dataset into Data Store

Step 2 → Clean & Prepare Dataset

Step 3 → Spawn Jupyter Notebook

Step 4 → Explore Data & Develop Model

Step 5 → Model Export

DATAOPS

Automated using CI/CD

Step 6.A → Distributed Training

Step 6.B → Model Testing

Step 6.C → Model Optimization

Step 6.D → Model / Metadata Storage

Step 7 → Model Serving

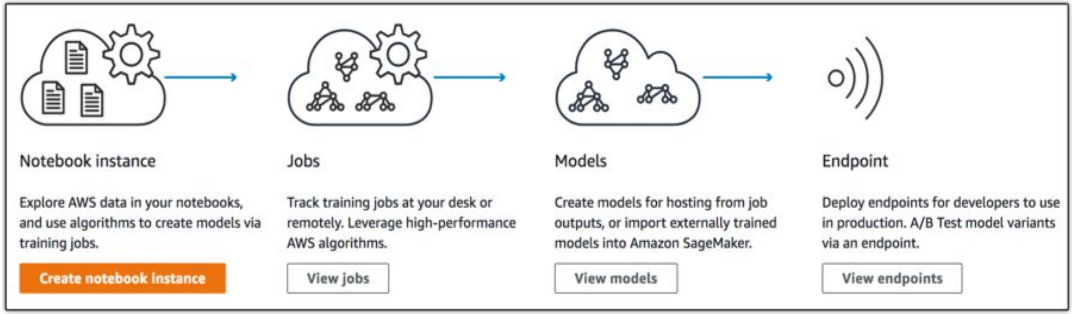
Step 8 → Kafka Queue with Requests

Data Science Platforms

- AWS Sagemaker
  - + Spark, MXNet, TF
  - + Serving/AB
  - - Cloud Only

- Google Datalab/ML-Engine
  - + TF, Keras, Scikit, XGBoost
  - + Serving/AB
  - - Cloud Only
  - - No control of docker images

- KubeFlow
  - + TF Everywhere
  - - TF only



https://medium.com/intuitionmachine/google-and-ubers-best-practices-for-deep-learning-58488a8899b6

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

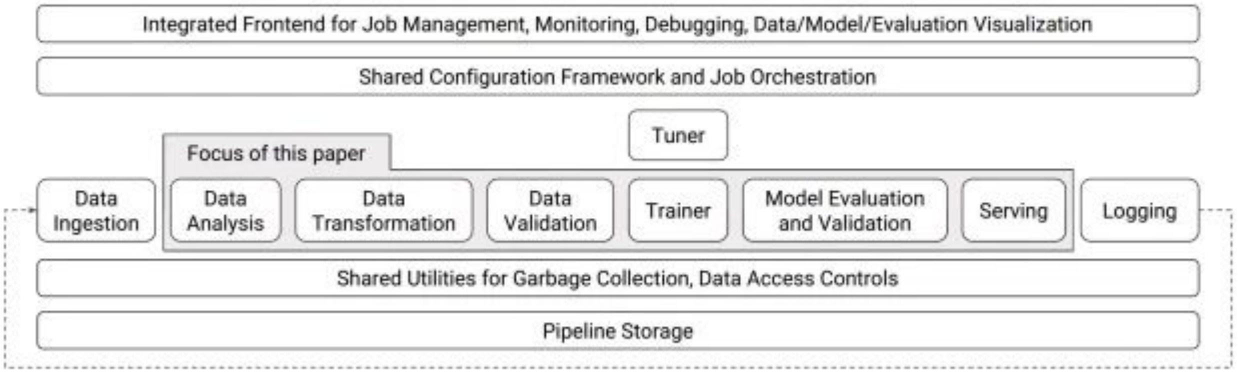


Figure 1: High-level component overview of a machine learning platform.

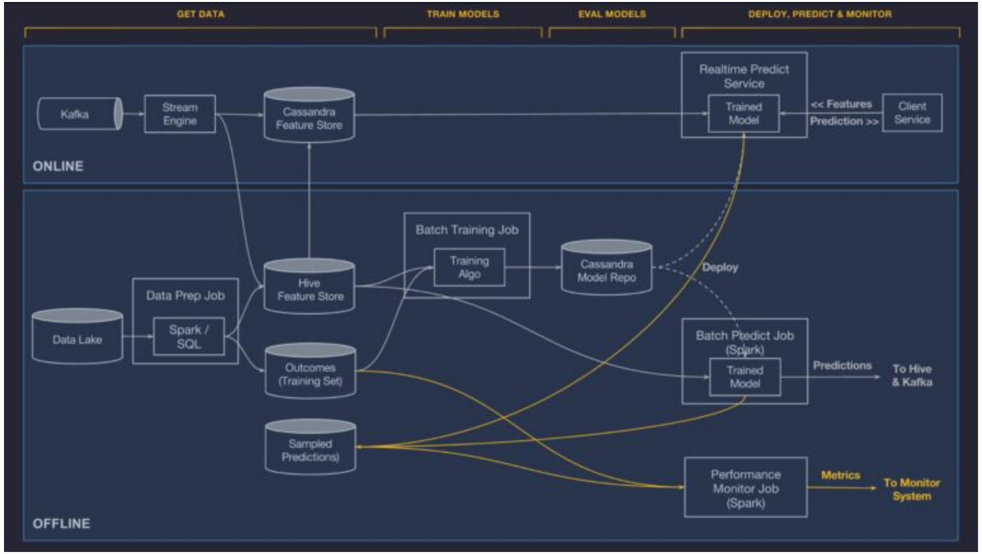
http://www.kdd.org/kdd2017/papers/view/tfx-a-tensorflow-based-production-scale-machine-learning-platform

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

Uber Michelangelo

“..there were no systems in place to build reliable, uniform, and reproducible pipelines for creating and managing training and prediction data at scale.”

- Feature store (later)



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

Challenge: Data Quality

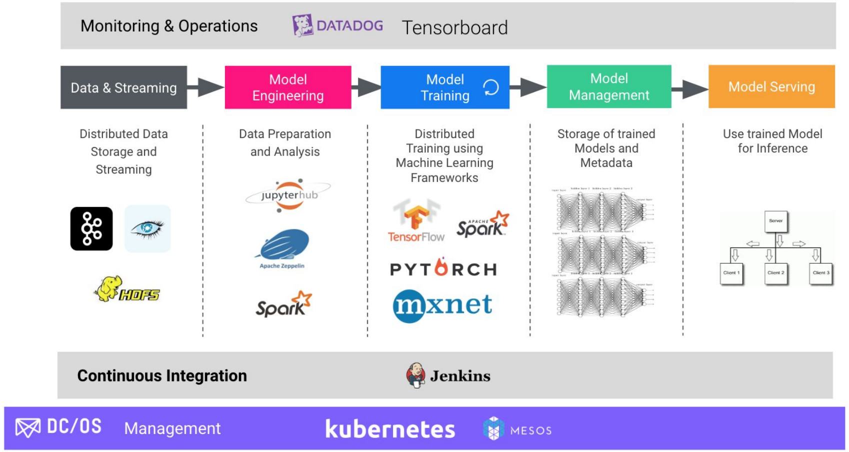
- Data is typically not ready to be consumed by ML job*
  - Data Cleaning
    - Missing/incorrect labels
  - Data Preparation
    - Same Format
    - Same Distribution

* 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

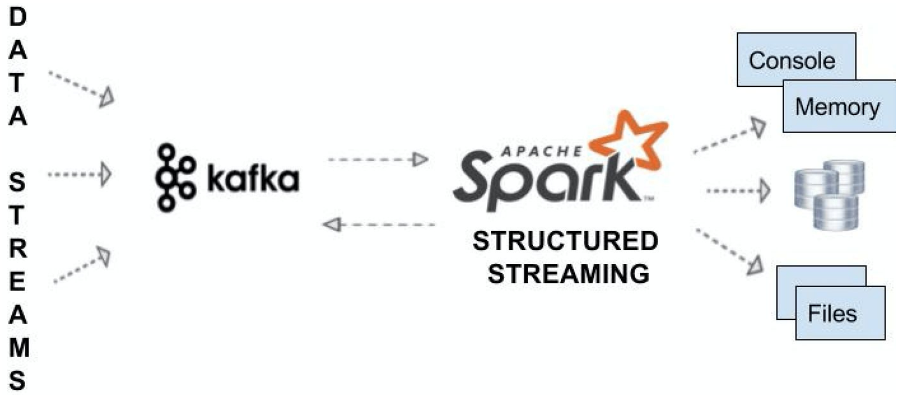
Don't forget about the serving environment!!



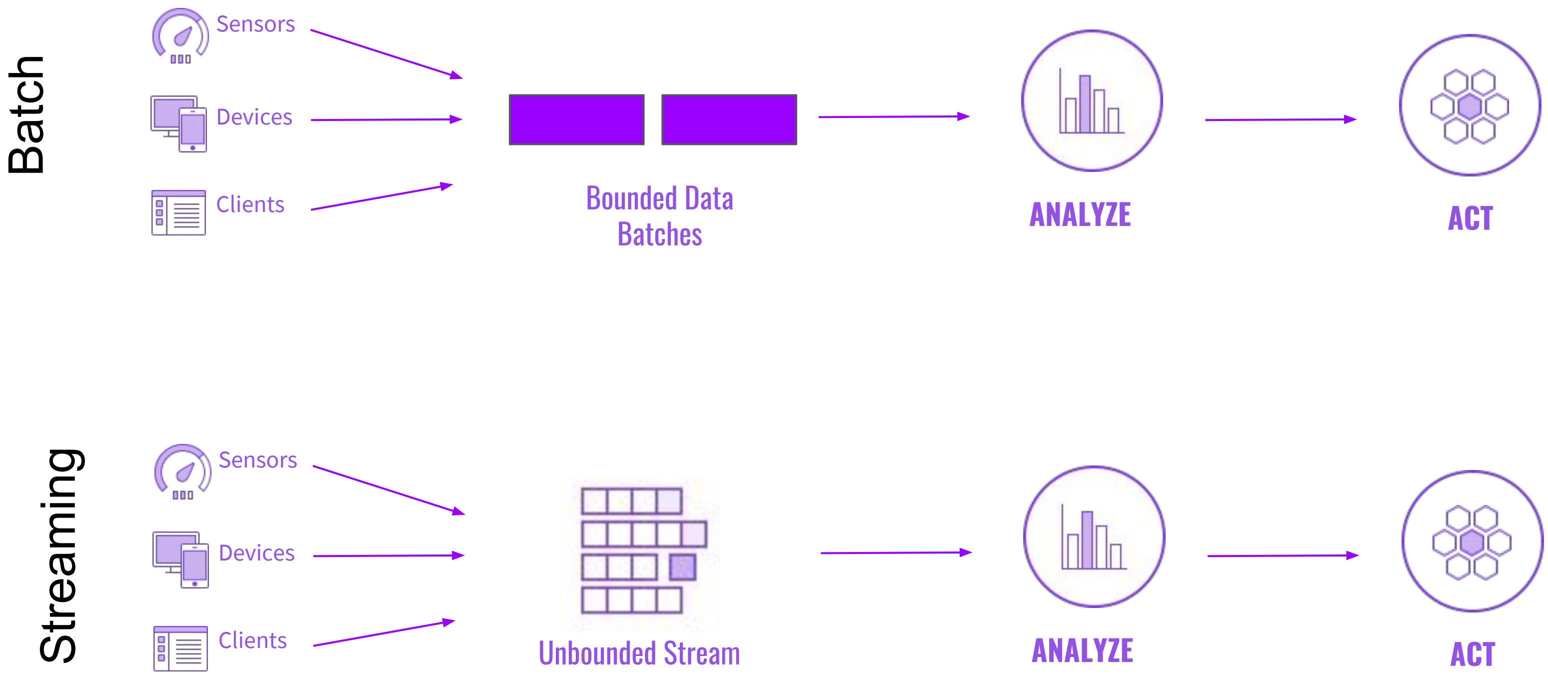
* Demo datasets are a fortunate exception :)

Spark Overview

- Unified Analytics Engine
  - Started as batch system
  - < 2.0: Spark Streaming
    - Micro-Batches
  - > 2.0 Structure Streaming
    - Native Streaming



Batch vs Streaming



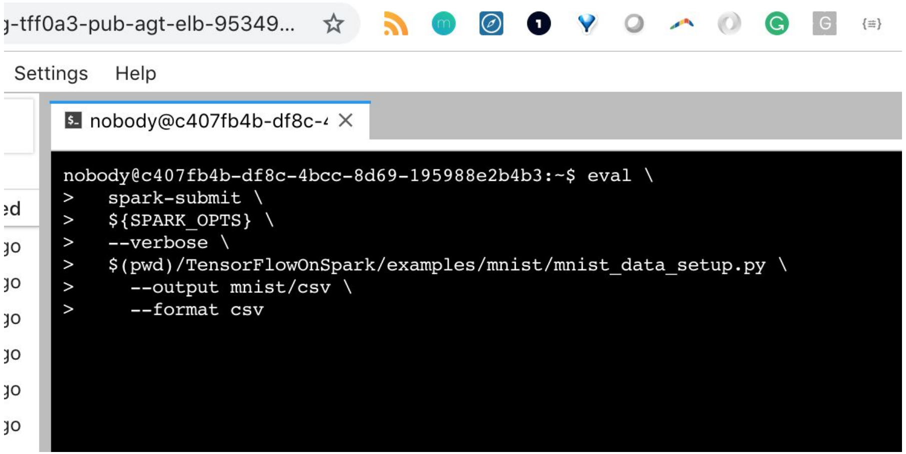
Lab 3: Spark Data Cleaning

```
$ git clone https://github.com/yahoo/TensorFlowOnSpark

$ cd $MESOS_SANDBOX
$ curl -fsSL -O https://s3.amazonaws.com/vishnu-mohan/tensorflow/mnist/mnist.zip
$ unzip mnist.zip

$ // Should return error
$ hdfs dfs -ls  mnist/
```

