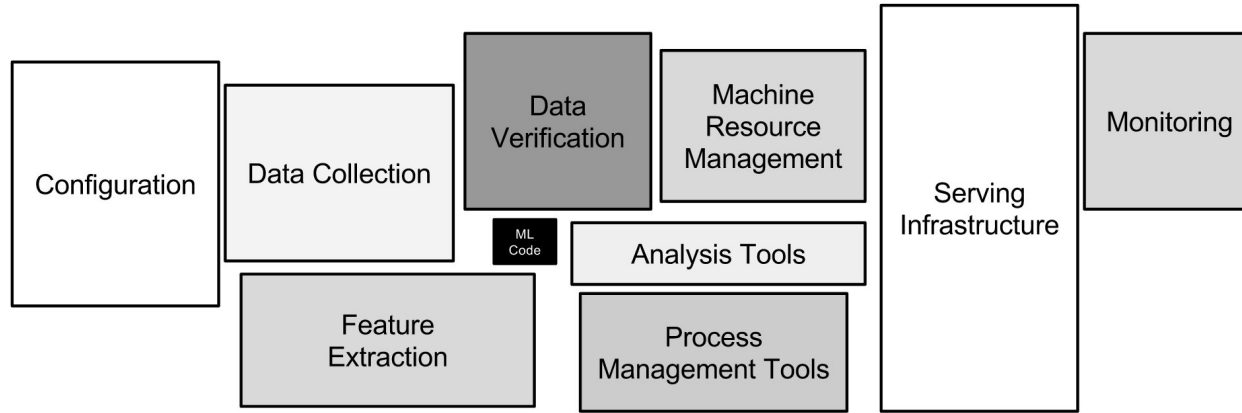


@joerg\_schad #DataSciencePrinciples

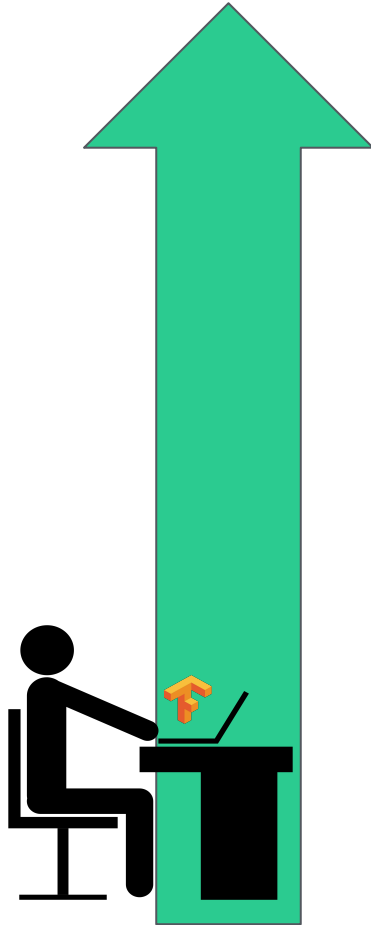
# Building an Open Source Data Science Platform



<https://goo.gl/ZnrR98>



# 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

Data Preparation and Analysis

Distributed Training using Machine Learning Frameworks

Storage of trained Models and Metadata

Use trained Model for Inference

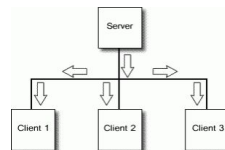
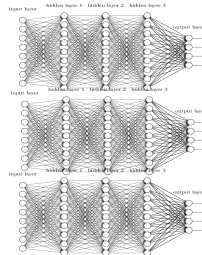


TensorFlow



APACHE SPARK

PYTORCH



Feature Catalogue



Jenkins

Continuous Integration

TensorFlow Hub

Model Library



Resource and Service Management

kubernetes



# Challenge...

Sunday, October 21 • 13:30 - 17:15



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

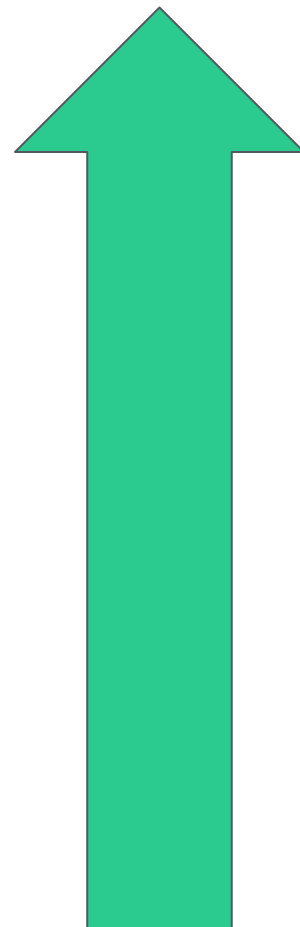
Click here to remove from My Sched.



Who are you?

What is your ML experience?

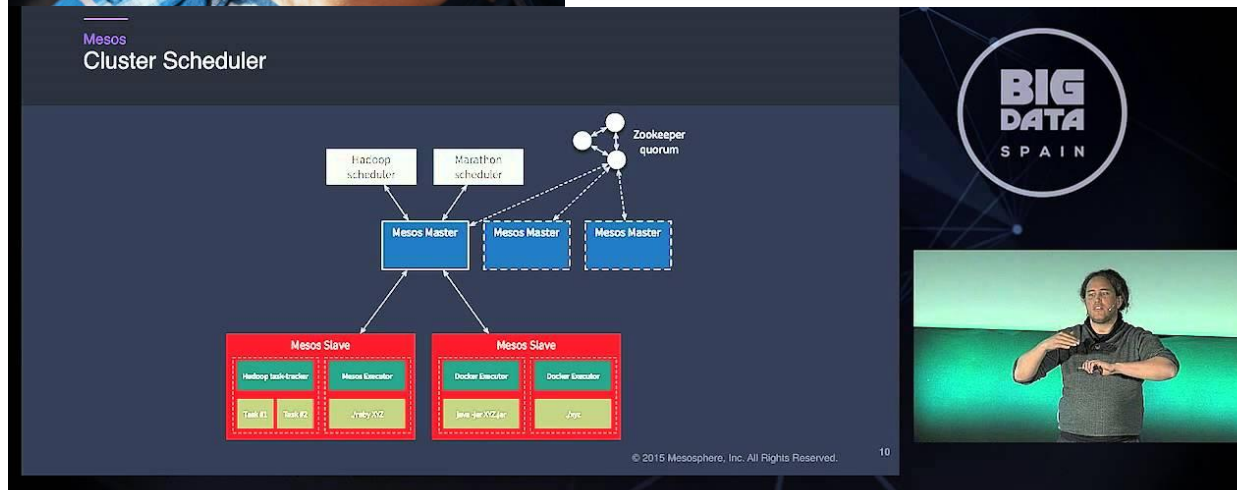
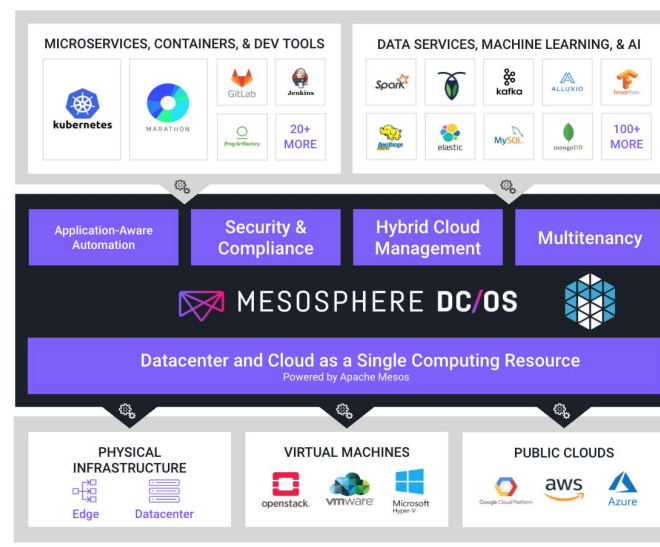
What do you expect to learn today?



# Jörg Schad

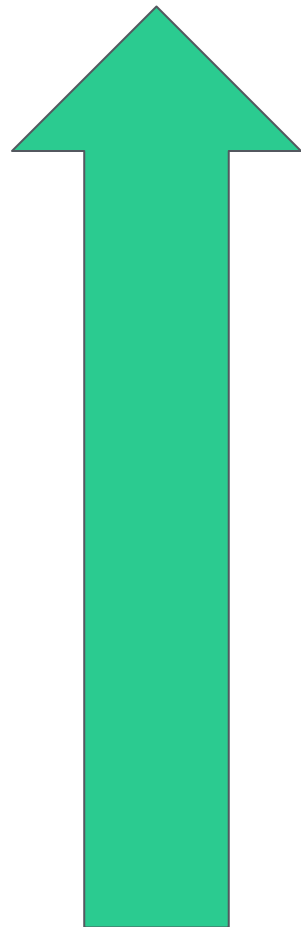
Technical Lead/Engineer  
Deep Learning

- 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?





---

# 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

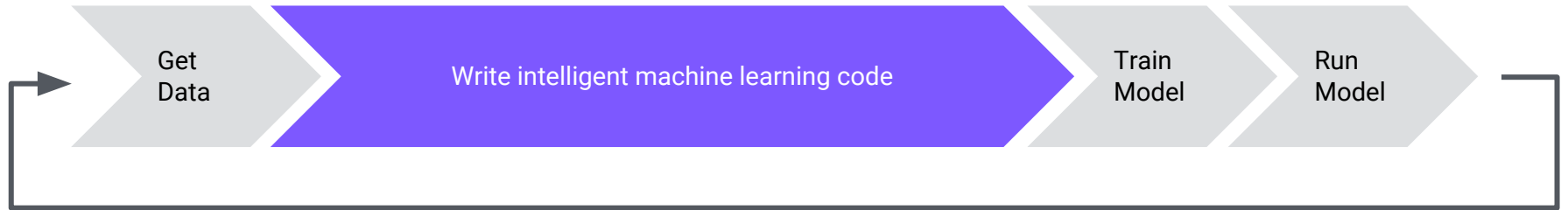




DEEPBACH: A STEERABLE MODEL FOR BACH CHORALES  
GENERATION



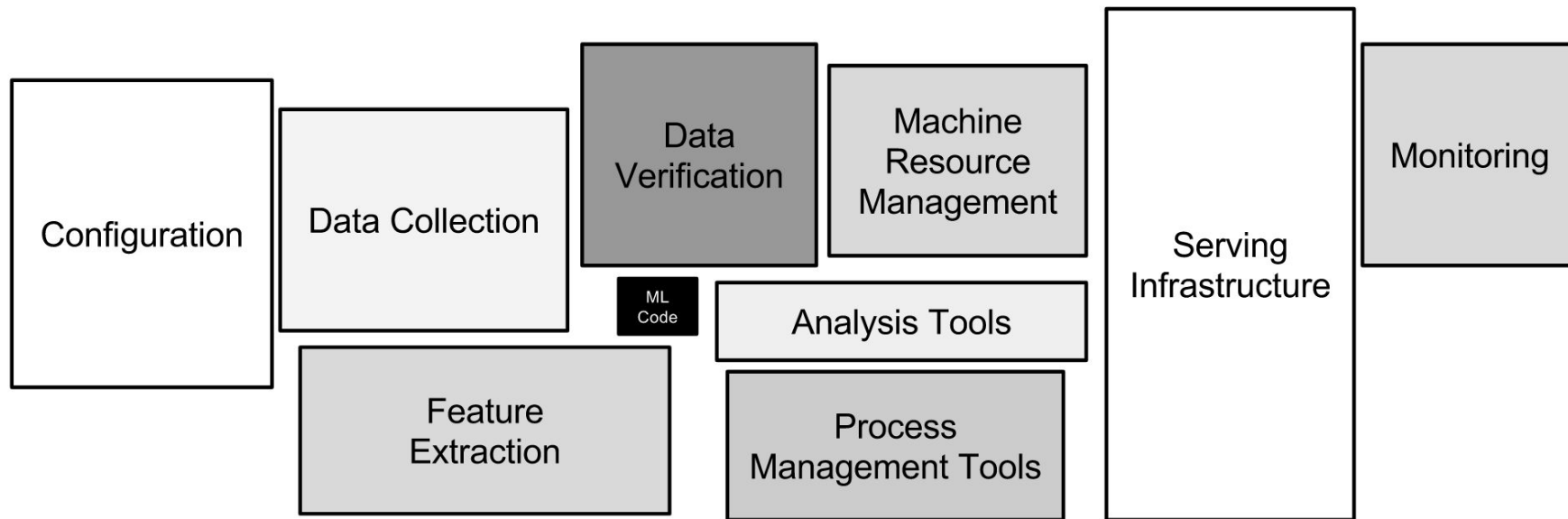
# What you want to be doing



Repeat



## 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



A screenshot of a tweet from Ian Goodfellow (@goodfellow\_ian) replying to @rctatman. The tweet text reads: "I have a different controversial opinion: ML development is a different kind of software development and has a different set of best practices." The tweet is dated 7:59 PM - 18 Sep 2018. The interface includes a profile picture, name, handle, a "Follow" button, and a dropdown arrow.

**Ian Goodfellow**  
@goodfellow\_ian

Following

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

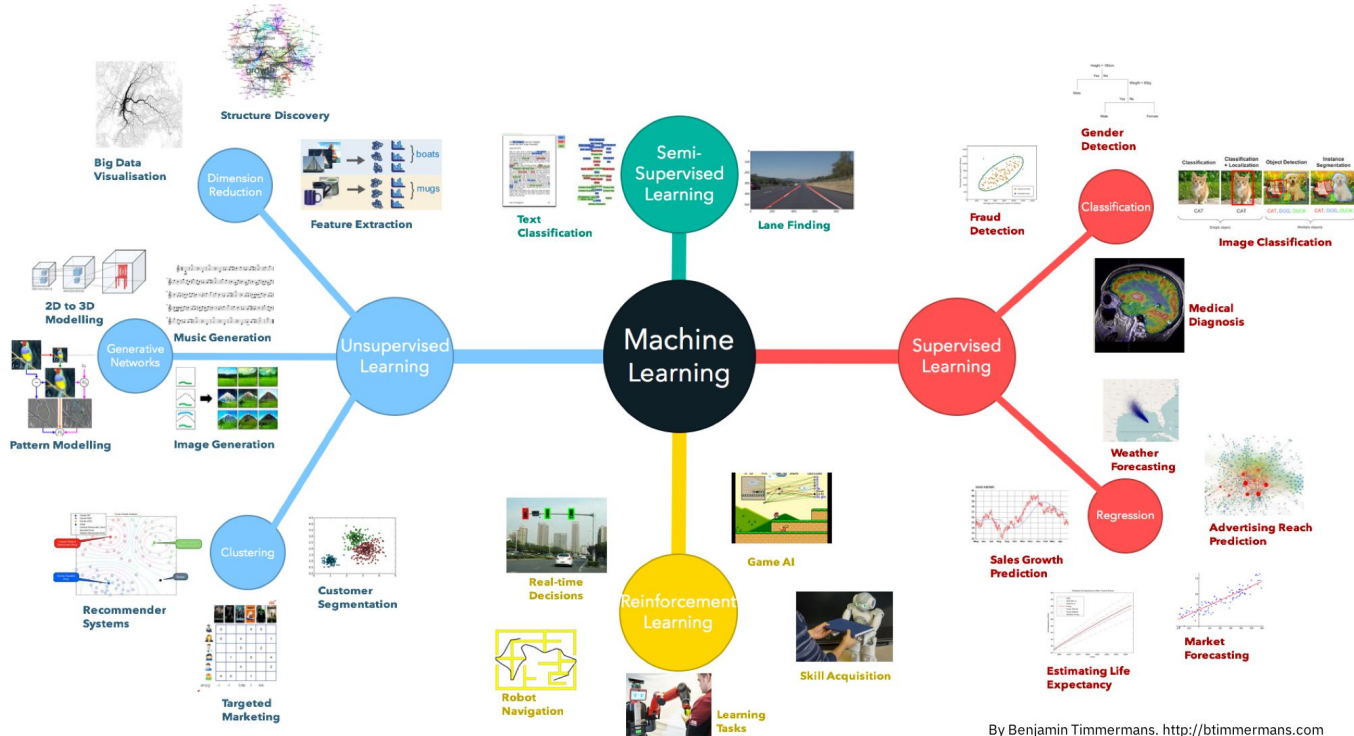
ering

# 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



By Benjamin Timmermans. <http://btimmermans.com>

# Fast.ai

**Could this computer save you**

False Positive Rate	False Negative Rate
66.3%	7.0%
47.5%	0.0%

Chose is good at finding.

It's so good that sometimes our patients are sent home with a clear bill of health.

It's so good that sometimes our patients are sent home with a clear bill of health.

It's so good that sometimes our patients are sent home with a clear bill of health.

Layer 5

Multi-label classification

haze primary      agriculture clear primary water

www.kaggle.com/competitions/covid19 leaderboard

view Data    Kernels    Discussion    Leaderboard    Rules    Taxes    My Submissions    Submit Prediction

Leaderboard    Private Leaderboard

leaderboard is calculated with all of the test data.

Team Name	Kernel	Team Members	Score @	E entries
Thiago			0.99592	9
James Reque			0.99006	2
Mohit Rahman			0.98740	2
Alex Rane   festal			0.98698	10
Abdelhakim Ahmed			0.98532	5
Jeremy Howard			0.98110	4

The neural network to compare words on the fly, and to synthesize their meanings with each other.

Proof. Obvious.

Lemma 0.1. Let  $C$  be a set of the construction. Let  $C$  be a proper covering. Let  $\mathcal{F}$  be a quasi-coherent sheaf on  $C$ .

$$\mathcal{O}_X \otimes \mathcal{F} = \mathcal{O}_X \otimes \mathcal{F}(C)$$

Proof. This is an algebraic space with the composite map.

$$\mathcal{O}_X \otimes \mathcal{F} = \text{morph } \mathcal{O}_X \otimes \mathcal{F} \rightarrow \mathcal{F}$$

where  $\mathcal{G}$  defines an immersion  $\mathcal{F} \rightarrow \mathcal{F}$  of Chow's lemma.

Lemma 0.2. This is an integer  $Z$  is injective.

Proof. See Space, Lemma 77.

Lemma 0.3. Let  $S$  be a scheme. Let  $X$  be a scheme covering. Let  $C$  be a monoidal and locally of fin. Let  $X$  be a scheme which is equal to the formal completion.

The following is the construction of the lemma follows.

Let  $X$  be a scheme. Let  $Z$  be a scheme covering. Let  $i: X \rightarrow Y \rightarrow Y' \rightarrow Y'' \rightarrow X$  be a morphism of algebraic spaces over  $S$  and  $Y$ .

Proof. Let  $X$  be a monoidal scheme of  $X$ . Let  $X$  be an open covering of  $X$ .

(1)  $\mathcal{F}$  is an algebraic space over  $S$ .

(2) If  $X$  is an affine open covering.

Consider a monoidal structure on  $X$  and  $X$  the finite bute type.

begin[proof] We may assume that  $\mathcal{S}(\text{mathcal{F}})$  is a  $\mathcal{S}(\text{mathcal{F}})$ - $\mathcal{S}$ -bim. Given a morphism  $\mathcal{S}(\text{mathcal{F}})$  is an injective and let  $\mathcal{S}(\text{mathcal{F}})$  be an abelian at  $\mathcal{S}(\text{mathcal{F}})$  is a filtered complex. Let  $\mathcal{S}(\text{mathcal{F}})$

char 4 using chars 1, 2 & 3

Hidden

Output

Hidden



# Deep Learning: The Promise

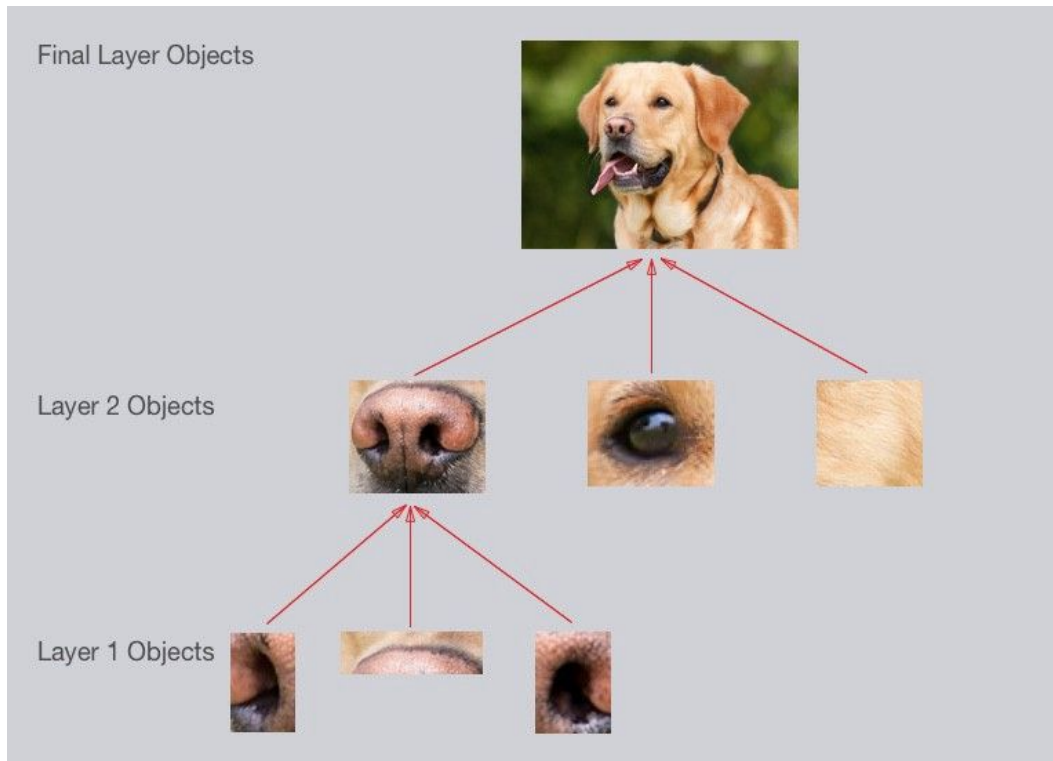
## TRADITIONAL MACHINE LEARNING



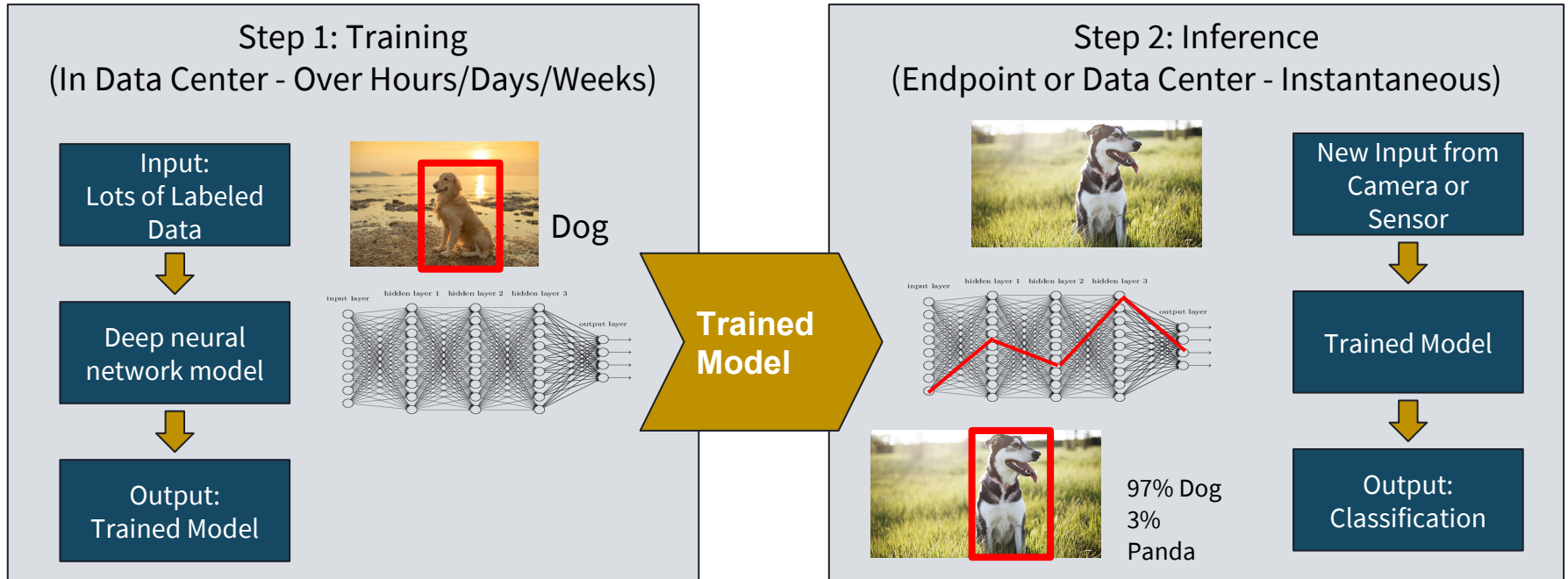
## DEEP LEARNING



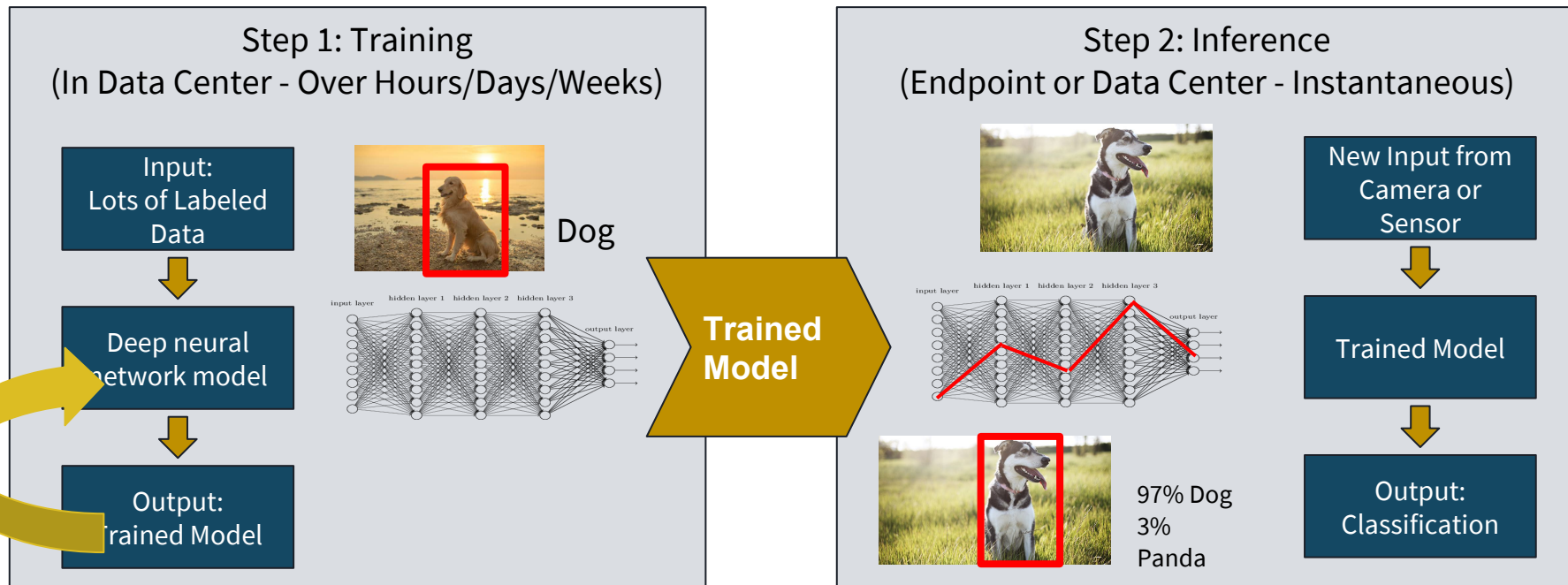
# Deep Learning: Insights



# Deep Learning: The Process



# Deep Learning: The Process



## Challenge: Persona(s)



# The Rise of the *DataOps Engineer*

Combines two key skills:

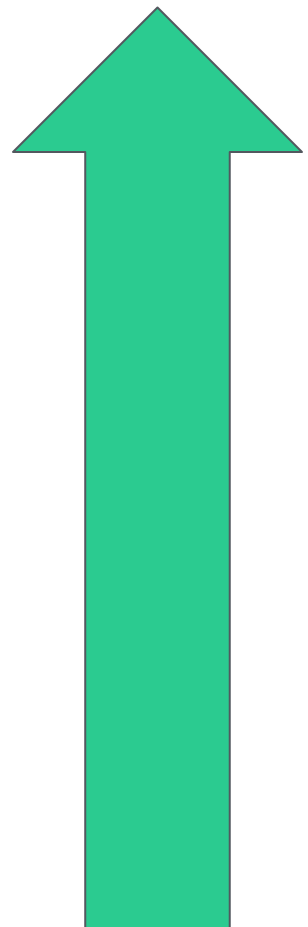
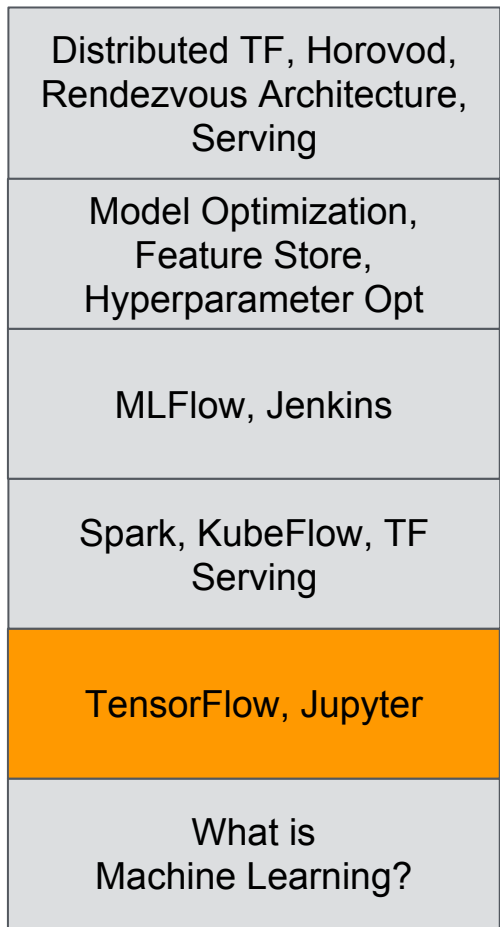
- Data science
- Distributed systems engineering

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



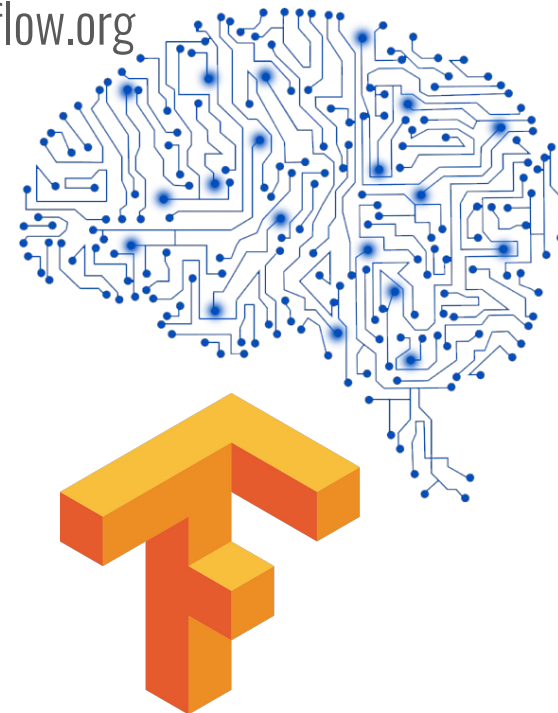
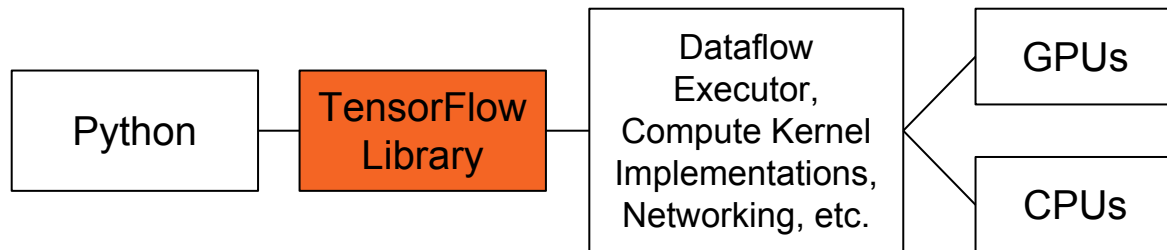




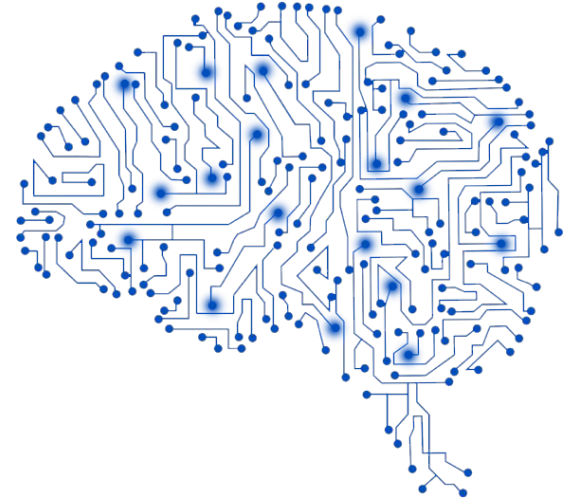
# TensorFlow Overview

“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]
```

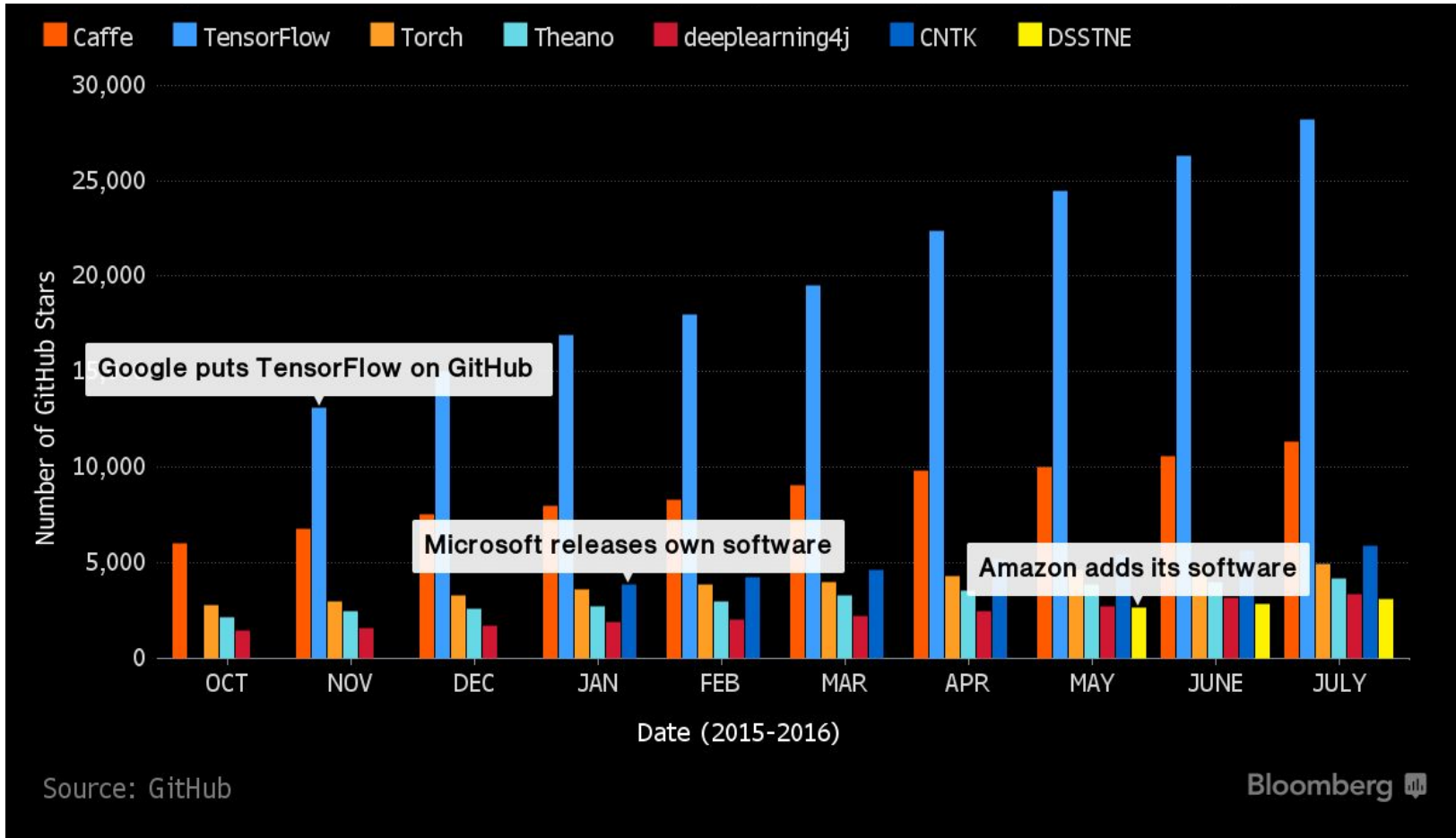
# Alternatives



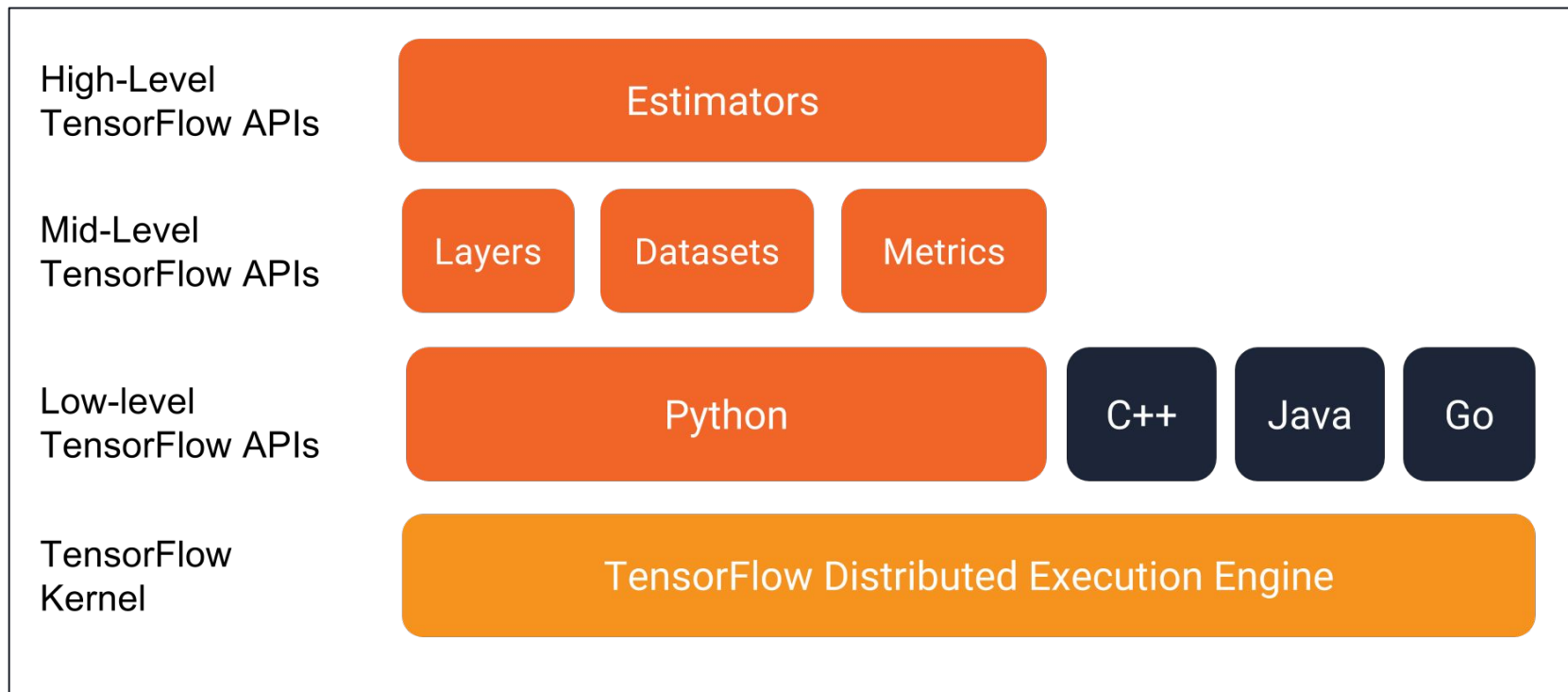
```
import tensorflow as tf
import numpy as np
X = tf.placeholder("float")
Y = tf.placeholder("float")
W = tf.Variable(np.random.random(), name="weight")
pred = tf.multiply(X, W)
cost = tf.reduce_sum(tf.pow(pred-Y, 2))
optimizer = tf.train.GradientDescentOptimizer
init = tf.global_variables_initializer()
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
```

```
tf.enable_eager_execution()
```

```
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 = optim.SGD(model.parameters(), lr=0.01)
for t in range(10000):
    x = torch.from_numpy(np.random.random((1,1)).astype(np.float32))
    y = torch.from_numpy(np.random.random((1,1)).astype(np.float32))
    y_pred = model(x)
    loss = loss_fn(y_pred, y)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    print loss.data[0]
```



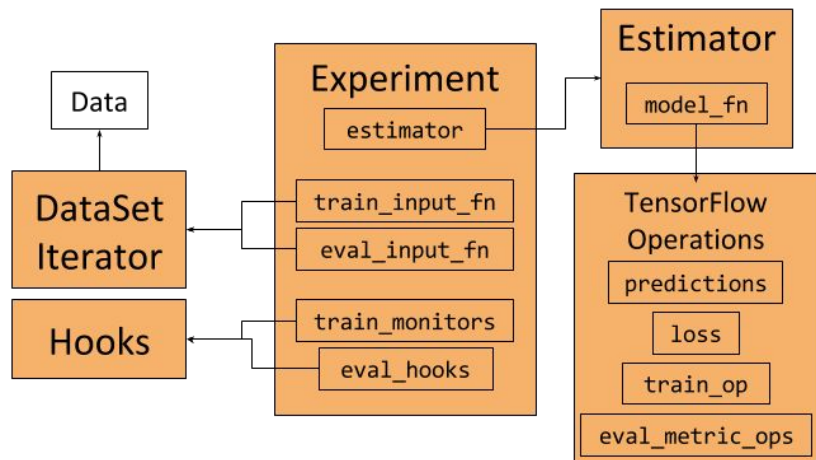
# Challenge: Writing Distributed Model Functions



# TensorFlow Estimator & Keras APIs

- Preferred APIs

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



# Lab 0: Fashion Mnist



TensorFlow™ Install **Develop** Community API Ecosystem

Develop

TUTORIALS GUIDE DEPLOY PERFORMANCE EXTEND

Get started with TensorFlow

Learn and use ML <sup>^</sup>

- Overview
- Basic classification**
- Text classification
- Regression
- Overfitting and underfitting
- Save and restore models



Research and experimentation <sup>^</sup>

ML at production scale <sup>^</sup>

Generative models <sup>^</sup>

## Train your first neural network: basic classification

☆☆☆☆☆

 [Run in Google Colab](#)  [View source on GitHub](#)

This guide trains a neural network model to classify images of clothing, like sneakers and shirts. It's okay if you don't understand all the details, this is a fast-paced overview of a complete TensorFlow program with the details explained as we go.

This guide uses [tf.keras](#), a high-level API to build and train models in TensorFlow.