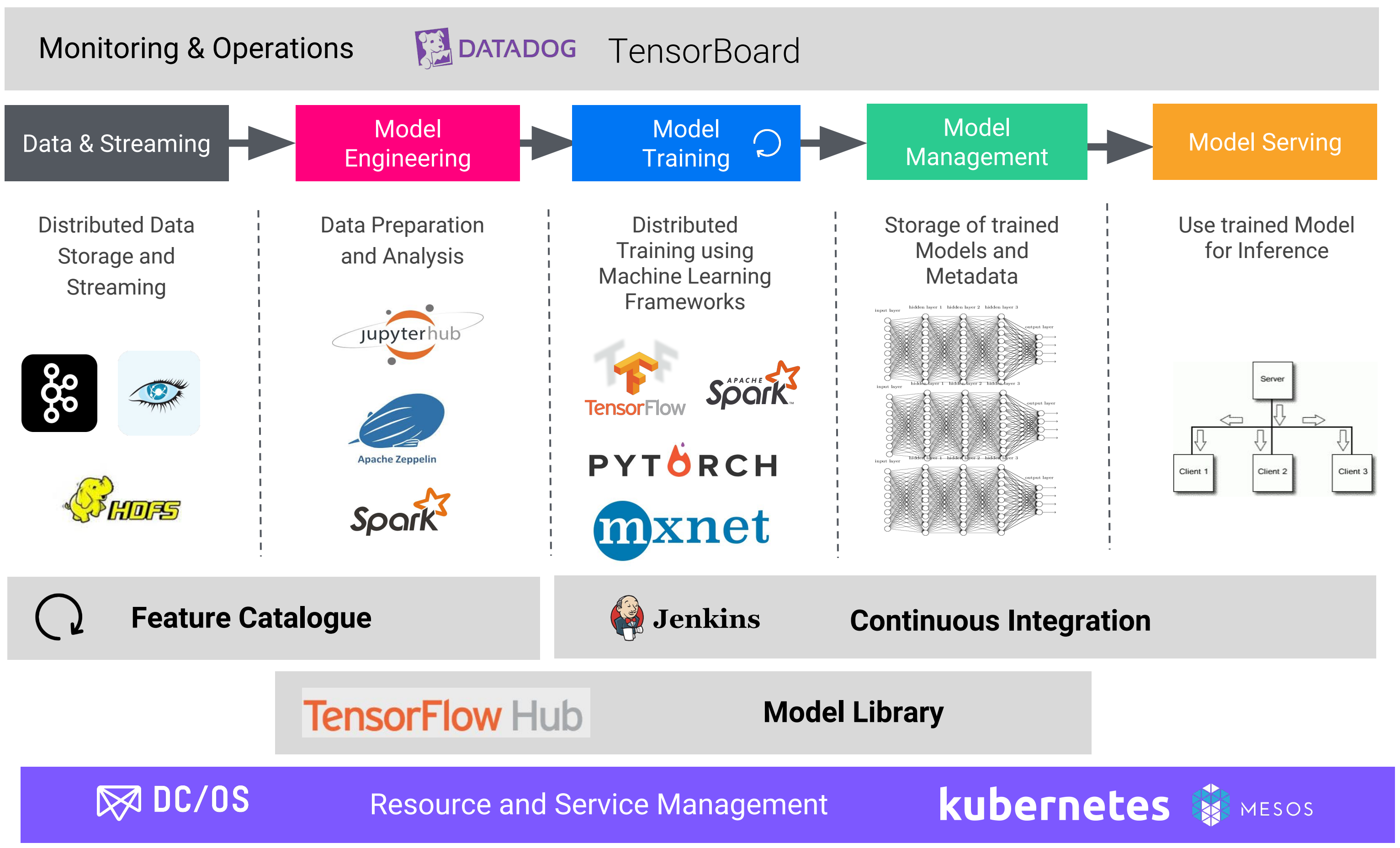
https://github.com/dcos-labs/dcos-jupyterlab-service/blob/master/notebooks/TFoS.ipynb

Lab 1: Spark Data Cleaning

```
eval \
  spark-submit \
  ${SPARK_OPTS} \
  --verbose \
$(pwd)/TensorFlowOnSpark/examples/mnist/mnist_data_setup.py \
    --output mnist/csv \
    --format csv

$ // Should not return an error
$ hdfs dfs -ls  mnist/
```



https://github.com/dcos/demos/tree/master/jupyterlab/1.11



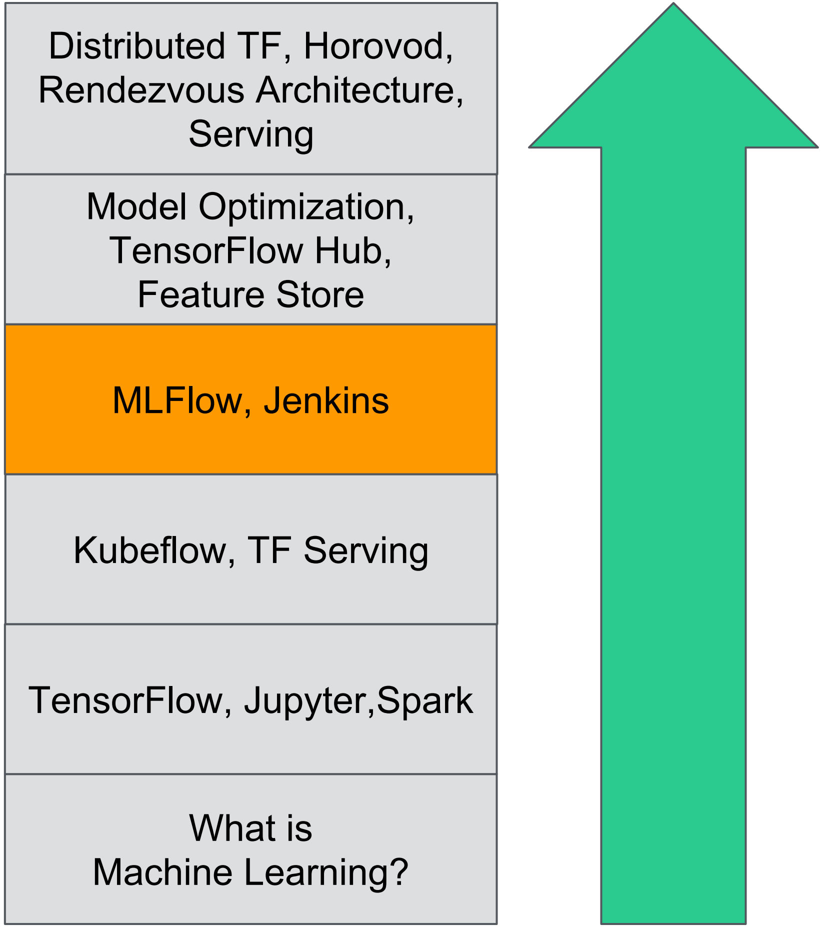
Monitoring & Operations
DATADOG
TensorBoard
Data & Streaming
Model Engineering
Model Training
Model Management
Model Serving
Distributed Data Storage and Streaming
HDFS
Data Preparation and Analysis
jupyterhub
Apache Zeppelin
Spark
Distributed Training using Machine Learning Frameworks
TensorFlow
APACHE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

Lab: (Optional) KubeFlow

https://www.kubeflow.org/docs/started/getting-started-multipass/

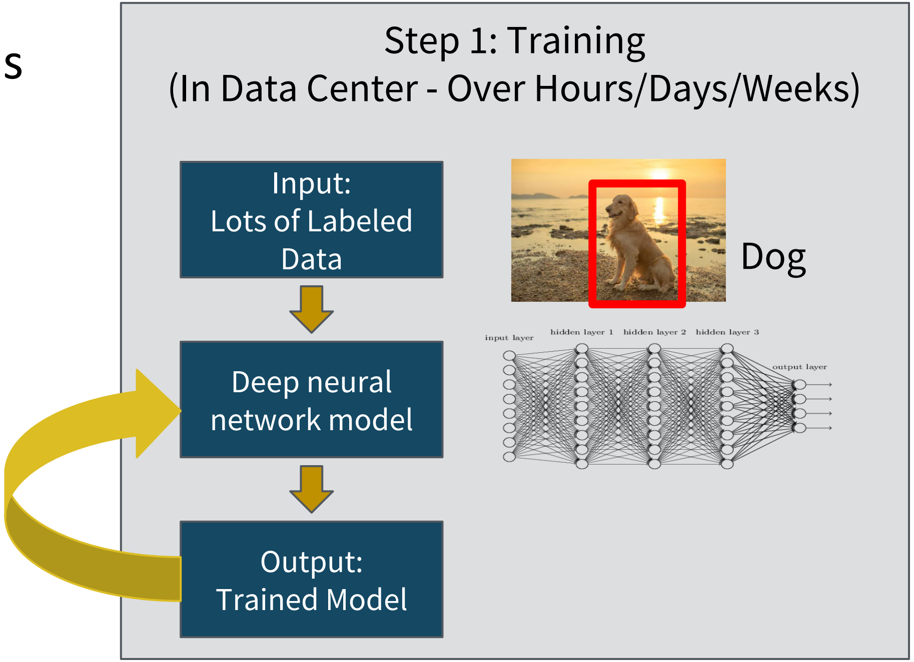
Automation

- Need for reproducibility
  - MLFlow
  - Jenkins
- **Open Source Technologies**
  - MLFlow
  - Jenkins
- **Labs**
  - Jenkins
  - [opt] MLFlow