```
# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras
```

[https://www.tensorflow.org/tutorials/keras/basic\\_classification](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



**Your connection is not private**

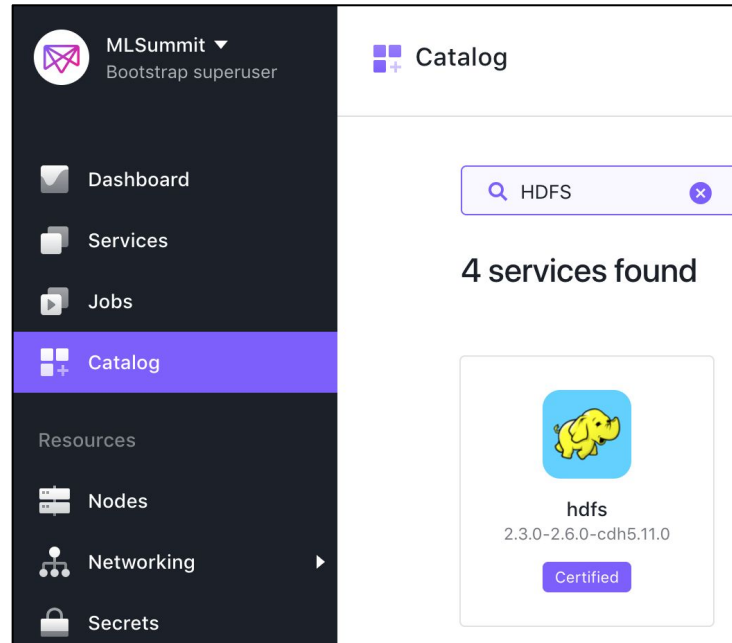
Attackers might be trying to steal your information from **34.211.62.66** (for example, passwords, messages or credit cards). [Learn more](#)

NET::ERR\_CERT\_AUTHORITY\_INVALID

Automatically send some [system information and page content](#) to Google to help detect dangerous apps and sites. [Privacy Policy](#)

ADVANCED [Back to safety](#)

# Lab 1: HDFS



The screenshot displays the DCOS Catalog interface. On the left is a dark sidebar with navigation options: Dashboard, Services, Jobs, Catalog (highlighted in purple), Resources, Nodes, Networking, and Secrets. The top of the sidebar shows the user 'MLSummit' as a 'Bootstrap superuser'. The main content area is titled 'Catalog' and features a search bar with the query 'HDFS'. Below the search bar, it states '4 services found'. A single service card is visible, featuring a yellow elephant icon on a blue background, the name 'hdfs', the version '2.3.0-2.6.0-cdh5.11.0', and a purple 'Certified' badge.

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

# Lab 1: Marathon-LB

The screenshot displays the DCOS Catalog interface. On the left is a dark sidebar with navigation options: MLSummit (Bootstrap superuser), Dashboard, Services, Jobs, Catalog (highlighted in purple), Resources, Nodes, and Networking. The main content area shows the 'marathon-lb' service page. At the top, it says 'Catalog > marathon-lb'. The service card features a blue circular icon with three white squares and arrows, the name 'marathon-lb', version '1.12.3', and a 'Community' tag. A purple 'Review & Run' button is in the top right. Below the card is a 'Description' section with the text 'HAProxy configured using Marathon state'. A yellow 'Preinstall Notes' box contains the following text: 'Preinstall Notes: We recommend at least 2 CPUs and 1GiB of RAM for each Marathon-LB instance. NOTE: For additional Enterprise Edition DC/OS instructions, see <https://docs.mesosphere.com/administration/id-and-access-mgt/service-auth/mlb-auth/>.'

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

# Lab 1: Jupyter

Cancel Edit Configuration  
Jupyterlab 1.2.0-0.33.7

---

Service  
Oidc  
S3  
Spark  
Storage  
**Networking**  
Environment

**Networking**  
DC/OS JupyterLab networking configuration properties

**Cni Support**  
Enable Container Networking Interface (CNI) Support.  
 enabled ?

**External Access**  
Enable access from outside the cluster through Marathon-LB. NOTE: this connection is unencrypted.  
 enabled ?

external public agent hostname \* ?

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

# Lab 1: Jupyter

### Edit Configuration

Jupyterlab 1.2.0-0.33.7

-server -XX-+USEGITOC -XX-+heapDumpOnOut

- Service
- Oidc
- S3
- Spark
- Storage
- Networking
- Environment**

**jupyter conf urls** ?

**http://api.hdfs.marathon.141b.thisdcos.directory/v1/endpoints**

**jupyter config dir** ?

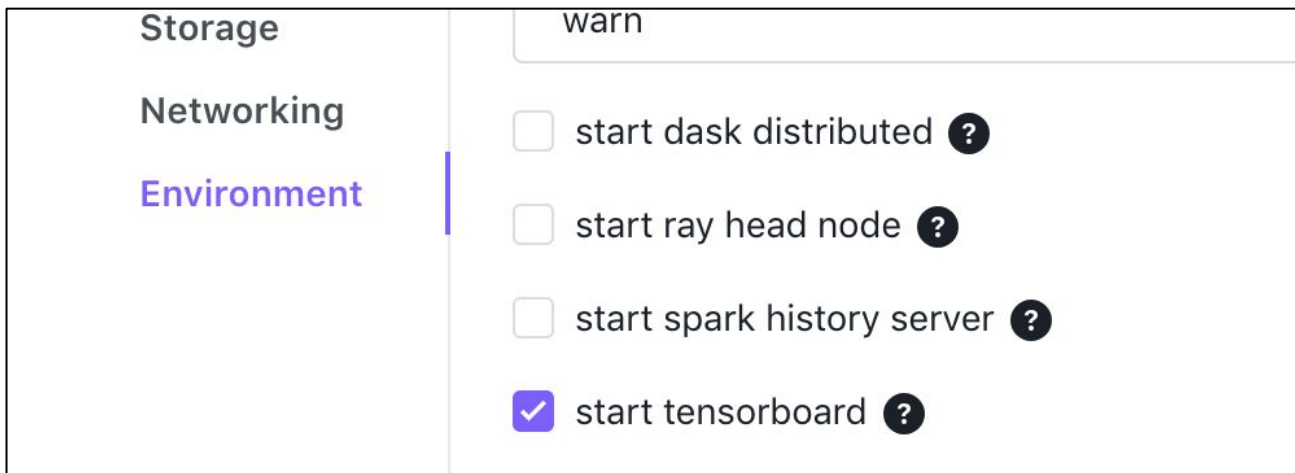
**jupyter password** ?

**jupyter runtime dir** ?

<http://api.hdfs.marathon.141b.thisdcos.directory/v1/endpoints>

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

# Lab 1: Jupyter

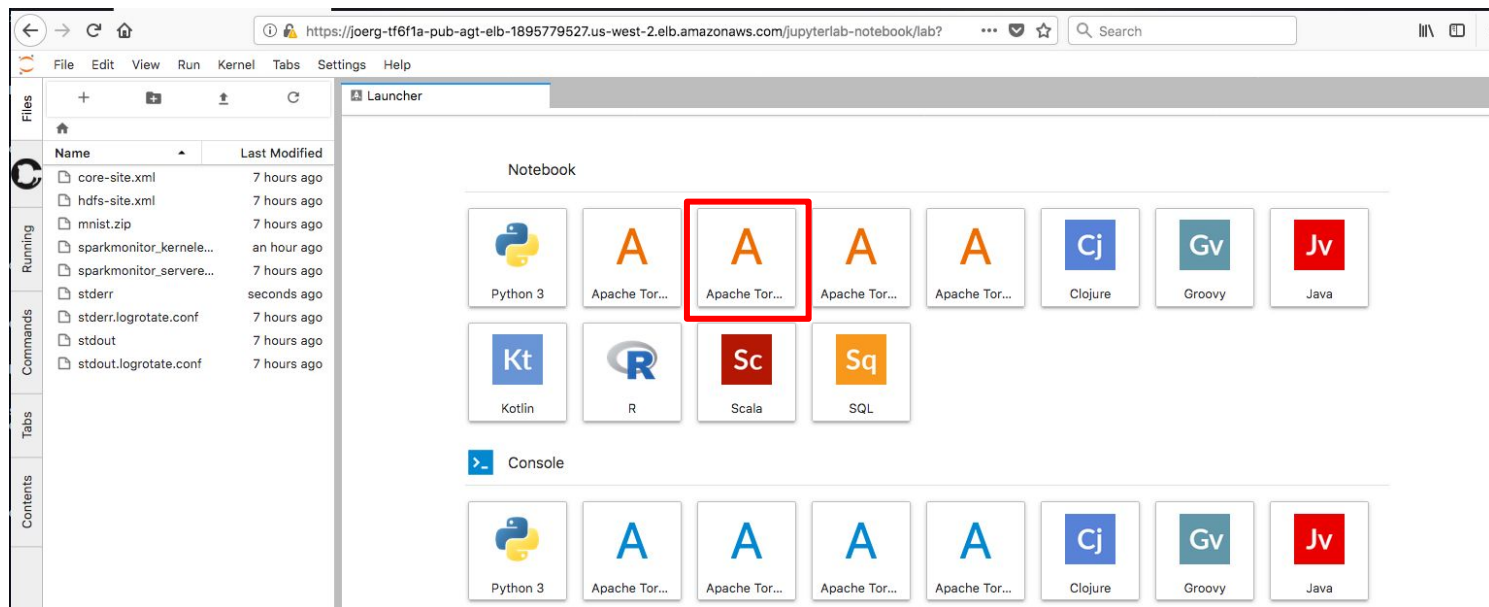


The screenshot shows the JupyterLab configuration interface. On the left, a sidebar contains three menu items: "Storage", "Networking", and "Environment". The "Environment" item is highlighted with a blue vertical bar. The main content area on the right is titled "warn" and contains a list of four options, each with a checkbox and a help icon (a question mark in a circle):

- start dask distributed ?
- start ray head node ?
- start spark history server ?
- start tensorboard ?

# Lab 1: Jupyter

<External>/jupyterlab-notebook  
PW: jupyter



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

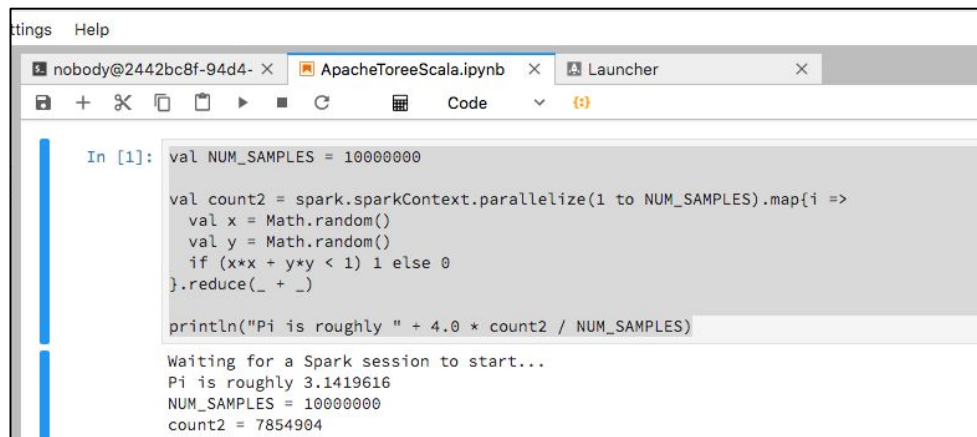


# Lab 1: Jupyter

```
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)
```



The screenshot shows a Jupyter Notebook window titled "ApacheToreeScala.ipynb". The code in the notebook is as follows:

```
In [1]: 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(_ + _)

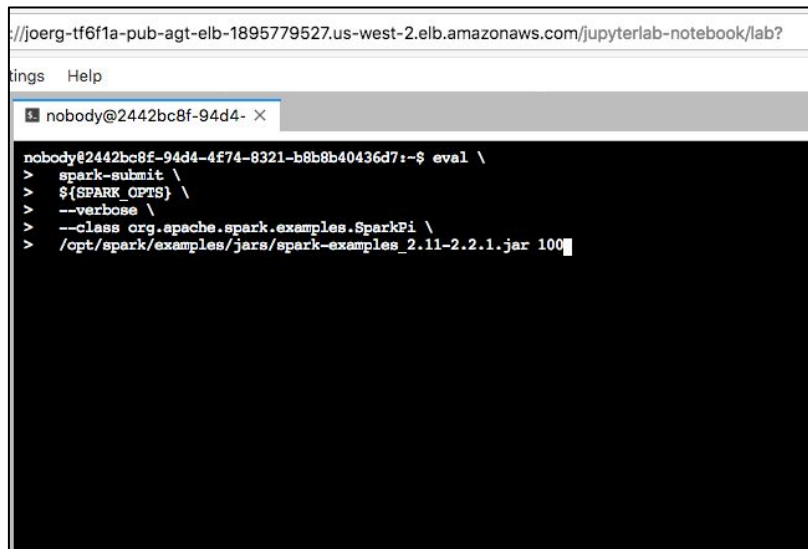
println("Pi is roughly " + 4.0 * count2 / NUM_SAMPLES)
```

The output of the code is:

```
Waiting for a Spark session to start...
Pi is roughly 3.1419616
NUM_SAMPLES = 10000000
count2 = 7854904
```

# Lab 1: Jupyter

```
eval \  
  spark-submit \  
  ${SPARK_OPTS} \  
  --verbose \  
  --class  
org.apache.spark.examples.SparkPi \  
  
/opt/spark/examples/jars/spark-examples_2.11-2.2.1.jar 100
```



The screenshot shows a JupyterLab interface with a terminal window. The terminal prompt is `nobody@2442bc8f-94d4-4f74-8321-b8b8b40436d7:~$`. The user has entered the following command:

```
eval \  
  spark-submit \  
  ${SPARK_OPTS} \  
  --verbose \  
  --class org.apache.spark.examples.SparkPi \  
  /opt/spark/examples/jars/spark-examples_2.11-2.2.1.jar 100
```

The terminal output shows the command being executed with a cursor at the end of the last line.

# Lab 1: Jupyter

The screenshot displays the Apache Mesos web interface for a cluster named 'GPU\_Demo'. The top navigation bar includes 'Frameworks', 'Agents', 'Roles', 'Offers', and 'Maintenance'. The current page is 'Active Tasks', which shows a table of running tasks. On the left, there is a sidebar with cluster details and a list of agents.

**Cluster: GPU\_Demo**  
Leader: 10.0.2.65:5050  
Version: 1.5.0  
Built: 3 months ago by  
Started: 9 hours ago  
Elected: 9 hours ago

**Agents**

Activated	5
Deactivated	0
Unreachable	0

**Tasks**

Staging	5
---------	---

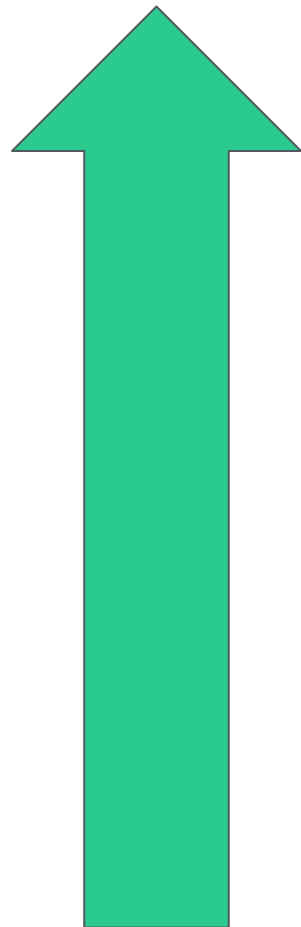
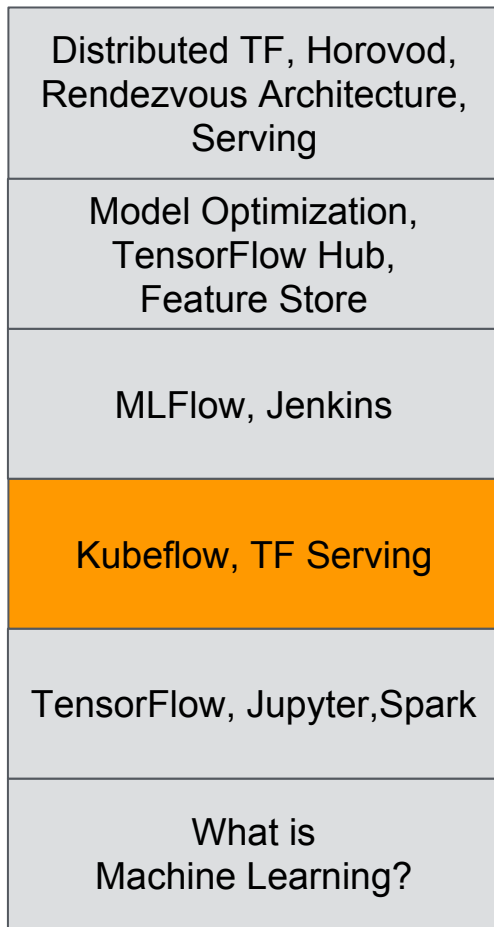
**Active Tasks**

Framework ID	Task ID	Task Name	Role	State	Health	Started	Host	
...e512ac82c5c4-0007	4	Spark Pi 4	*	RUNNING	-	just now	10.0.4.225	Sandbox
...e512ac82c5c4-0007	2	Spark Pi 2	*	RUNNING	-	just now	10.0.4.225	Sandbox
...e512ac82c5c4-0007	0	Spark Pi 0	*	RUNNING	-	just now	10.0.4.225	Sandbox
...e512ac82c5c4-0007	3	Spark Pi 3	*	RUNNING	-	just now	10.0.4.239	Sandbox
...e512ac82c5c4-0007	1	Spark Pi 1	*	RUNNING	-	just now	10.0.4.239	Sandbox
...e512ac82c5c4-0001	jupyterlab-notebook.b8df434b-9a5f-11e8-a9c2-461a4e785879	jupyterlab-notebook	slave_public	RUNNING	Healthy	8 hours ago	10.0.4.225	Sandbox

<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

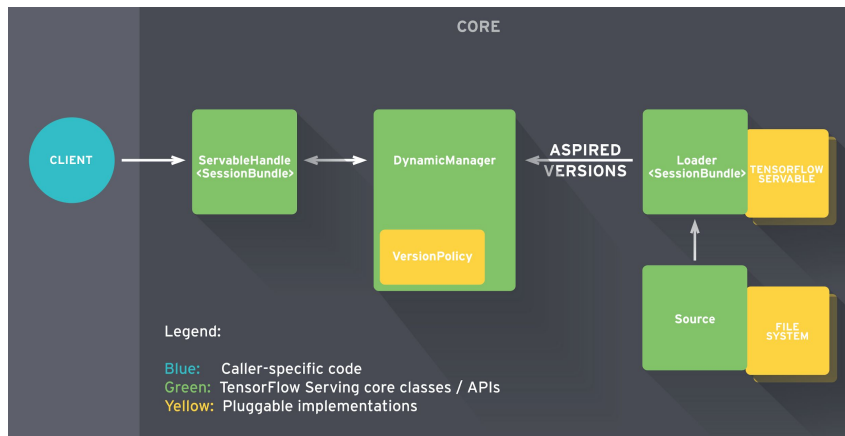




# Challenge: Serving

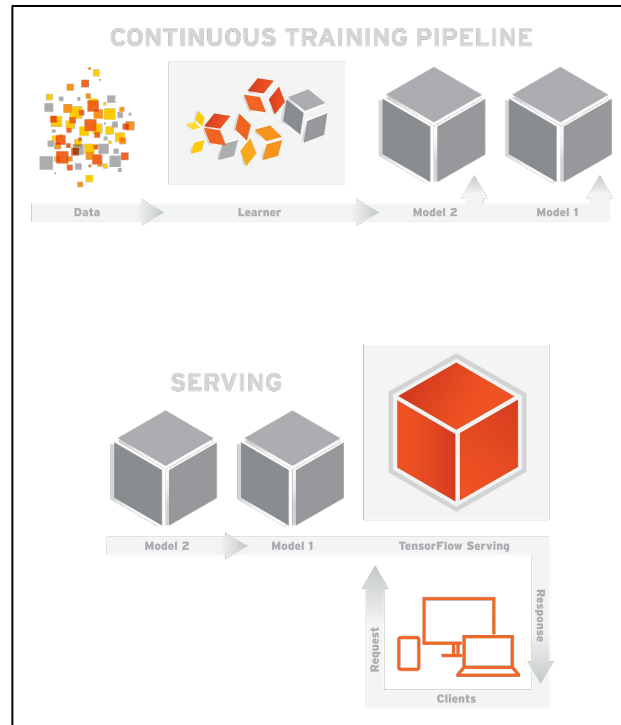
- How to Deploy Models?
  - Zero Downtime
  - Canary
- Multiple Models?
  - Testing

- 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_basic)

[https://www.tensorflow.org/serving/serving\\_advanced](https://www.tensorflow.org/serving/serving_advanced)



Pipeline.ai



<https://pipeline.ai/>

# Pipeline.ai

Train Evaluate Compare

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