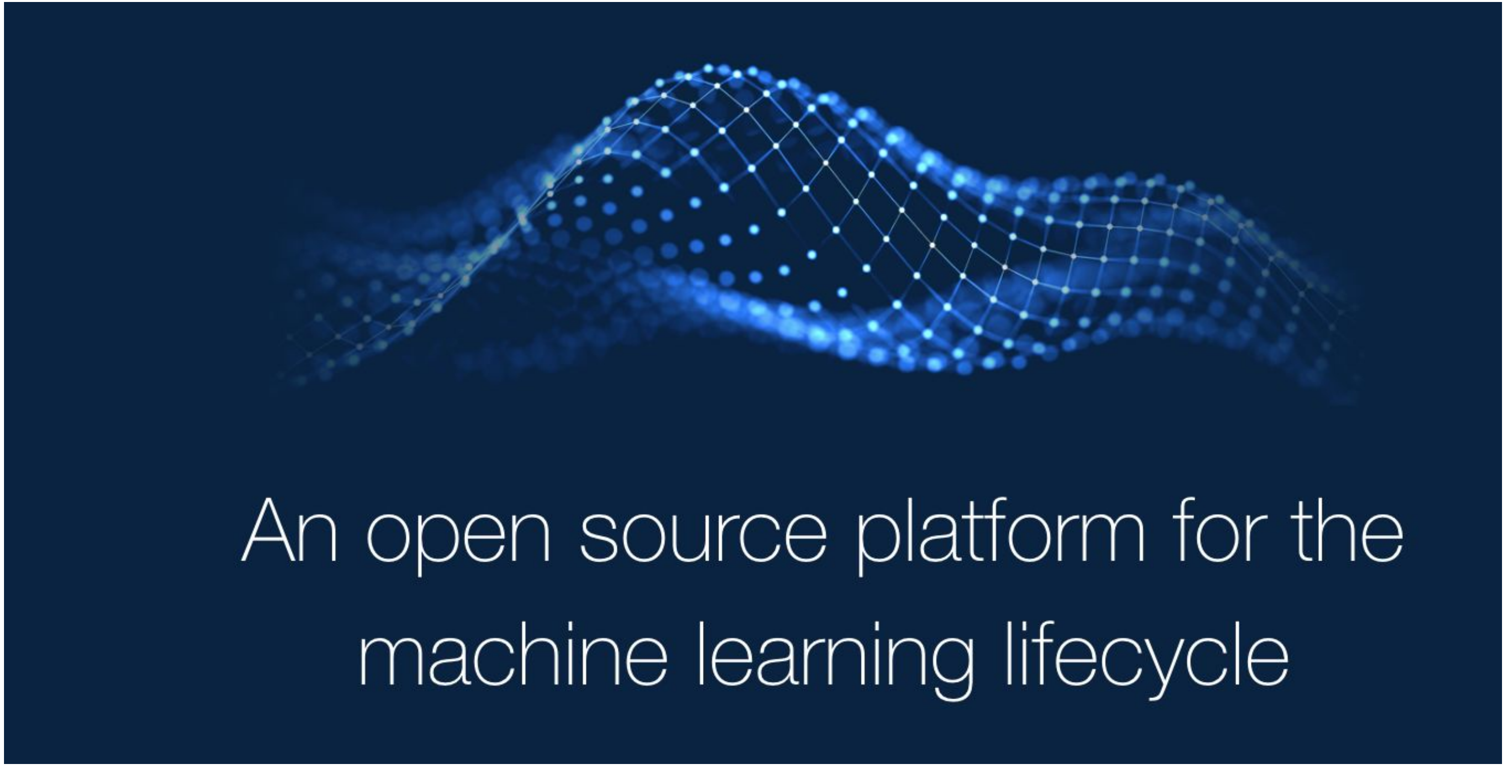


Challenge: Reproducible Builds

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

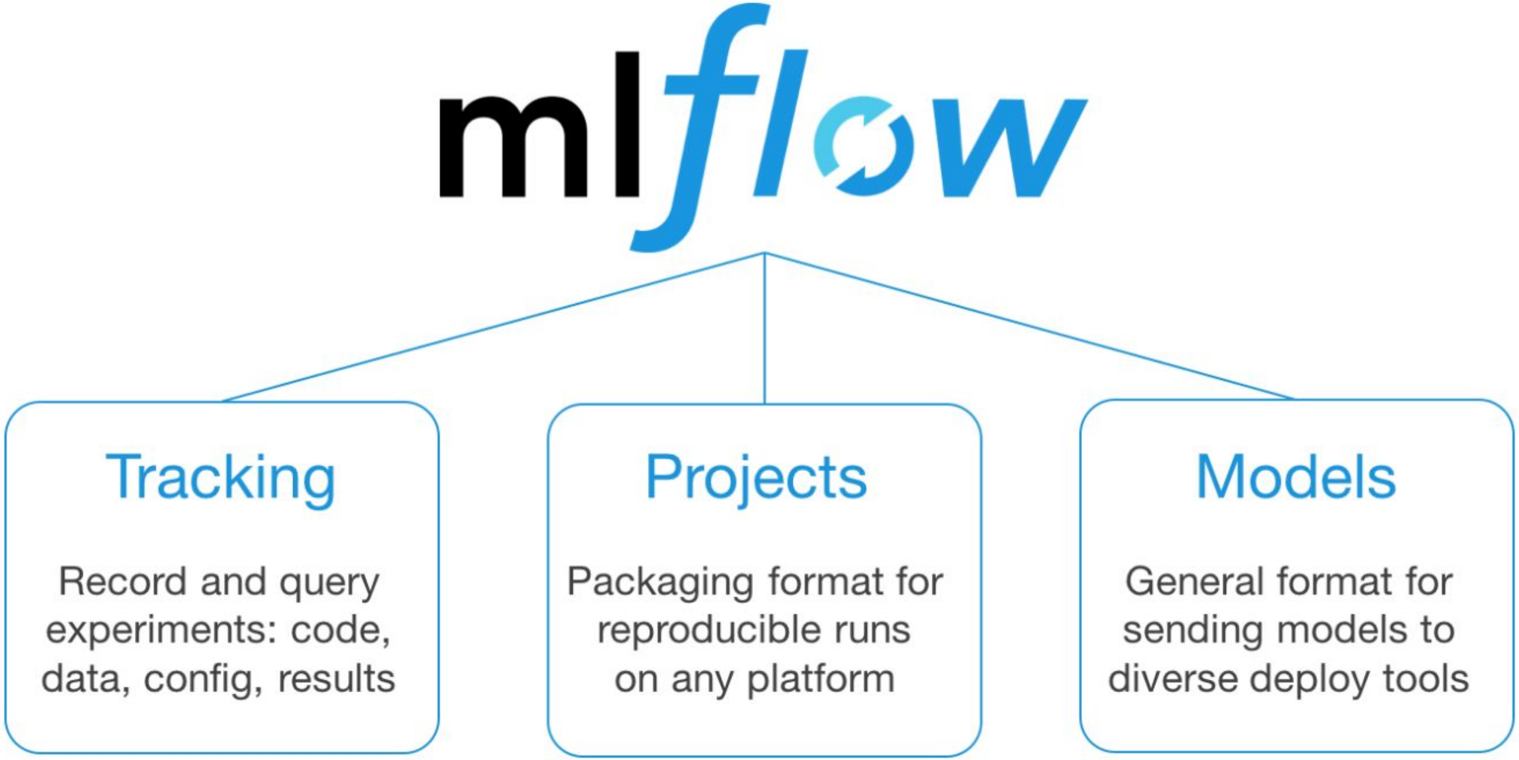


MFlow



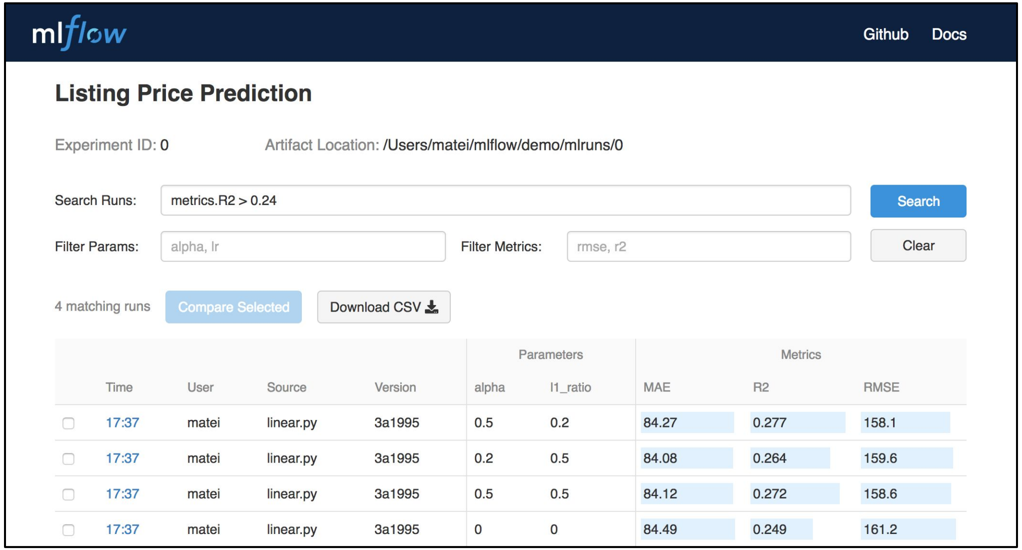
https://mlflow.org/

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

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")
```



				Parameters		Metrics		
Time	User	Source	Version	alpha	l1_ratio	MAE	R2	RMSE
17:37	matei	linear.py	3a1995	0.5	0.2	84.27	0.277	158.1
17:37	matei	linear.py	3a1995	0.2	0.5	84.08	0.264	159.6
17:37	matei	linear.py	3a1995	0.5	0.5	84.12	0.272	158.6
17:37	matei	linear.py	3a1995	0	0	84.49	0.249	161.2

MFlow Project

```
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
```

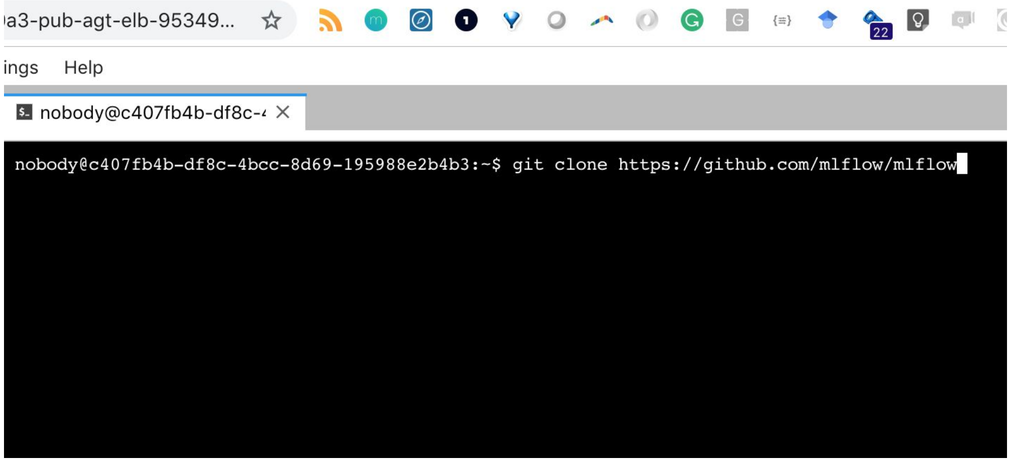
MFlow Model

```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
```

Lab 2: MLflow

```
$ git clone https://github.com/mlflow/mlflow
$ cd mlflow/
$ python
examples/sklearn_elasticnet_wine/train.py
$ mlflow run
https://github.com/mlflow/mlflow-example.git
-P alpha=0.42
$ mlflow sklearn serve -m
./mlruns/0/024e295715d64b3fb0730008a07c
1f75/artifacts/model -p 1234
```



https://www.mlflow.org/docs/latest/quickstart.html

Lab 2: MLflow

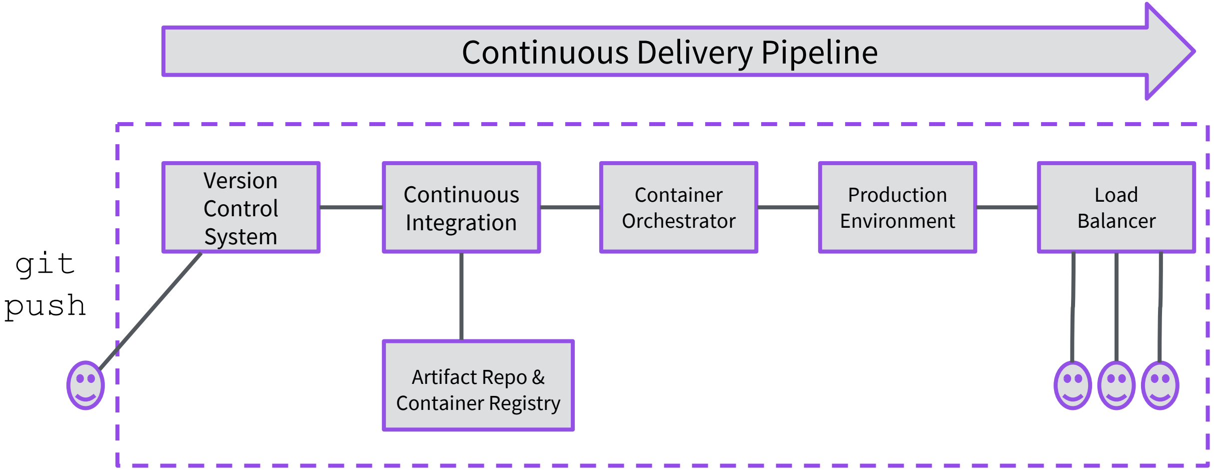
```
curl -X POST -H "Content-Type:application/json"
--data '[{"fixed acidity": 6.2, "volatile
acidity": 0.66, "citric acid": 0.48, "residual
sugar": 1.2, "chlorides": 0.029, "free sulfur
dioxide": 29, "total sulfur dioxide": 75,
"density": 0.98, "pH": 3.33, "sulphates": 0.39,
"alcohol": 12.8}]'
http://127.0.0.1:1234/invocations
```