The screenshot displays the Pipeline.ai interface with three prediction results for the digit '2'. Each result includes a table of digit and confidence values and a corresponding image of the digit.

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



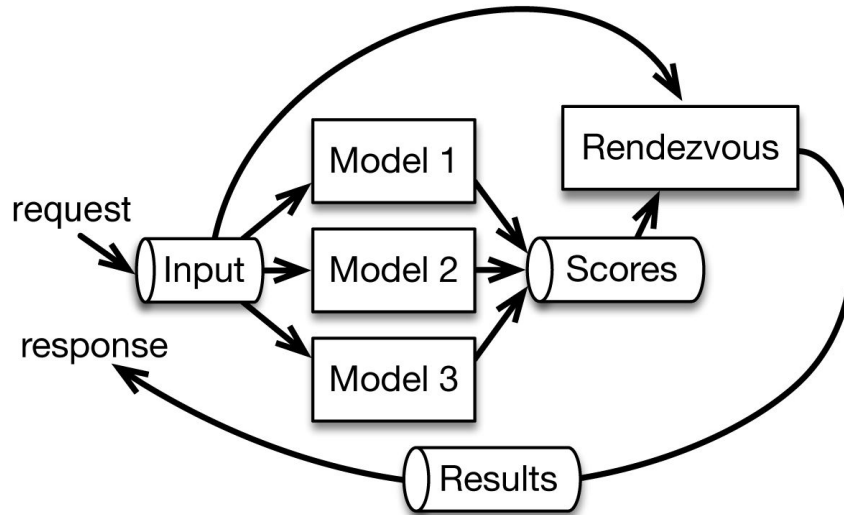
Digit	Confidence
0	0.0022526539396494627
1	2.63791100074684e-10
2	0.4638307988643646
3	0.21909376978874207
4	3.2985670372909226e-07
5	0.29357224702835083
6	0.00019597385835368186
7	5.230629176367074e-05
8	0.020996594801545143
9	5.426473762781825e-06

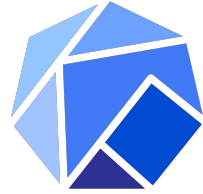


Digit	Confidence
0	0.0022526539396494627
1	2.63791100074684e-10
2	0.4638307988643646
3	0.21909376978874207
4	3.2985670372909226e-07
5	0.29357224702835083
6	0.00019597385835368186
7	5.230629176367074e-05
8	0.020996594801545143
9	5.426473762781825e-06



# Rendezvous Architecture





# Kubeflow



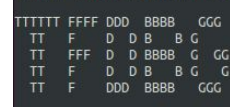
1. Data Preparation  
& Model Engineering



2. Model Training

TensorBoard

3. Monitoring

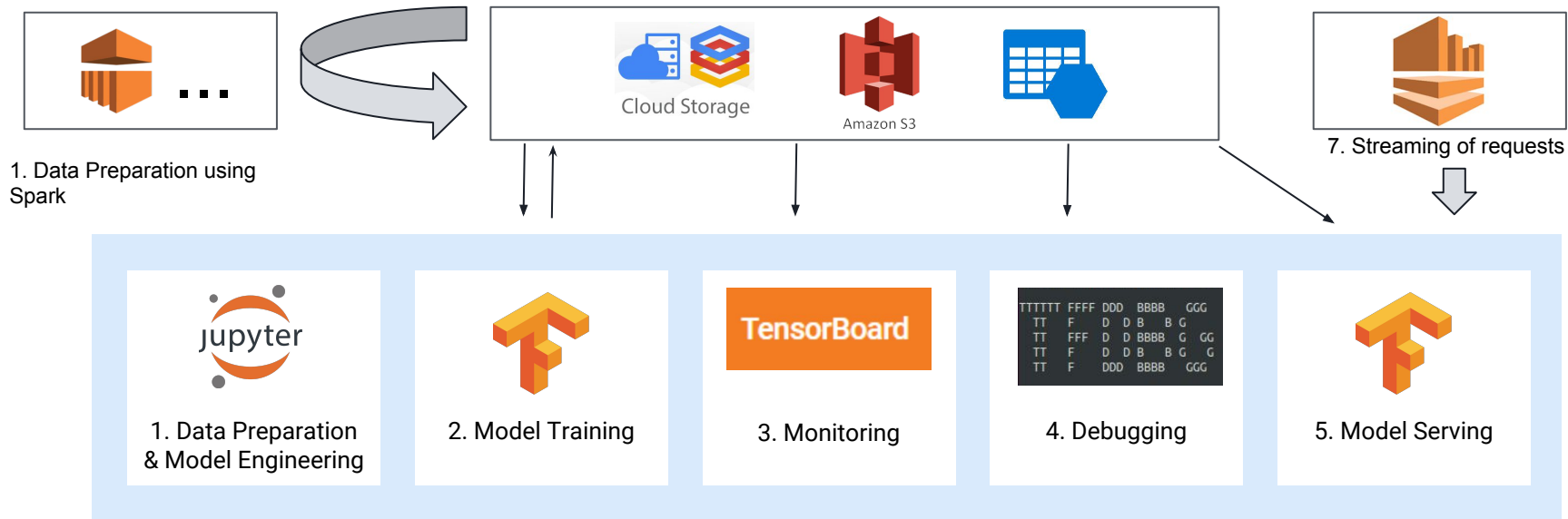


4. Debugging

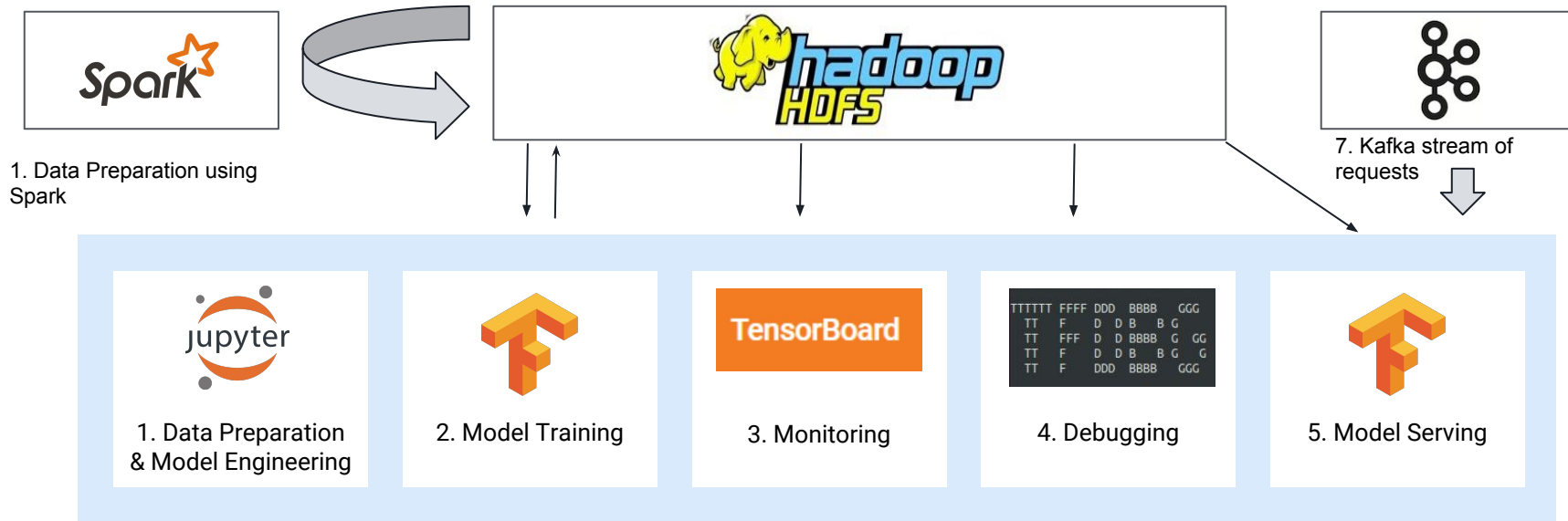


5. Model Serving

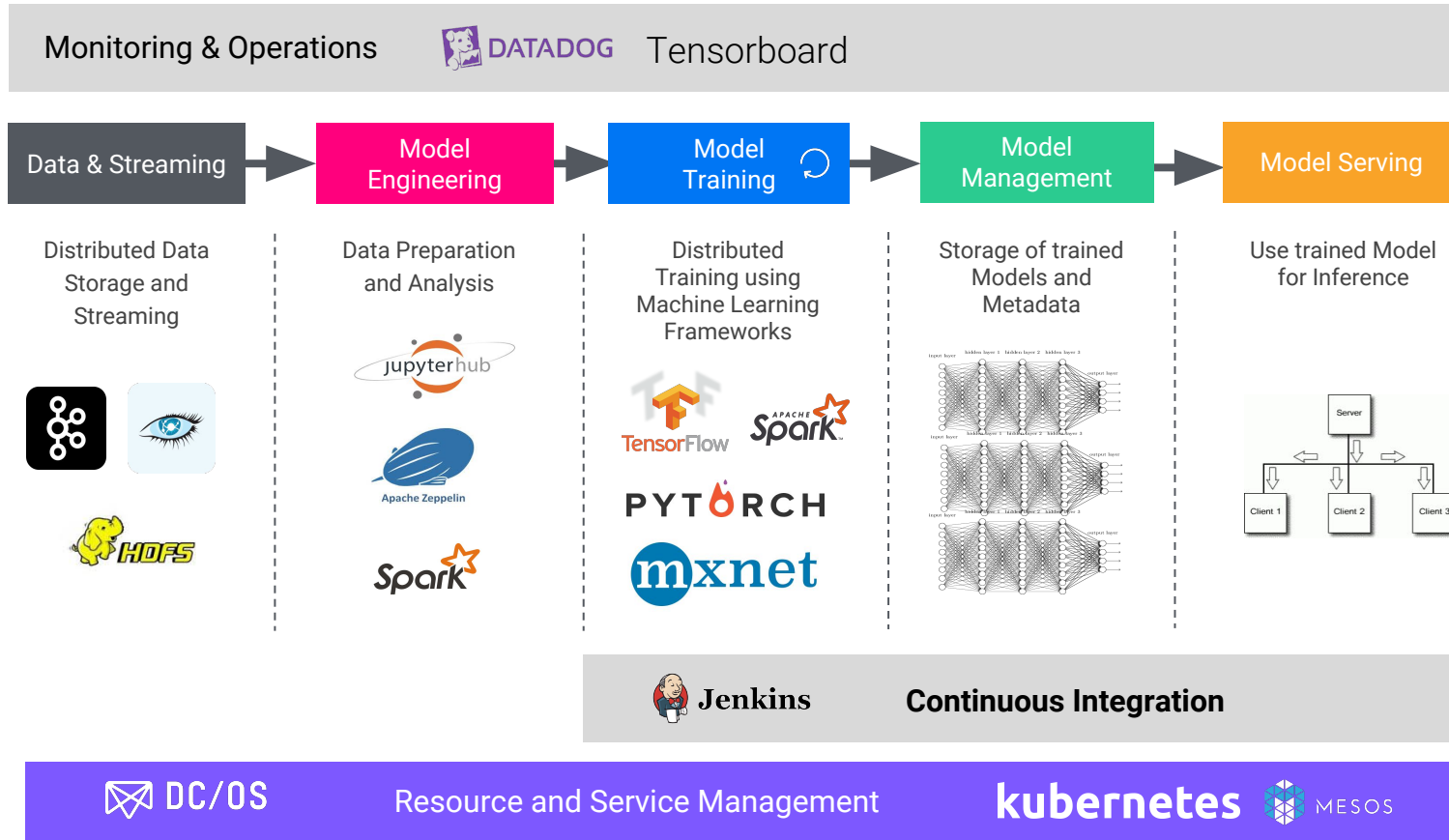
# Public Cloud Pipeline



# DIY Open Source Pipeline



# Data Science Pipeline



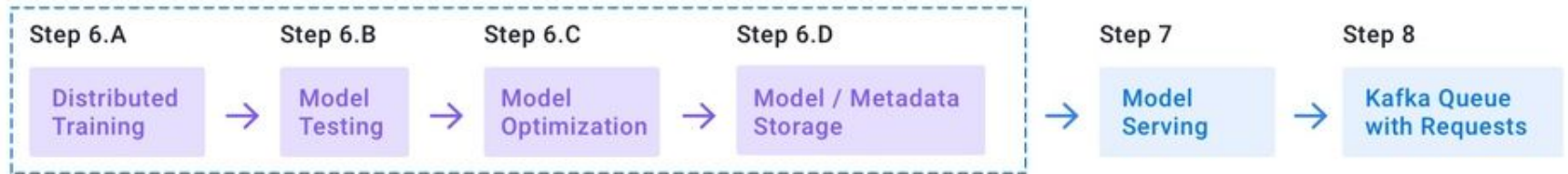


DATA SCIENCE



Automated using CI/CD

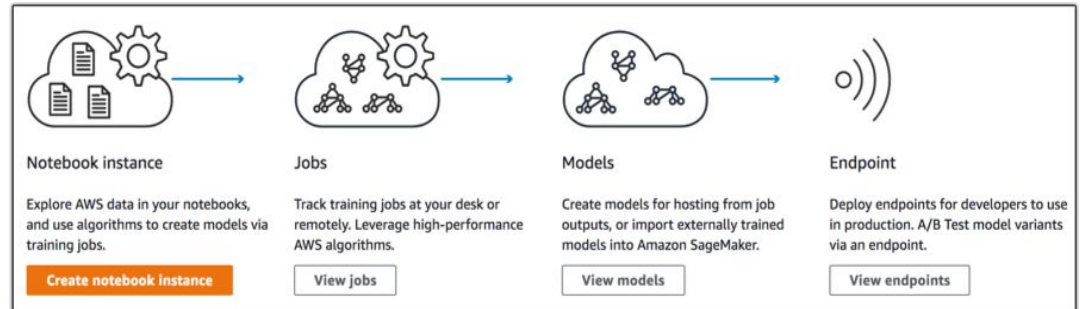
DATAOPS





# 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

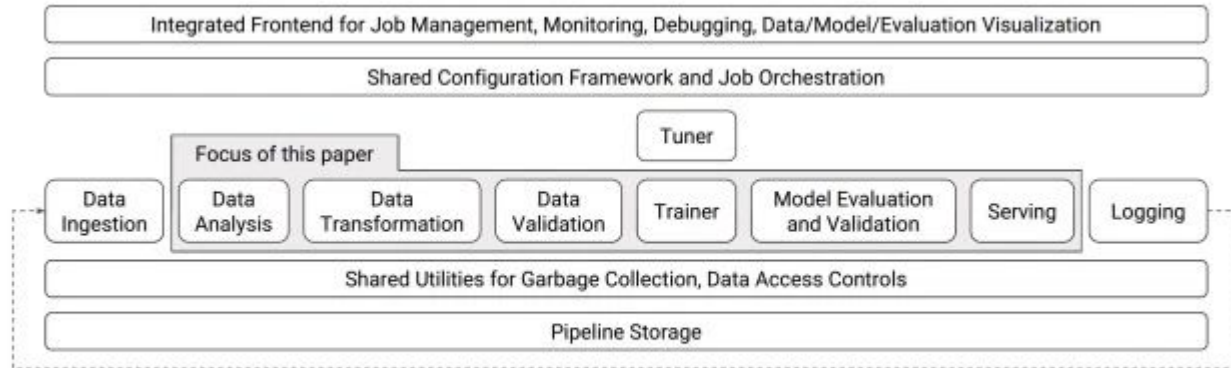


Figure 1: High-level component overview of a machine learning platform.

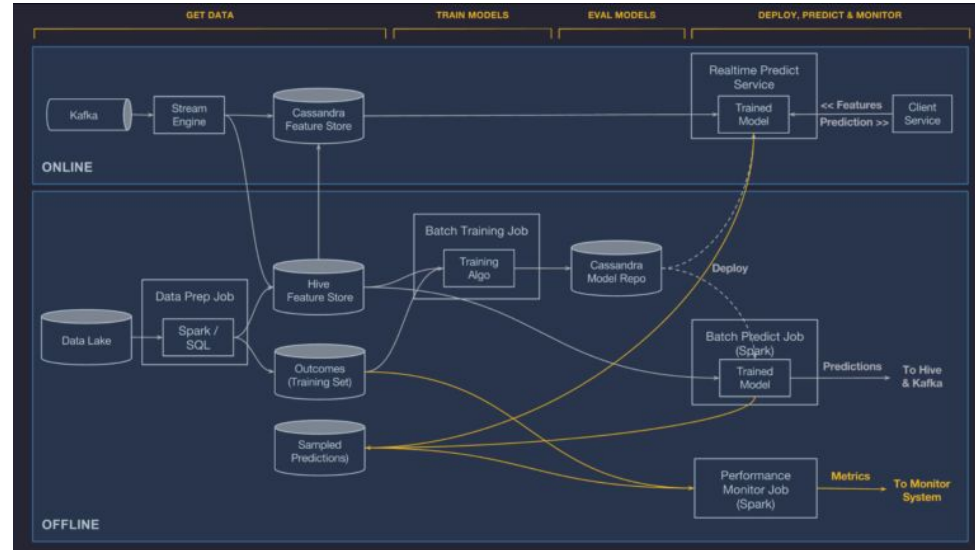
<http://www.kdd.org/kdd2017/papers/view/tfx-a-tensorflow-based-production-scale-machine-learning-platform>

<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

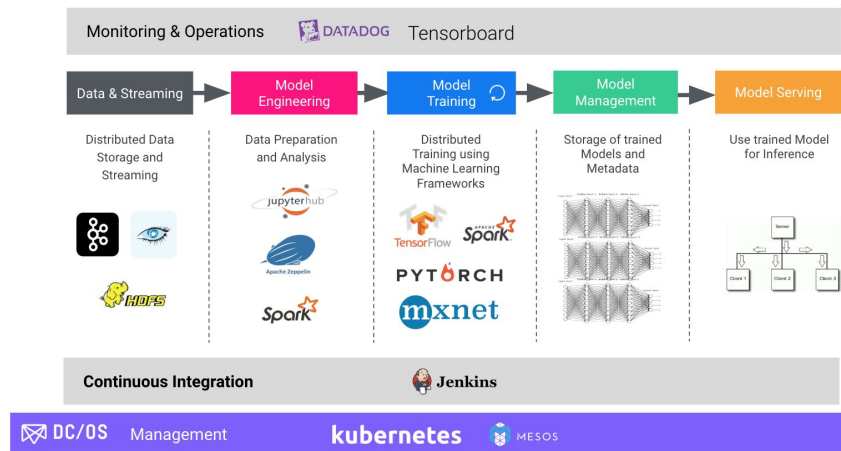
- Data is typically not ready to be consumed by ML job\*
  - Data Cleaning
    - Missing/incorrect labels
  - Data Preparation
    - Same Format
    - Same Distribution

\* 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

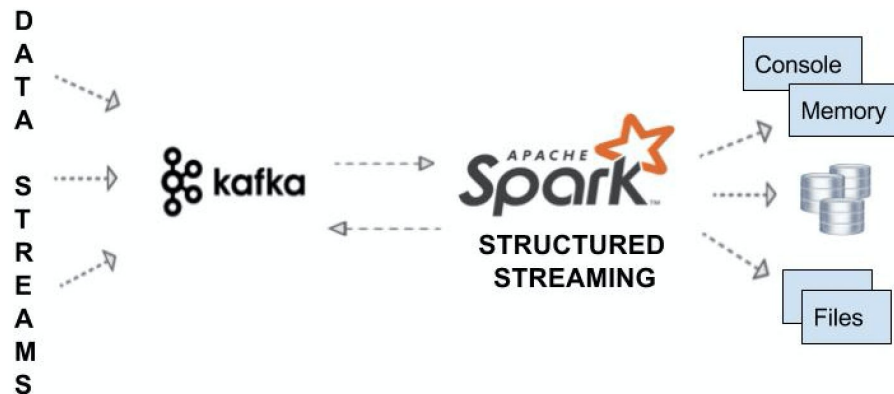
**Don't forget about the serving environment!!**



\* Demo datasets are a fortunate exception :)

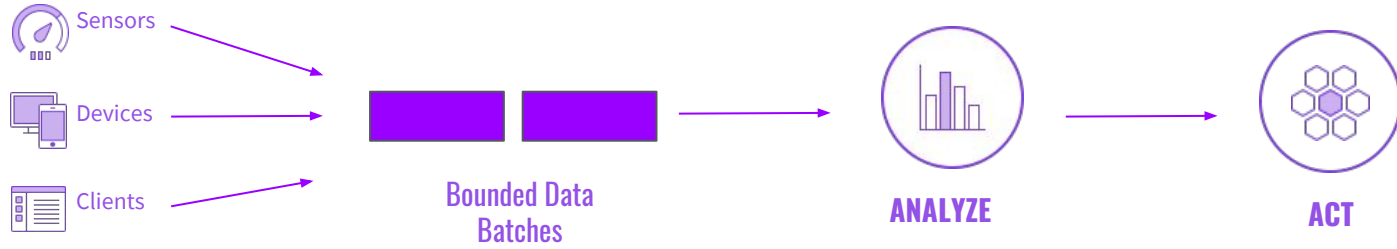
# Spark Overview

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

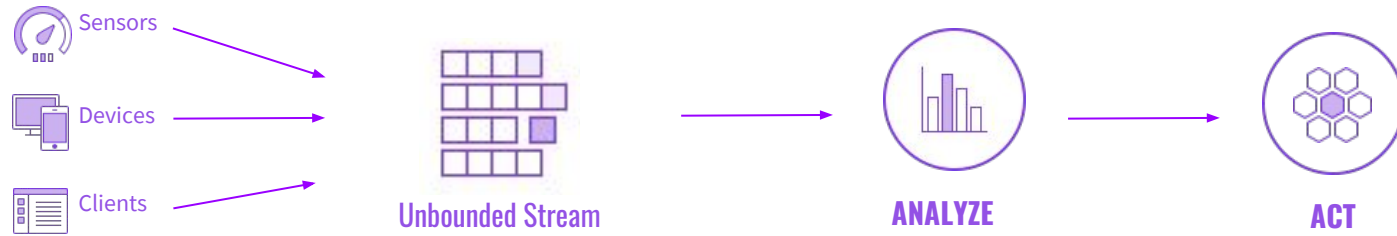


# Batch vs Streaming

Batch



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

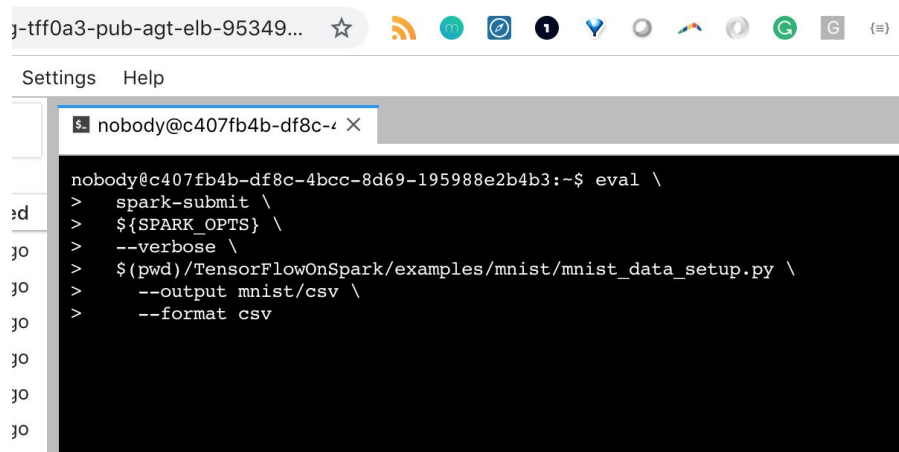
$ // Should return error
$ hdfs dfs -ls mnist/
```

<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/
```



The screenshot shows a terminal window with a dark background. The prompt is `nobody@c407fb4b-df8c-4bcc-8d69-195988e2b4b3:~$`. The user enters the command `eval \  
> spark-submit \  
> ${SPARK_OPTS} \  
> --verbose \  
> $(pwd)/TensorFlowOnSpark/examples/mnist/mnist_data_setup.py \  
> --output mnist/csv \  
> --format csv`. The terminal output is currently empty, indicating the command has been executed but its results have not yet been displayed.

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

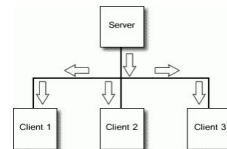
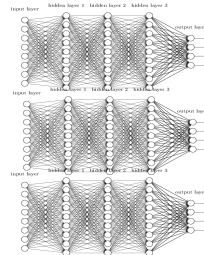


TensorFlow



APACHE SPARK

PYTORCH



Feature Catalogue



Jenkins

Continuous Integration

TensorFlow Hub

Model Library



Resource and Service Management

kubernetes

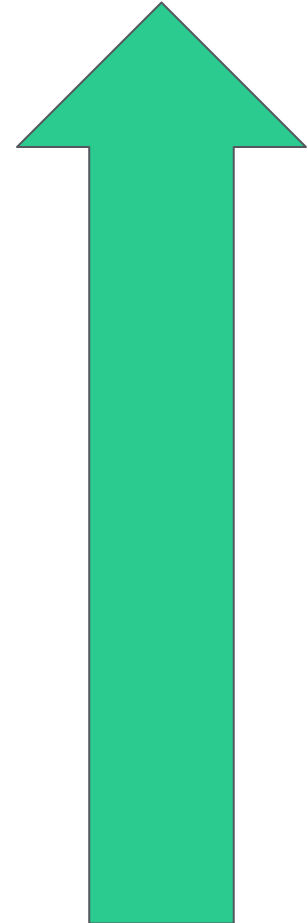
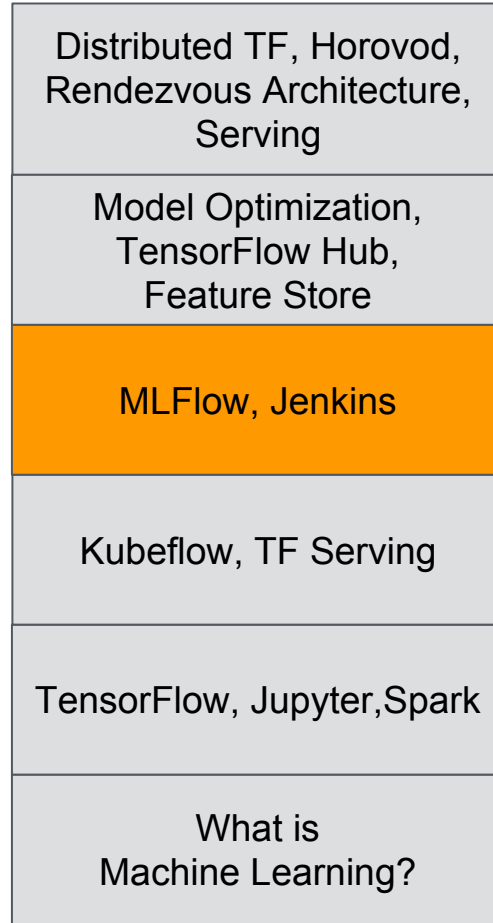


## Lab: (Optional) KubeFlow

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

# Automation

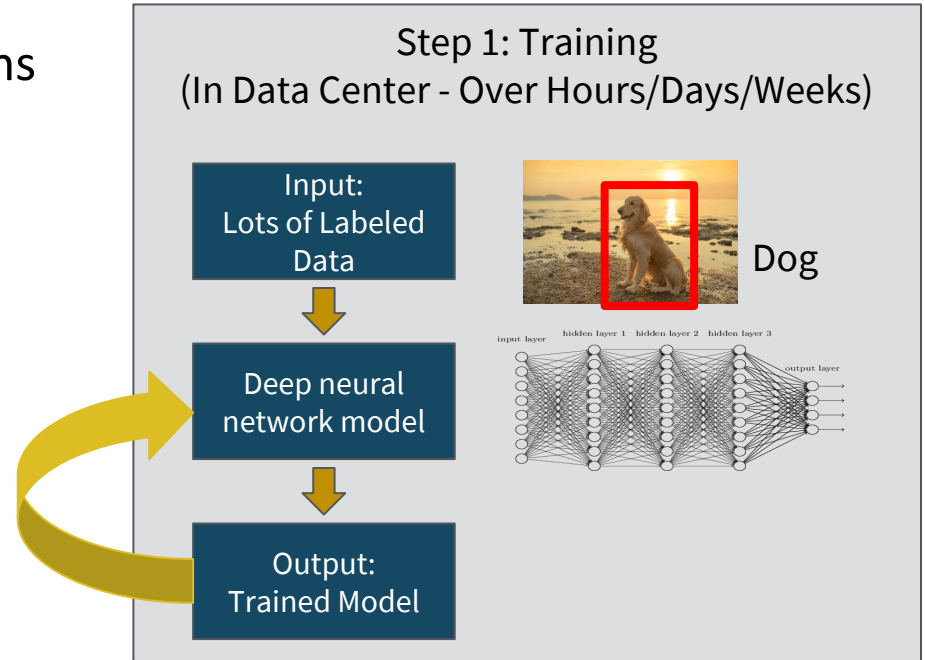
- Need for reproducibility
  - MLFlow
  - Jenkins
- **Open Source Technologies**
  - MLFlow
  - Jenkins
- **Labs**
  - Jenkins
  - [opt] MLFlow



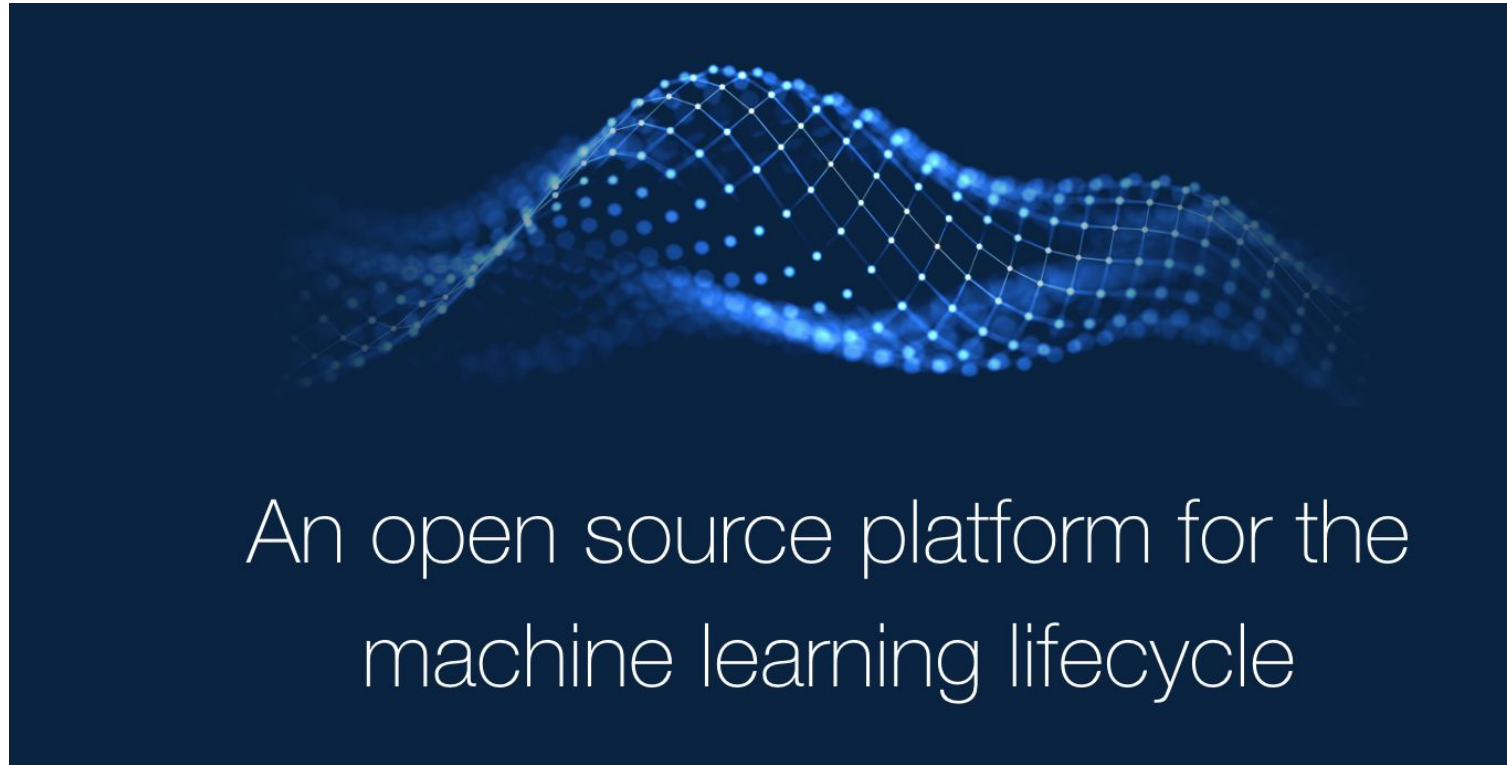
# Challenge: Reproducible Builds

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

mlflow



# MFlow



<https://mlflow.org/>

# 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

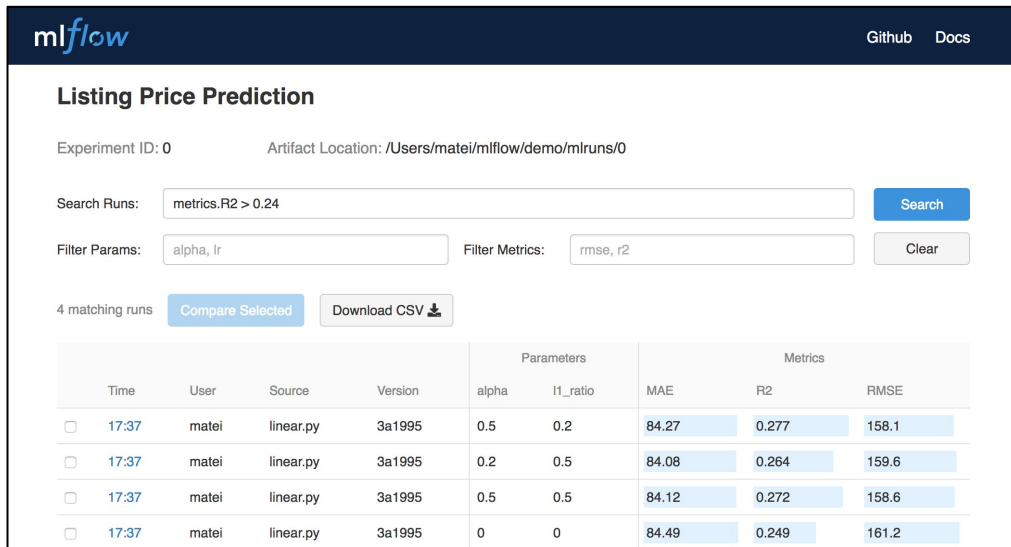
# MFlow Tracking