```
{"predictions": [6.379428821398614]}
```

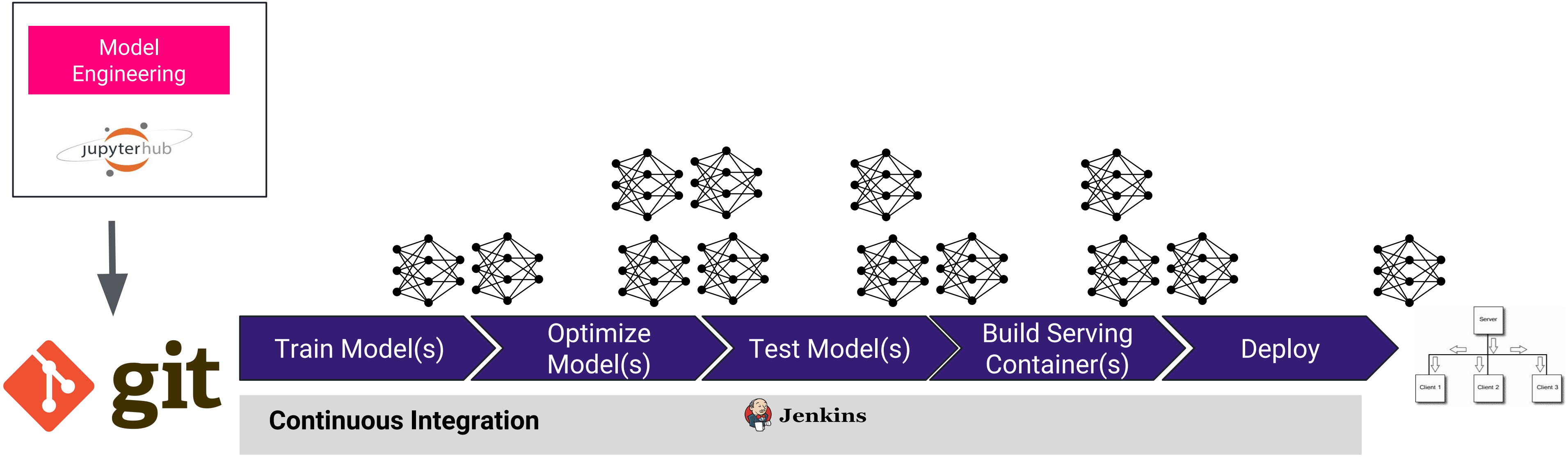
https://www.mlflow.org/docs/latest/quickstart.html

Challenge: CI/CD

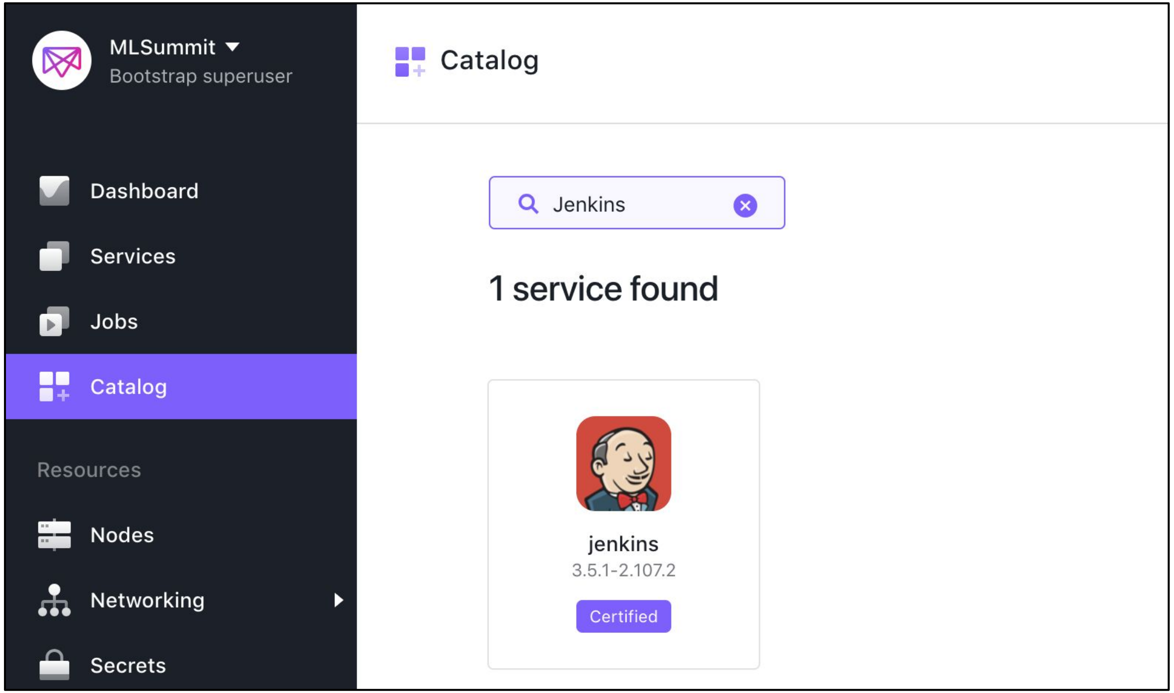
- Automatic Build, Test, Deploy
- Quality Barrier
- Options
  - Jenkins
  - Gitlab
  - TravisCI
  - Skaffold
  - Spinnaker



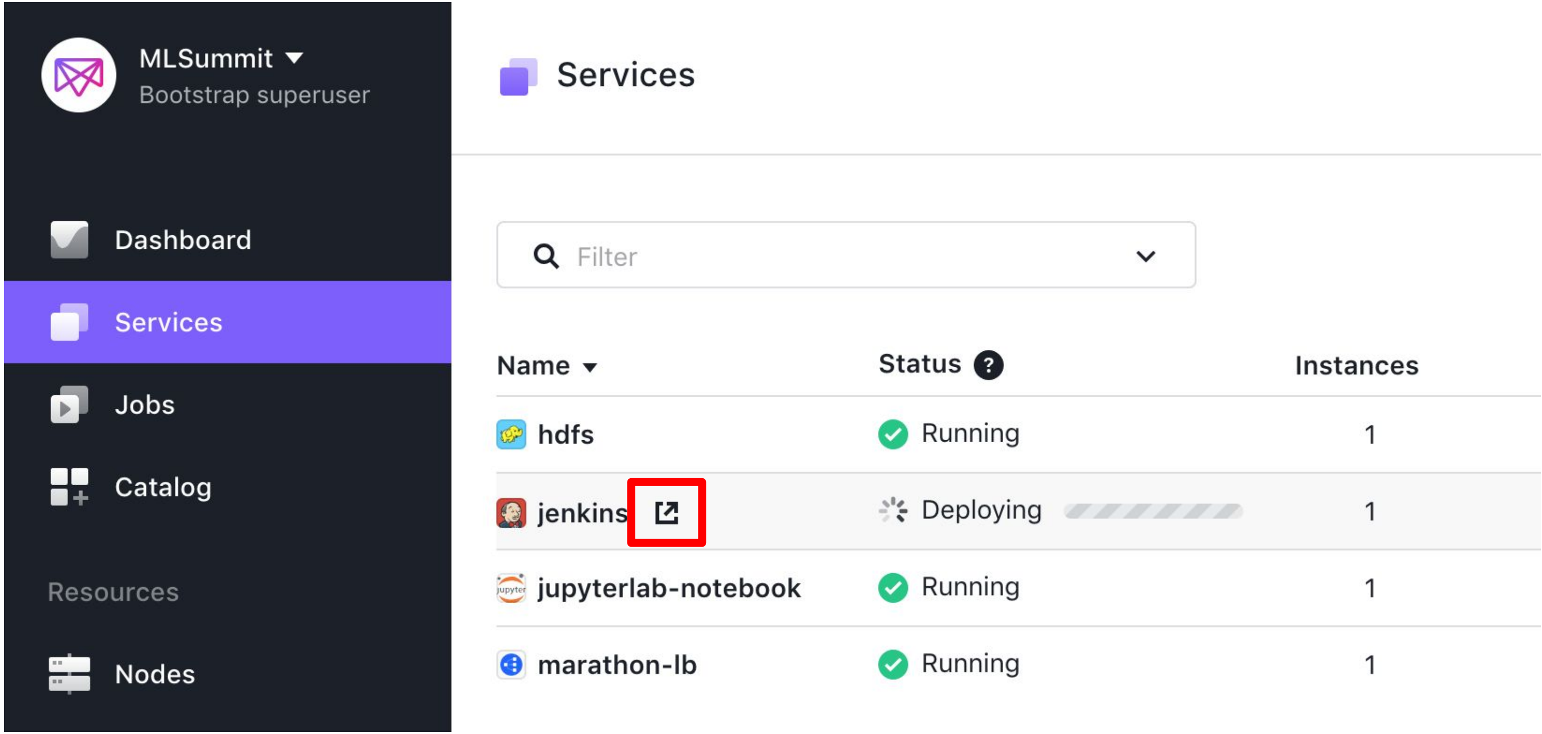
CI/CD for Data Science



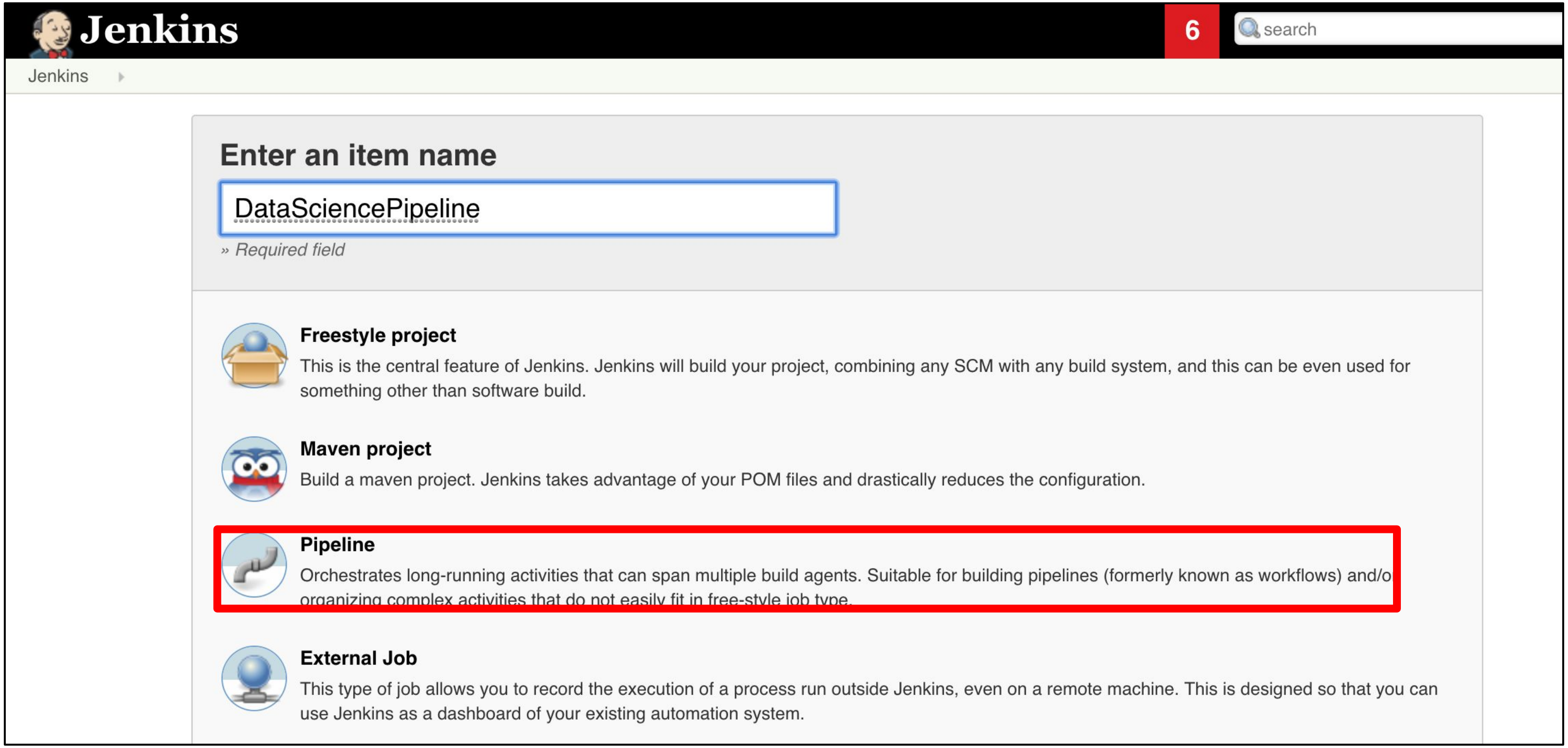
Lab 3: Jenkins



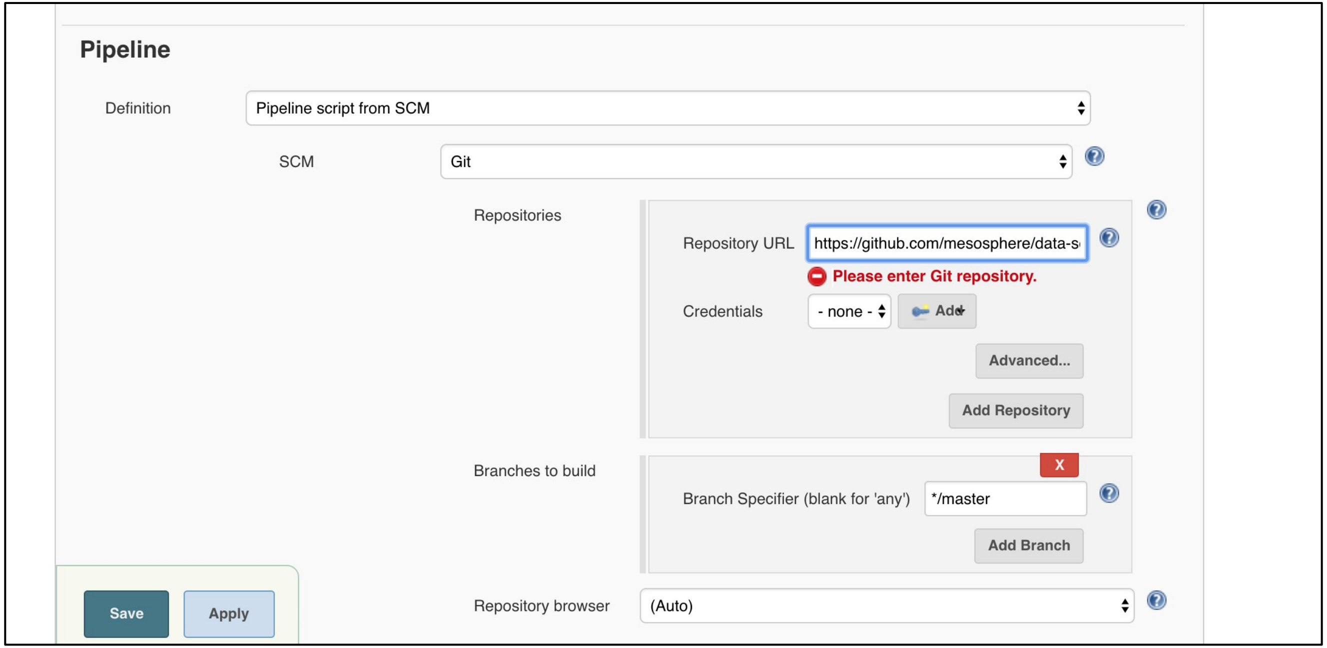
Lab 3: Jenkins



Lab 3: Jenkins

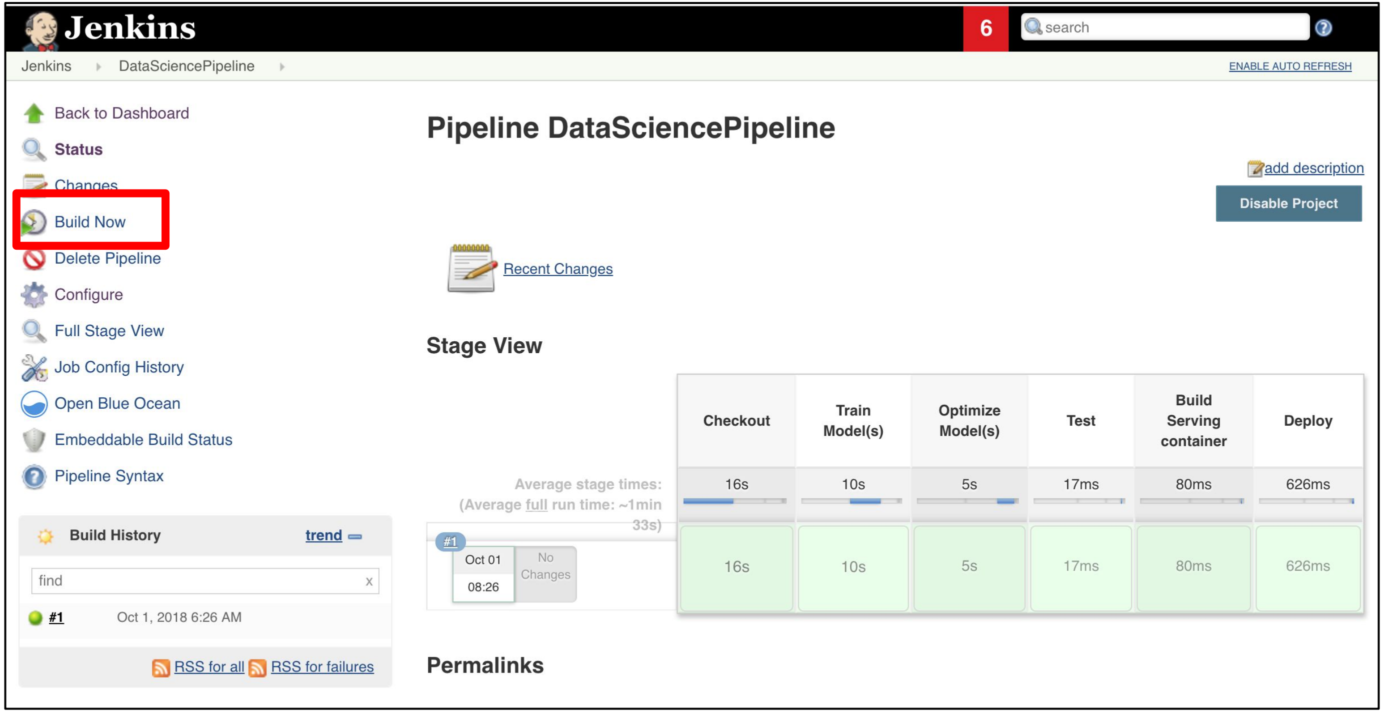


Lab 3: Jenkins



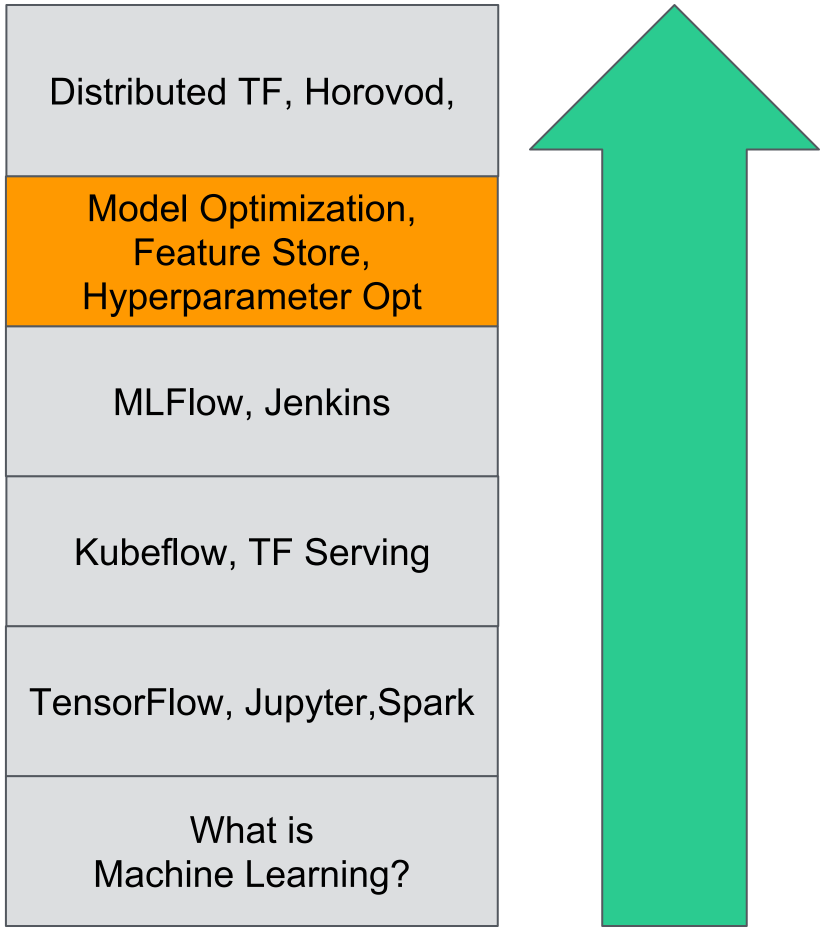
https://github.com/mesosphere/data-science-cicd

Lab 3: Jenkins



Model Management

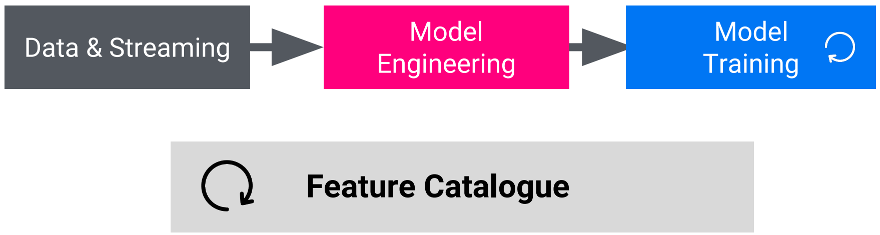
- How to manage Models
- **Open Source Technologies**
  - TensorFlow Hub
  - Dask
- **Labs**
  - Dask Hyperparameter Optimization



Challenge: Data (Preprocessing) Sharing

- Preprocessed Data Sets valuable
  - Sharing
  - Automatic Refresh

- Feature Catalogue ≈
  Preprocessing Cache + Discovery



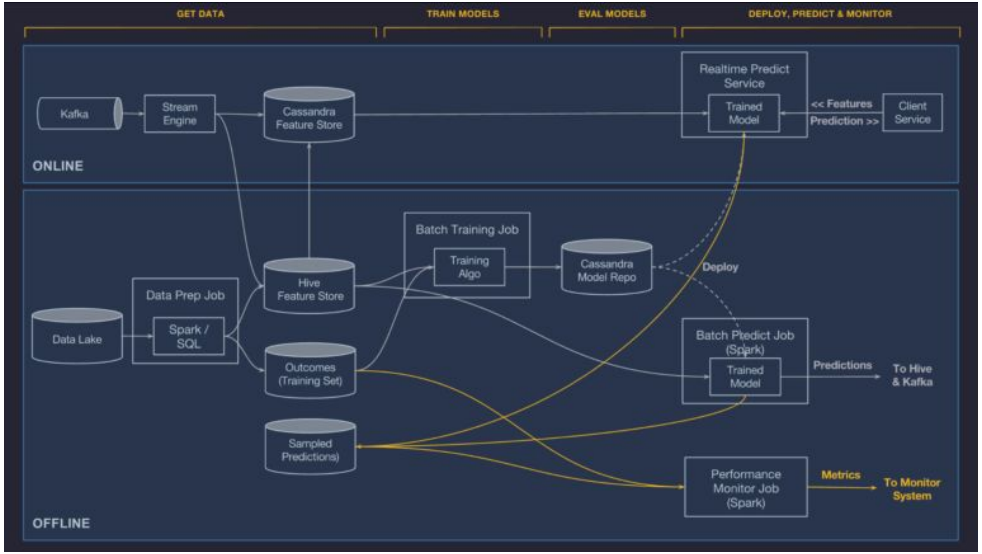
Feature Store

https://techblog.appnexus.com/lessons-learned-from-building-scalable-machine-learning-pipelines-822acb3412ad

Uber Michelangelo

"..there were no systems in place to build reliable, uniform, and reproducible pipelines for creating and managing training and prediction data at scale."

- **Feature store**

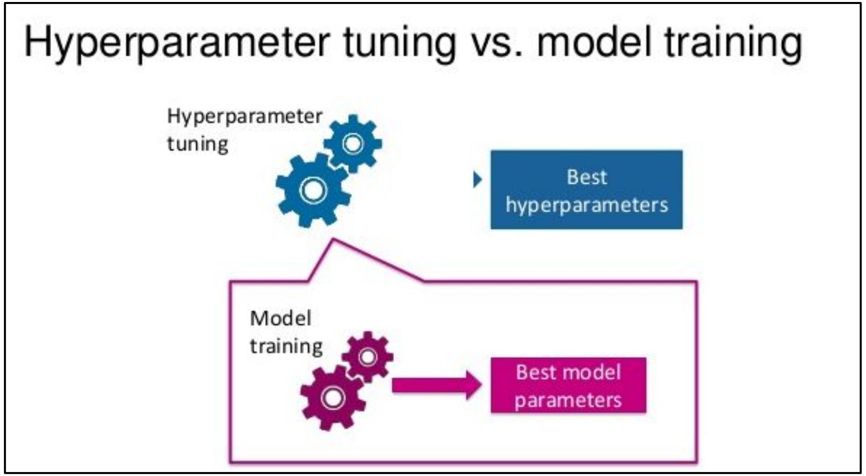


https://eng.uber.com/michelangelo/

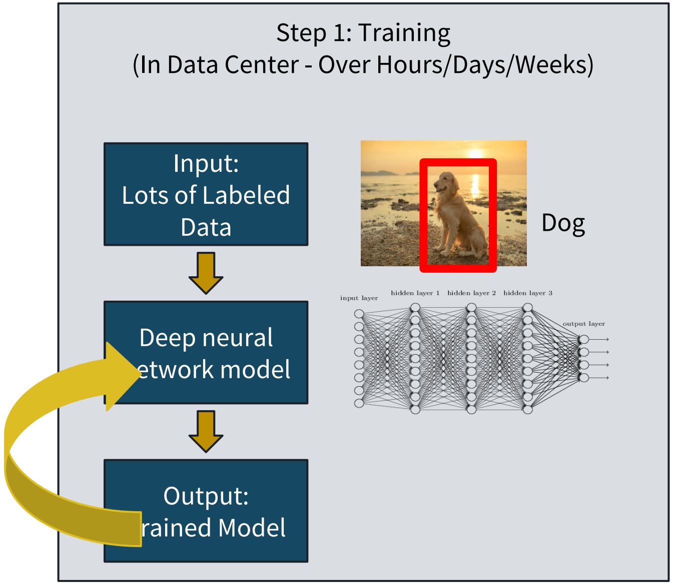
Feature Engineering

https://www.featuretools.com/

Hyperparameter Optimization

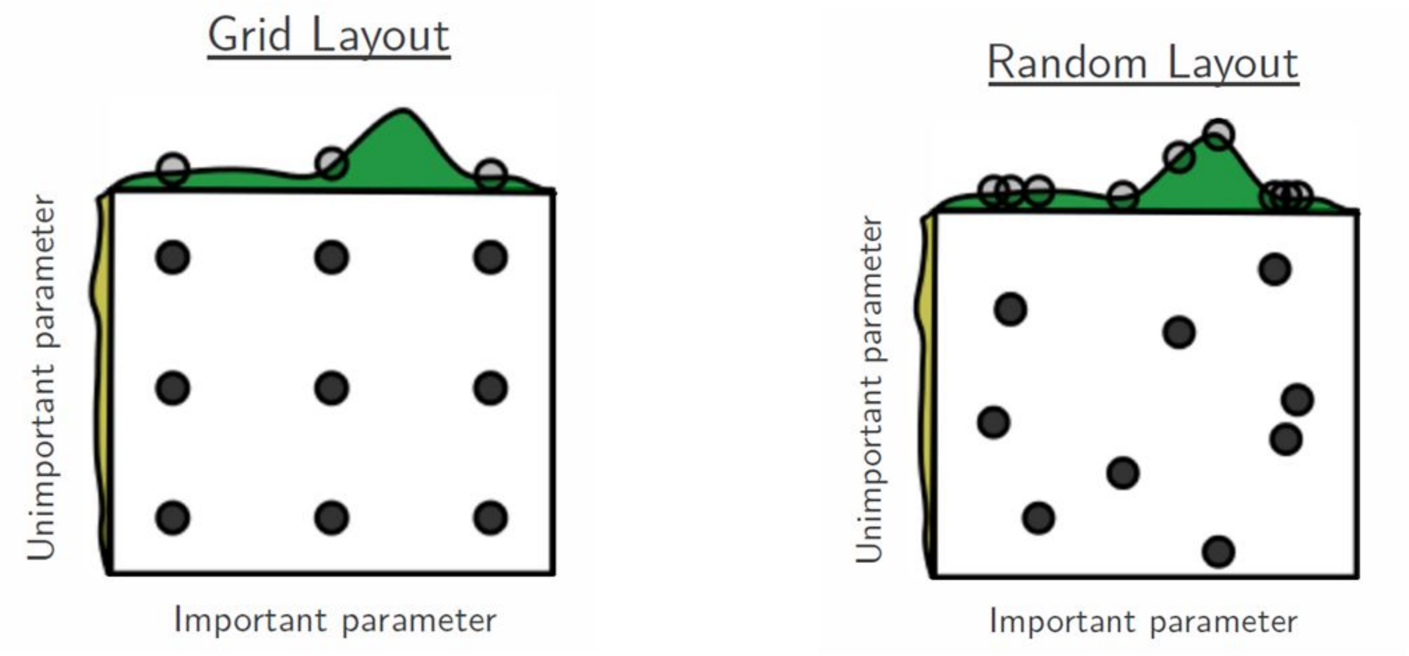


- Networks Shape
- Learning Rate
- ...



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

Hyperparameter Search



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

Dask-ML for Hyperparameter Search

```python
# use a full grid over all parameters
param_grid = {
    "C": [1e-5, 1e-3, 1e-1, 1],
    "fit_intercept": [True, False],
    "penalty": ["l1", "l2"]
}

clf = LogisticRegression()

# run grid search
dk_grid_search = GridSearchCV(clf, param_grid=param_grid, n_jobs=-1)
sk_grid_search = ms.GridSearchCV(clf, param_grid=param_grid, n_jobs=-1)
```

https://dask-ml.readthedocs.io/en/stable/examples/hyperparameter-search.html

Lab [Optional] Dask

Dask-ML

Dask-ML provides scalable machine learning in Python using Dask alongside popular machine learning libraries like Scikit-Learn.

You can try Dask-ML on a small cloud instance by clicking the following button:

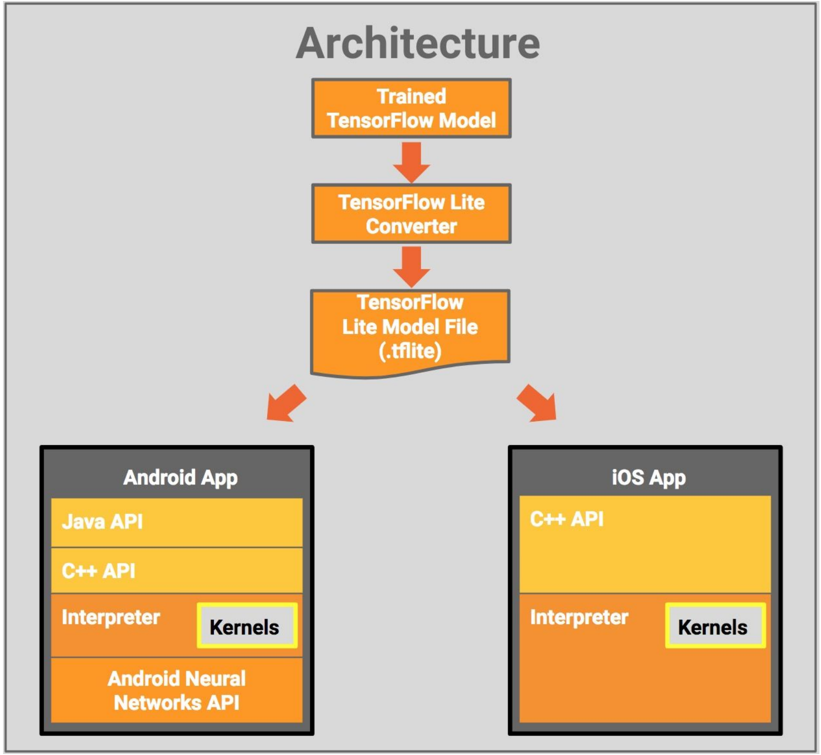
launch binder

```python
import dask.dataframe as dd
df = dd.read_parquet('...')
data = df[['age', 'income', 'married']]
labels = df['outcome']

from dask_ml.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(data, labels)
```

https://mybinder.org/v2/gh/dask/dask-examples/master?filepath=machine-learning.ipynb

Challenge: Serving Environment

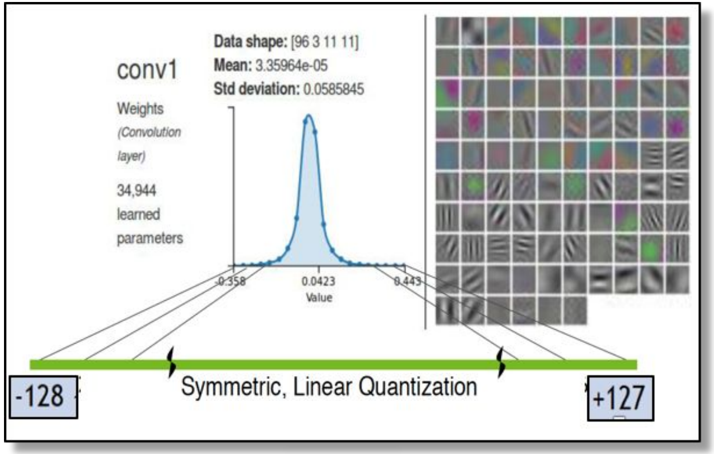
- Different Serving Environments
  - Mobile
  - GPU
  - CPU
- Small/Fast model without losing too much performance
- 500 KB models….



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'
```

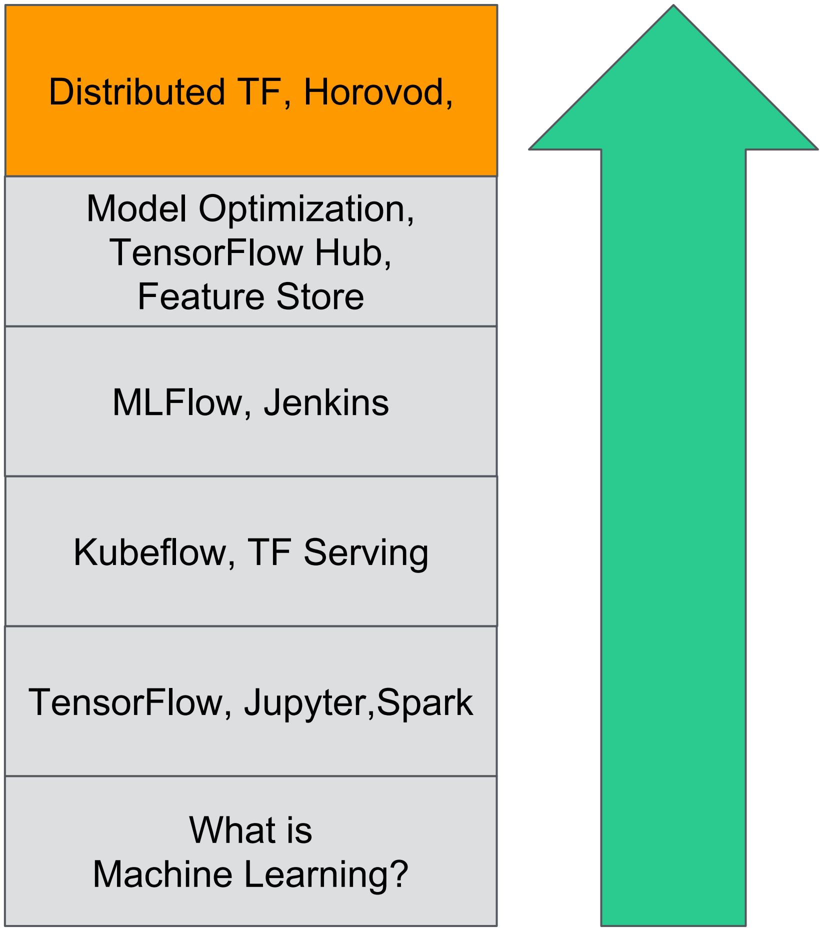
Model Optimization



	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

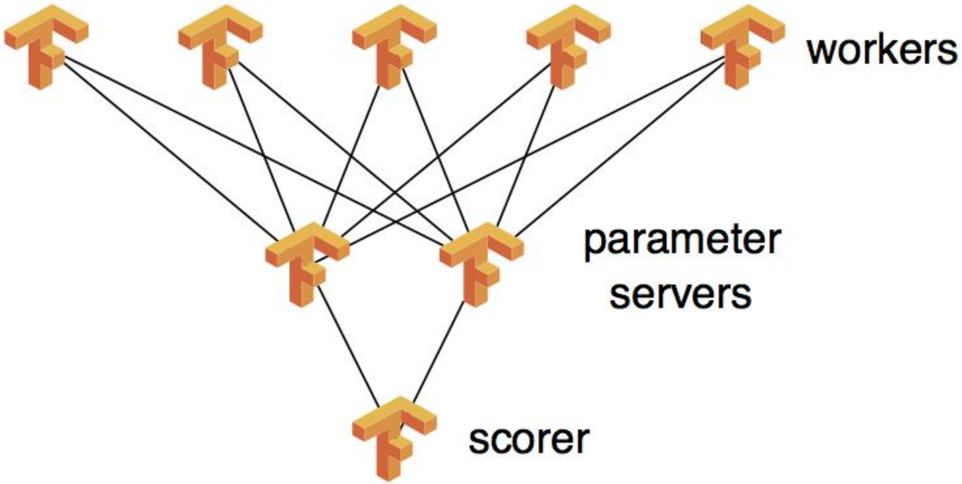
Distributed TensorFlow

- How to distribute TensorFlow
  - {TF, Horovod}onSpark
- **Open Source Technologies**
  - TensorFlow
  - Spark
- **Labs**
  - TFonSpark

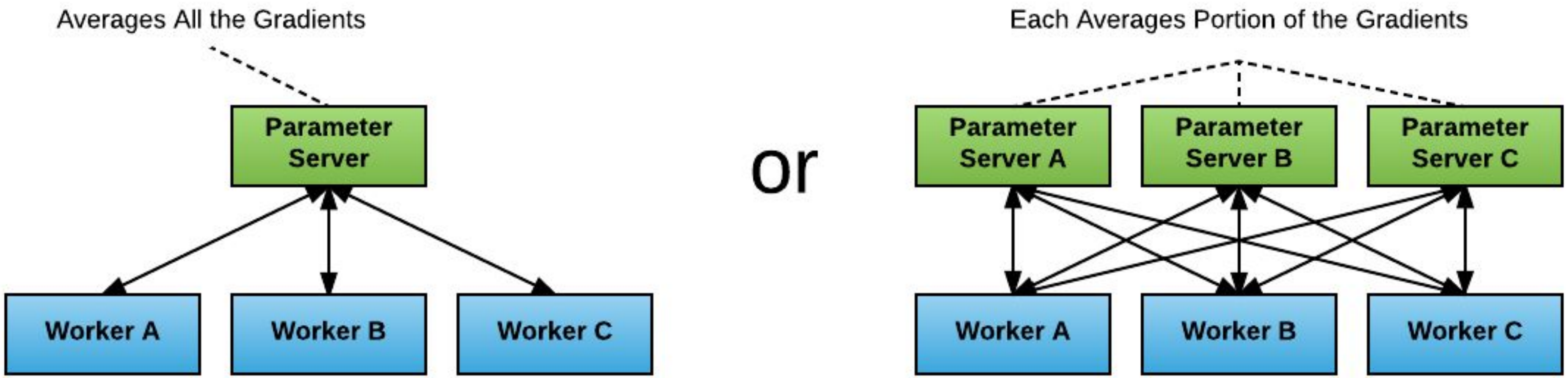


TensorFlow Overview

Challenge: Distributed TensorFlow



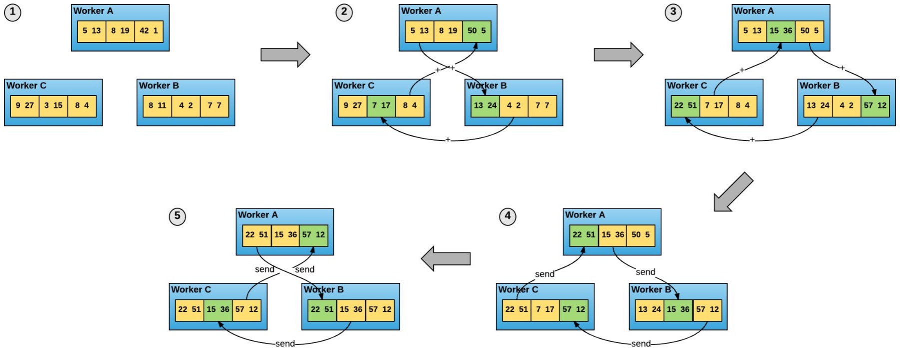
Challenge: Distributed TensorFlow



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

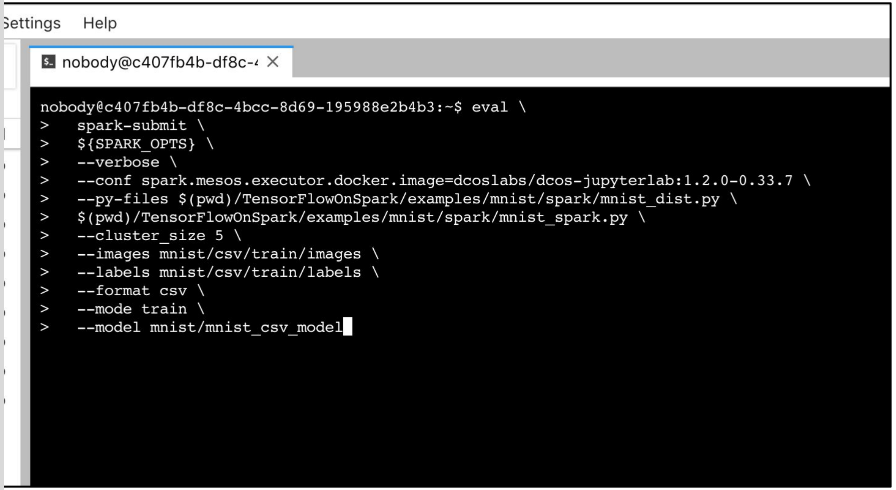
Challenge: “Libraries”

- Different Frameworks
- Existing architectures
- Pretrained models

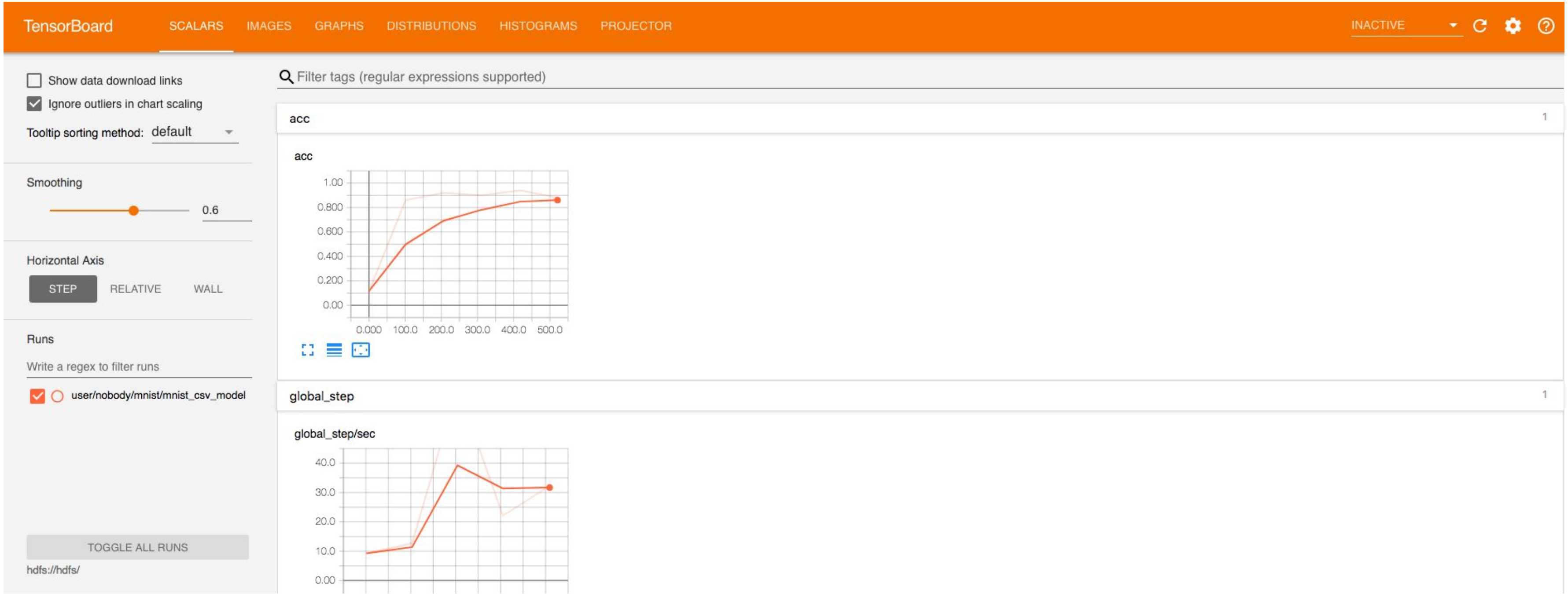
```python
import tensorflow as tf
import tensorflow_hub as hub

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))
```

Lab 4: TensorFlow

```
eval \
  spark-submit \
  ${SPARK_OPTS} \
  --verbose \
  --conf
spark.mesos.executor.docker.image=dcoslabs/dcos-jupyterlab:1.2.0-0.
33.7 \
  --py-files
$(pwd)/TensorFlowOnSpark/examples/mnist/spark/mnist_dist.py \
  $(pwd)/TensorFlowOnSpark/examples/mnist/spark/mnist_spark.py \
  --cluster_size 5 \
  --images mnist/csv/train/images \
  --labels mnist/csv/train/labels \
  --format csv \
  --mode train \
  --model mnist/mnist_csv_model
```

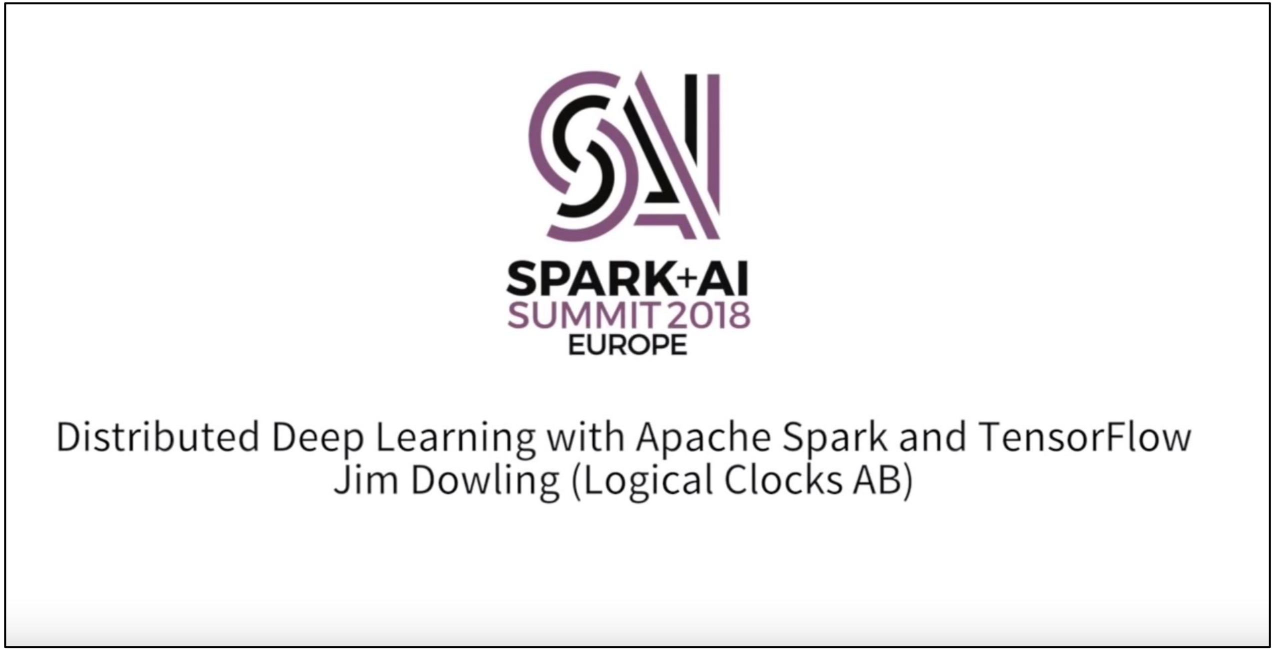


Lab 4: TensorFlow



`<VHOST>/jupyterlab-notebook/tensorboard`

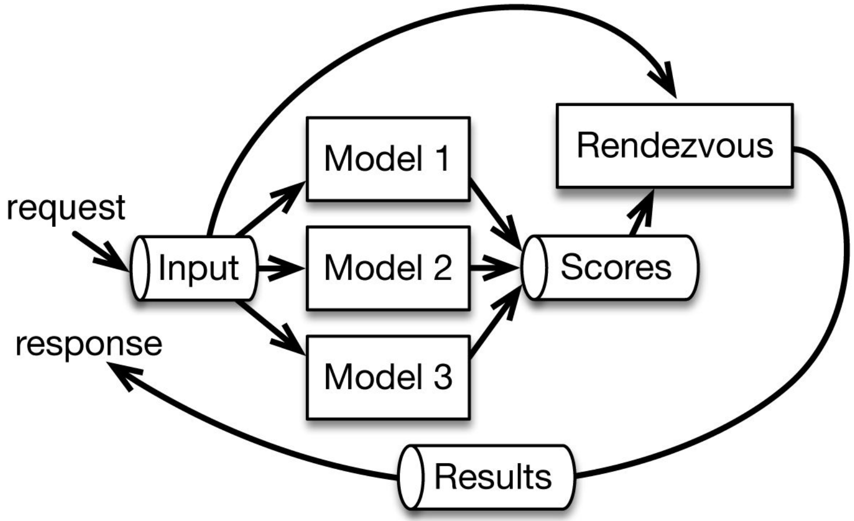
Further Reading/Watching



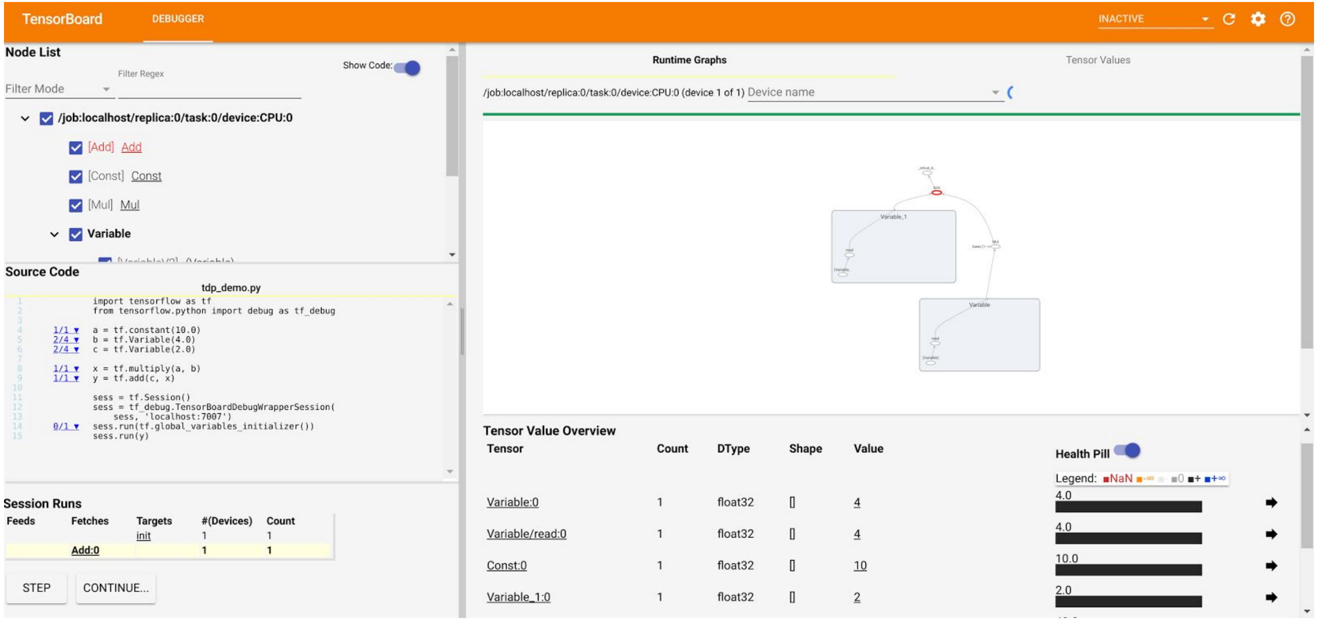
https://www.youtube.com/watch?v=tx6HyoUYGL0

Challenge: Testing

- Training/Test/Validation Datasets
- Unit Tests?
- Different factors
  - Accuracy
  - Serving performance
  - ....
- A/B Testing with live Data



Challenge: Debugging

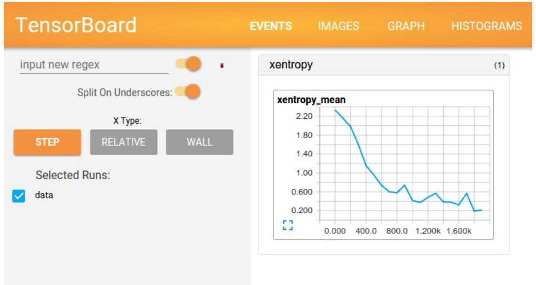


https://www.tensorflow.org/programmers_guide/debugger

Challenge: Monitoring

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

- TensorBoard



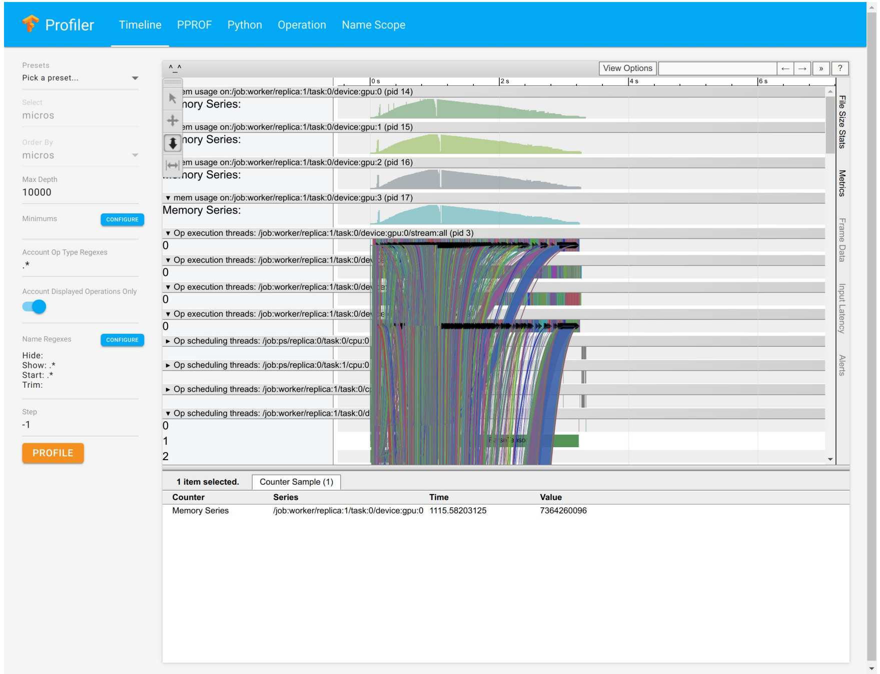
- Traditional Cluster Monitoring Tool

Lab 5: TensorBoard

To CSV mnist

Profiling

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



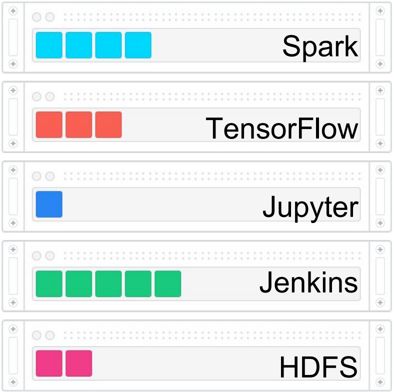
https://www.tensorflow.org/performance/performance_guide

Lab 6 (Optional): TFDebug

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

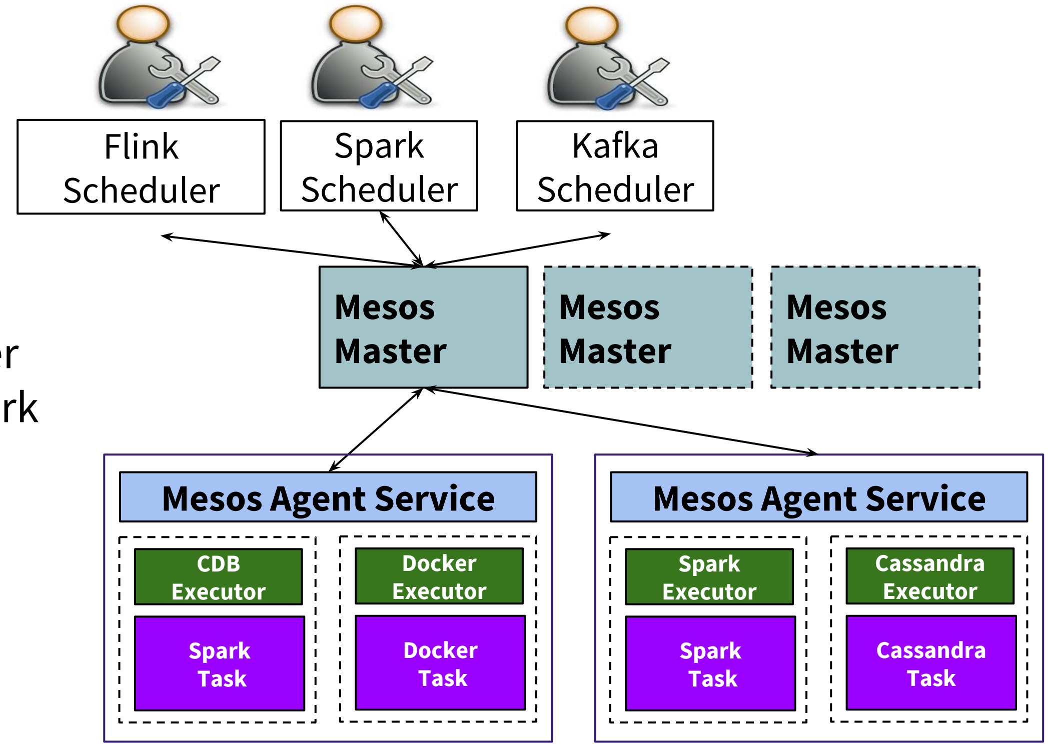
Challenge: Resource and Service Management

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



Typical Datacenter
siloed, over-provisioned servers,
low utilization

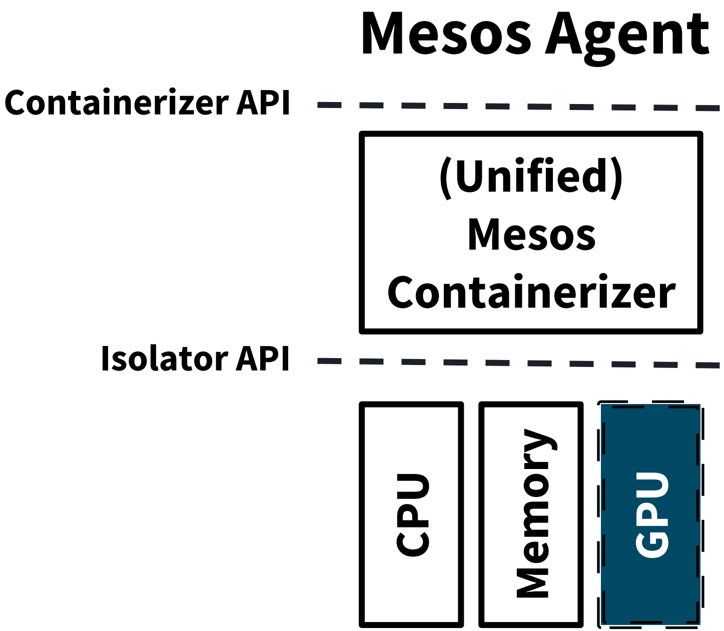
Apache Mesos



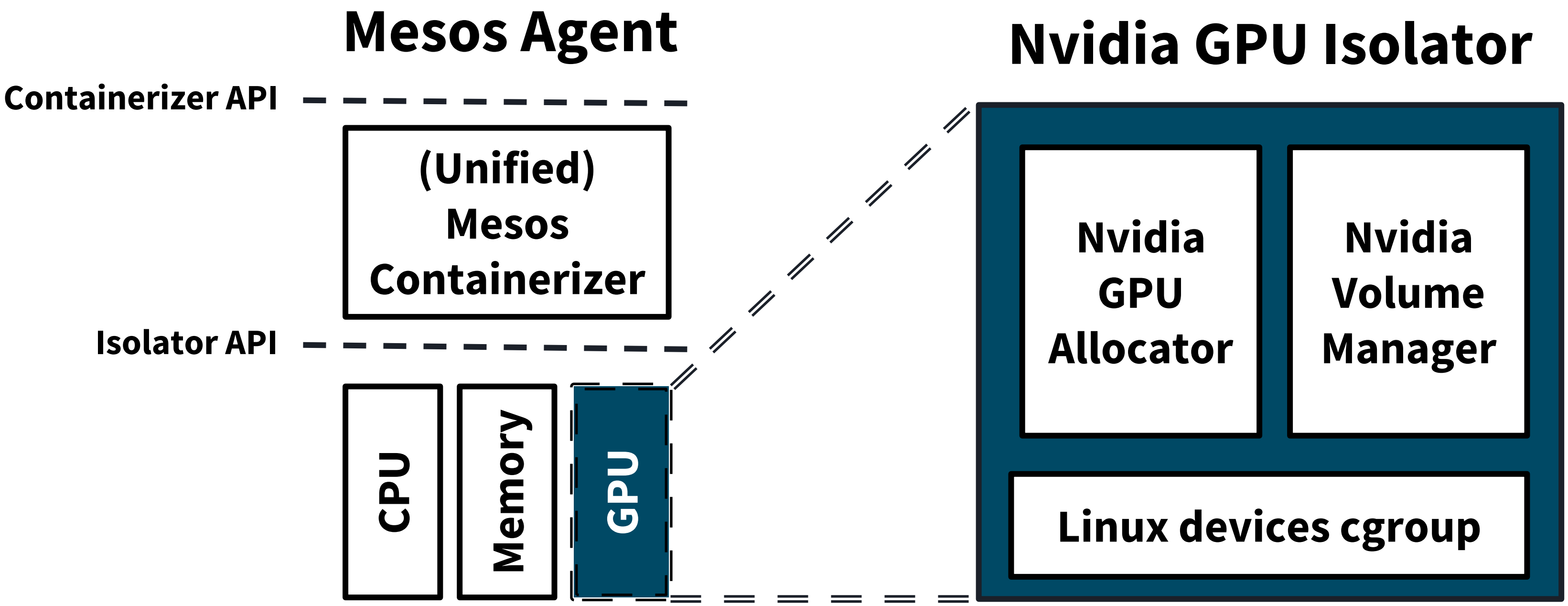
Two-level Scheduling

1. Agents advertise resources to Master
2. Master offers resources to Framework
3. Framework rejects / uses resources
4. Agent reports task status to Master

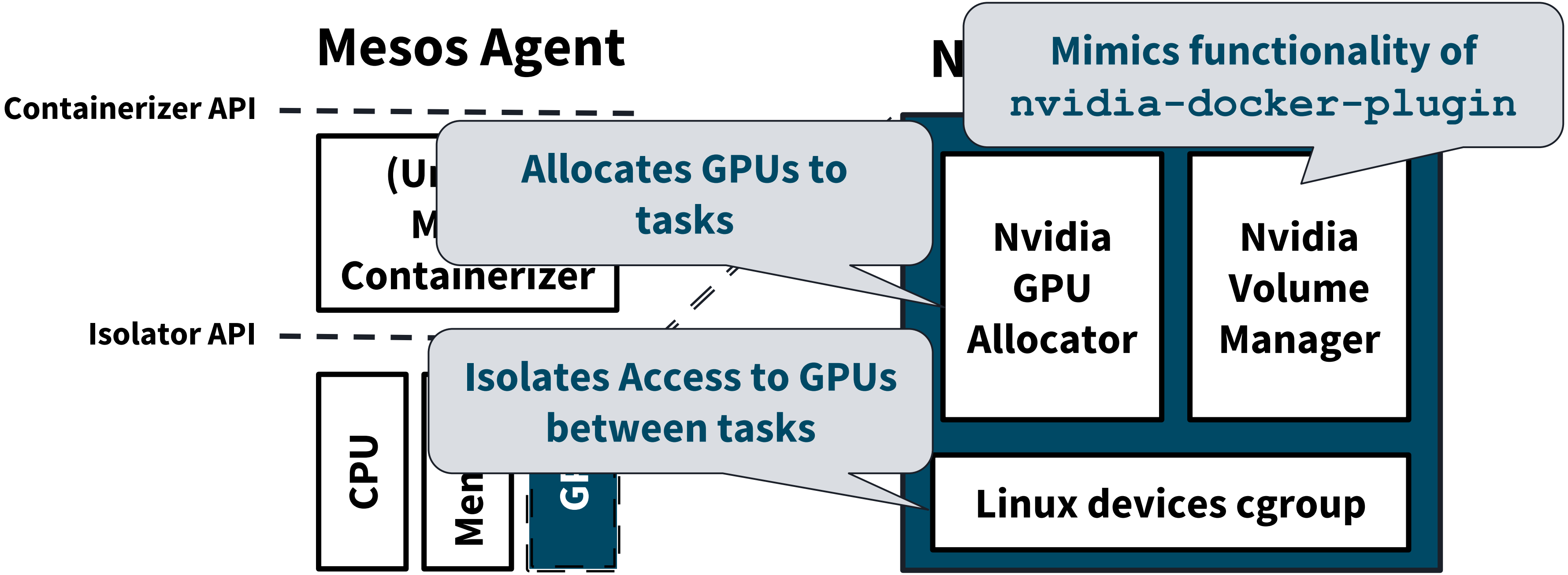
Example: GPU Isolation



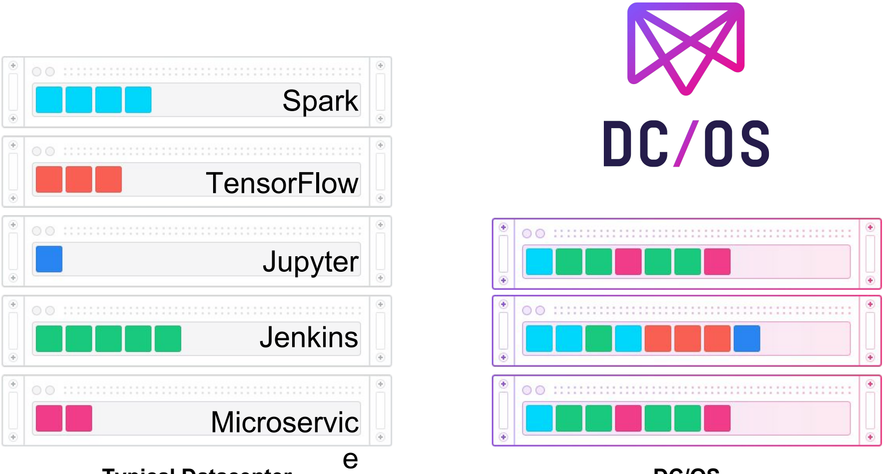
Example: GPU Isolation



Example: GPU Isolation



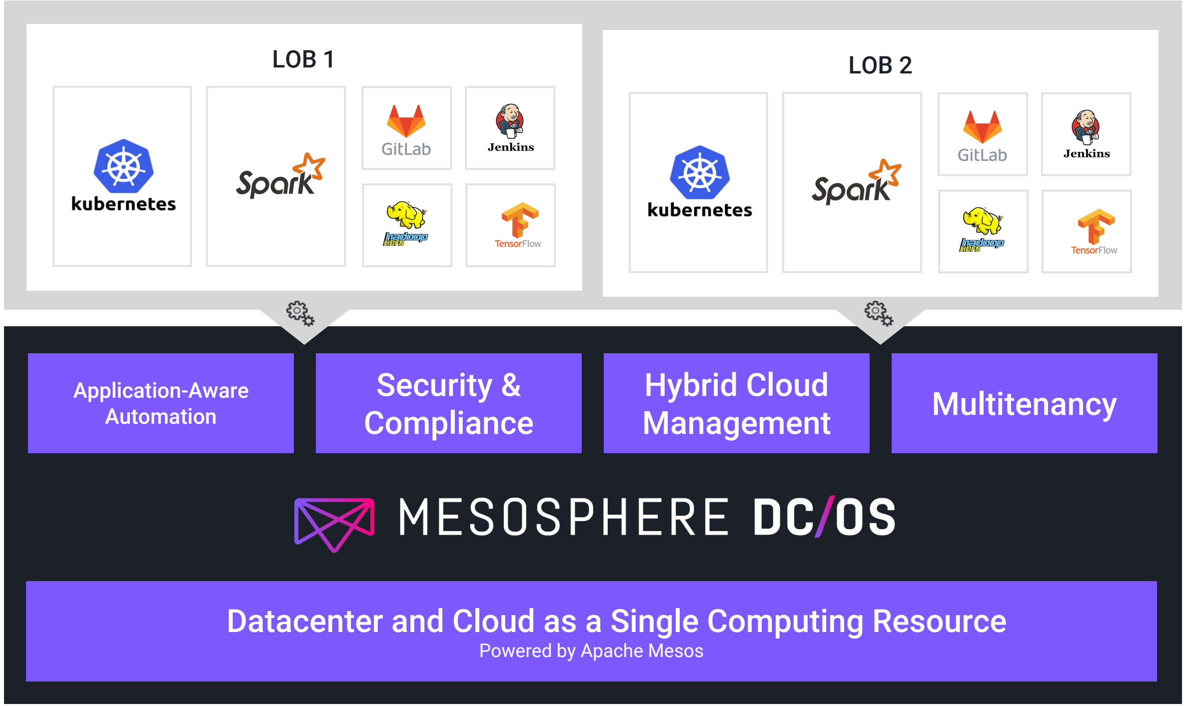
Resource Management



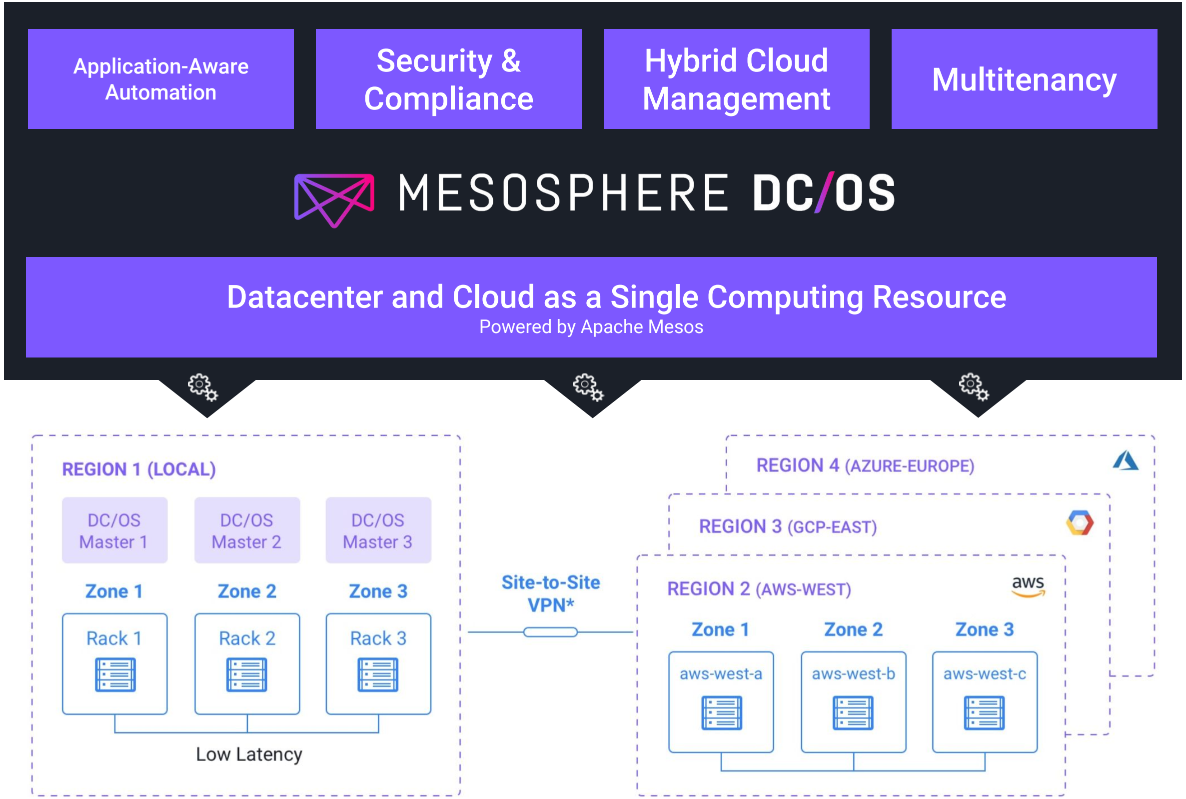
Typical Datacenter
siloed, over-provisioned servers,
low utilization

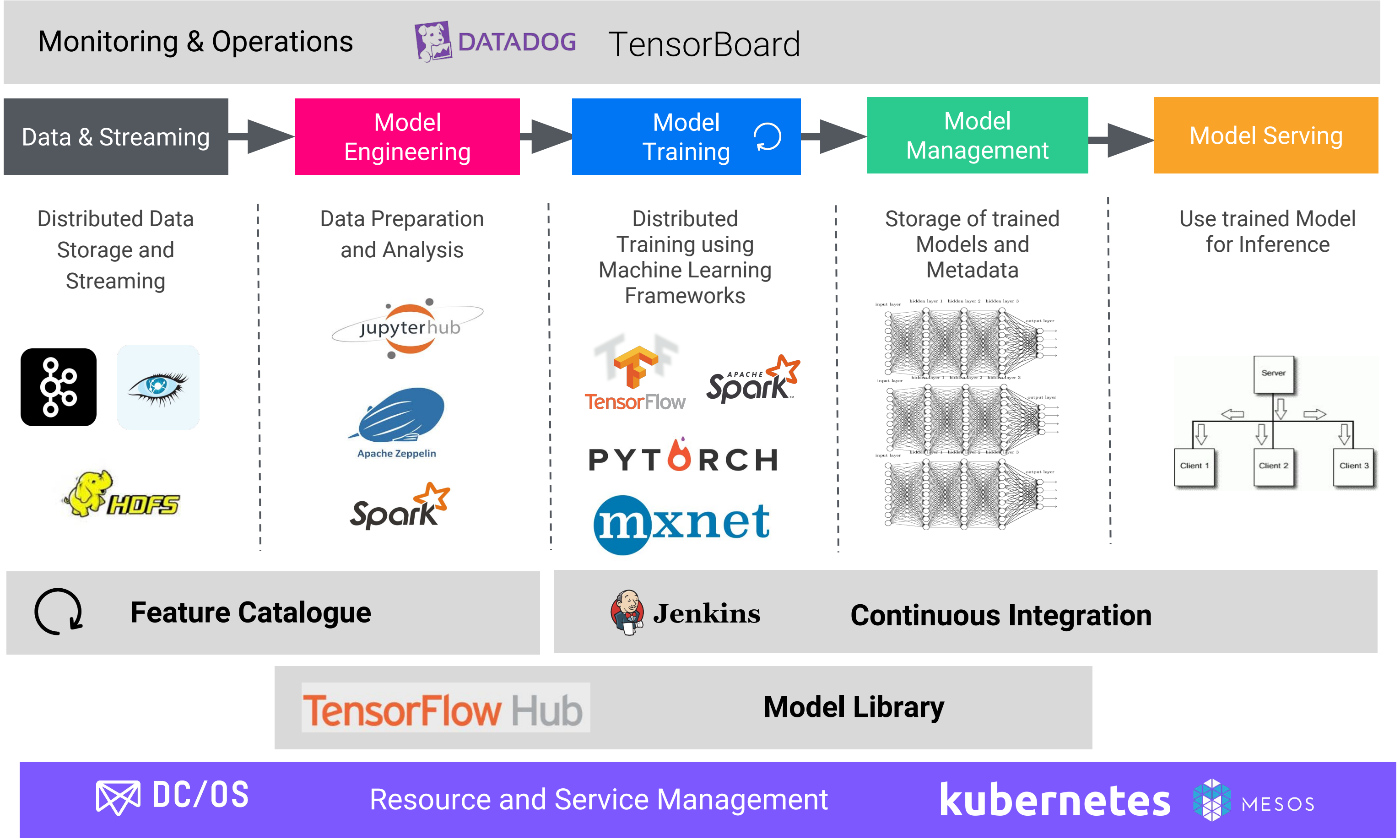
DC/OS
automated schedulers, workload multiplexing onto the same
machines

Service Orchestration



Resource Management




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

@dcos

chat.dcos.io

users@dcos.io

/groups/8295652

/dcos
/dcos/examples
/dcos/demos

THANK YOU!

ANY QUESTIONS?

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

Make it insanely easy to build and scale world-changing technology

MESOSPHERE