```
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")
```



The screenshot displays the MFlow web interface for an experiment titled "Listing Price Prediction". At the top right, there are links for "Github" and "Docs". Below the title, the "Experiment ID" is 0 and the "Artifact Location" is "/Users/matei/mlflow/demo/mlruns/0".

Search and filter options are provided:

- Search Runs:
- Filter Params:  Filter Metrics:

There are 4 matching runs. Below this, there are buttons for "Compare Selected" and "Download CSV".

	Time	User	Source	Version	Parameters		Metrics		
					alpha	l1_ratio	MAE	R2	RMSE
<input type="checkbox"/>	17:37	matei	linear.py	3a1995	0.5	0.2	84.27	0.277	158.1
<input type="checkbox"/>	17:37	matei	linear.py	3a1995	0.2	0.5	84.08	0.264	159.6
<input type="checkbox"/>	17:37	matei	linear.py	3a1995	0.5	0.5	84.12	0.272	158.6
<input type="checkbox"/>	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}"
```

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

# MFlow Model

```
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
```

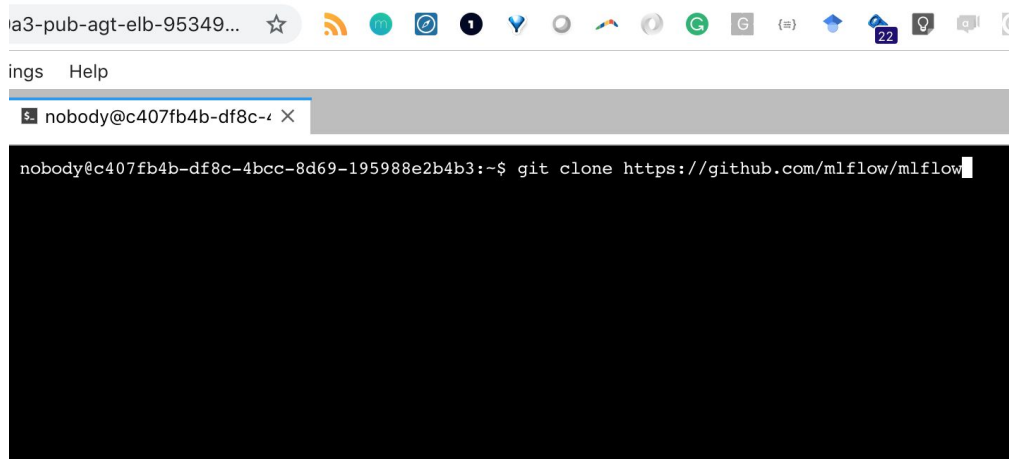
```
$mlflow run example/project -P alpha=0.5
```

```
$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>



The screenshot shows a terminal window with a dark background. The terminal prompt is `nobody@c407fb4b-df8c-4bcc-8d69-195988e2b4b3:~$`. The user has entered the command `git clone https://github.com/mlflow/mlflow`, and the terminal shows a cursor at the end of the command. The terminal window is titled `nobody@c407fb4b-df8c-4bcc-8d69-195988e2b4b3:~$`. The terminal window is open in a browser, with the address bar showing `ia3-pub-agt-elb-95349...`. The browser's address bar also contains a star icon and several extension icons. The terminal window is titled `ings Help`.

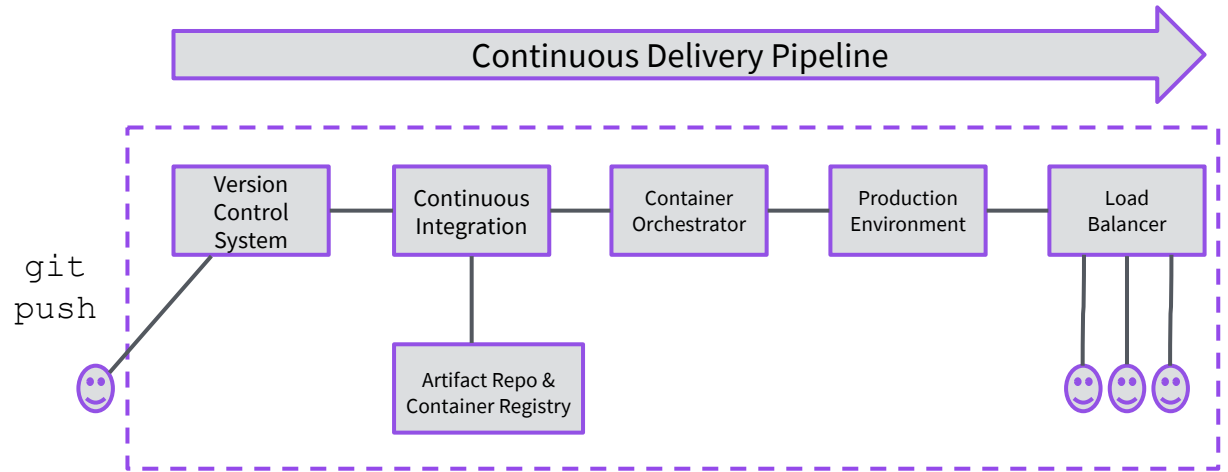
## Lab 2: MLflow

```
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
```

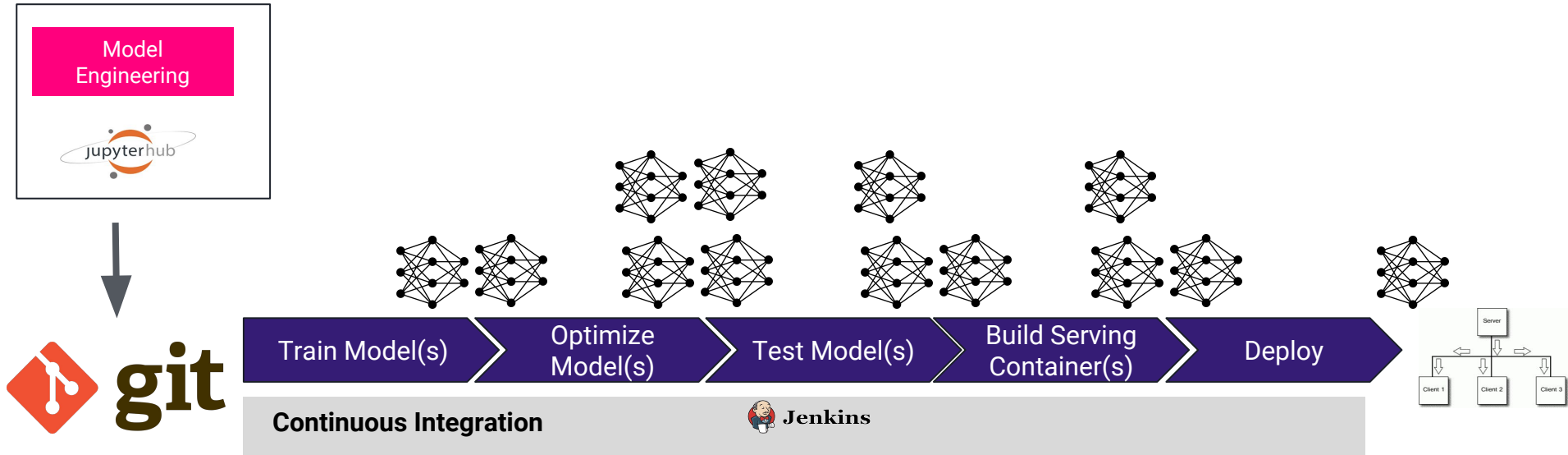
```
{"predictions": [6.379428821398614]}
```

# Challenge: CI/CD

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




# CI/CD for Data Science





# Lab 3: Jenkins


The screenshot displays the MLSummit user interface. On the left is a dark sidebar with the user profile 'MLSummit Bootstrap superuser' at the top. Below are navigation items: 'Dashboard', 'Services', 'Jobs', 'Catalog' (highlighted in purple), 'Resources', 'Nodes', 'Networking', and 'Secrets'. The main content area is titled 'Catalog' and features a search bar with 'Jenkins' entered. Below the search bar, it states '1 service found'. A single service card is shown with the Jenkins logo, the name 'jenkins', the version range '3.5.1-2.107.2', and a 'Certified' badge.


# Lab 3: Jenkins

 MLSummit ▾  
Bootstrap superuser


 Dashboard

 Services











 Jobs

 Catalog

Resources

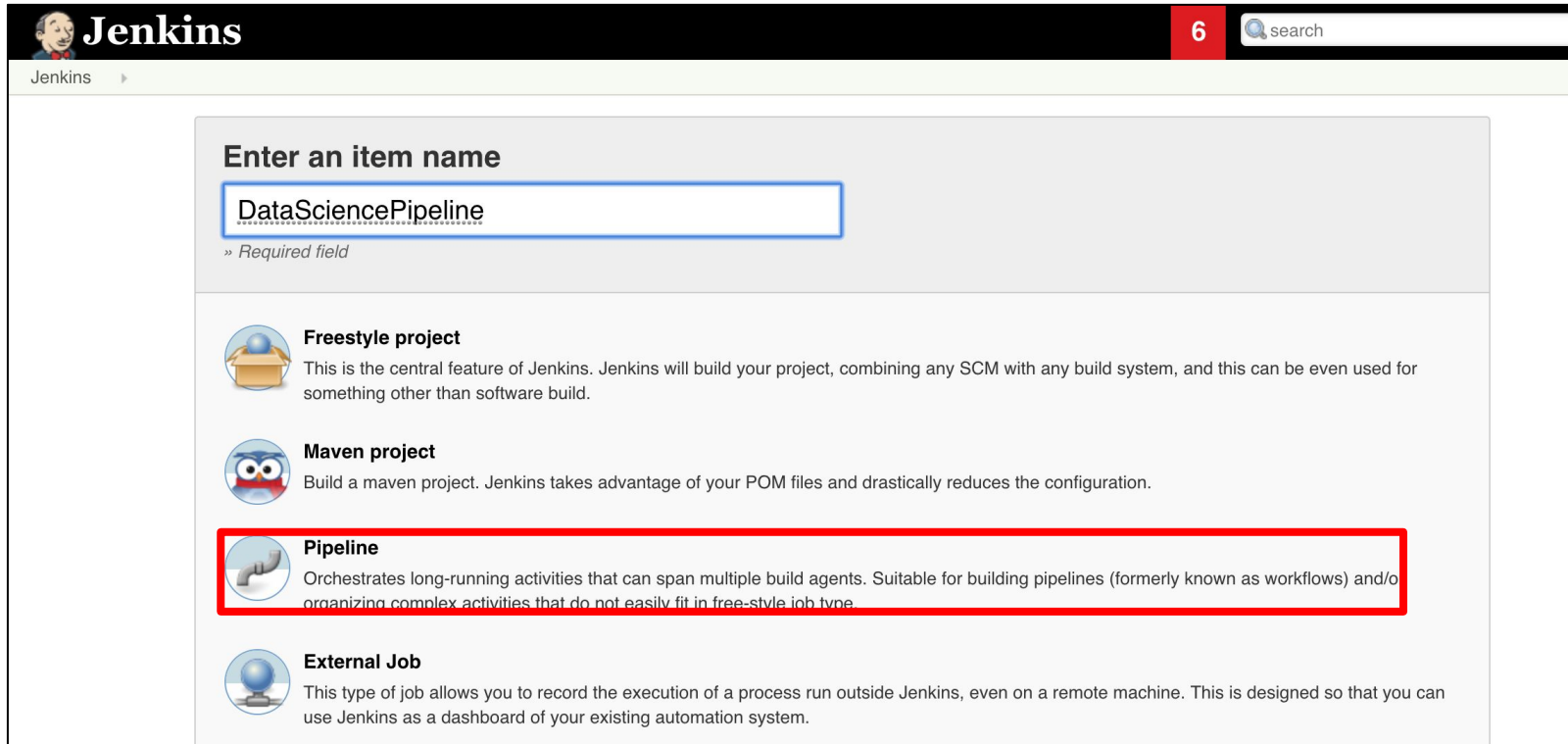
 Nodes

## Services

Name ▾	Status ?	Instances
 hdfs	 Running	1
 jenkins 	 Deploying 	1
 jupyterlab-notebook	 Running	1
 marathon-lb	 Running	1







# Lab 3: Jenkins



The screenshot shows the Jenkins web interface. At the top left is the Jenkins logo and name. At the top right is a search bar and a red tab with the number '6'. Below the header is a breadcrumb 'Jenkins >'. The main content area is titled 'Enter an item name' and contains a text input field with 'DataSciencePipeline' entered. Below the input field is a note '» Required field'. Below this is a list of job types, each with an icon and a description. The 'Pipeline' option is highlighted with a red border.

**Enter an item name**

» Required field

-  **Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
-  **Maven project**  
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
-  **Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
-  **External Job**  
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.

# Lab 3: Jenkins

**Pipeline**

Definition: Pipeline script from SCM

SCM: Git

Repositories

Repository URL:  **Please enter Git repository.**

Credentials:

Branches to build

Branch Specifier (blank for 'any'):

Repository browser: (Auto)

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

# Lab 3: Jenkins

The screenshot shows the Jenkins web interface for a pipeline named "DataSciencePipeline". The top navigation bar includes the Jenkins logo, a search bar, and a notification badge with the number "6". The breadcrumb trail shows "Jenkins > DataSciencePipeline".

On the left sidebar, the "Build Now" button is highlighted with a red rectangle. Other sidebar options include "Back to Dashboard", "Status", "Changes", "Delete Pipeline", "Configure", "Full Stage View", "Job Config History", "Open Blue Ocean", "Embeddable Build Status", "Pipeline Syntax", and "Build History".

The main content area is titled "Pipeline DataSciencePipeline" and includes an "add description" link and a "Disable Project" button. Below this is a "Recent Changes" section with a "Recent Changes" link.

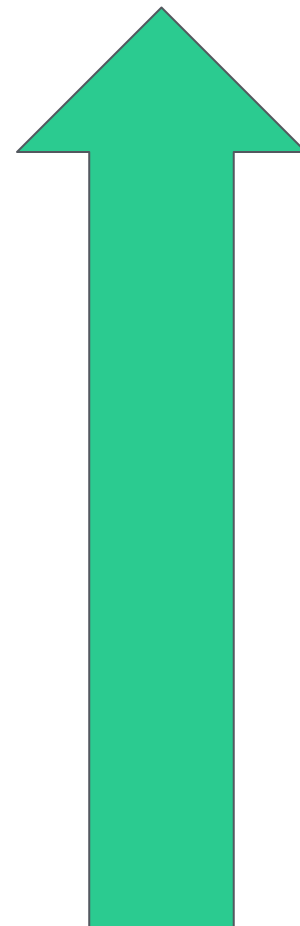
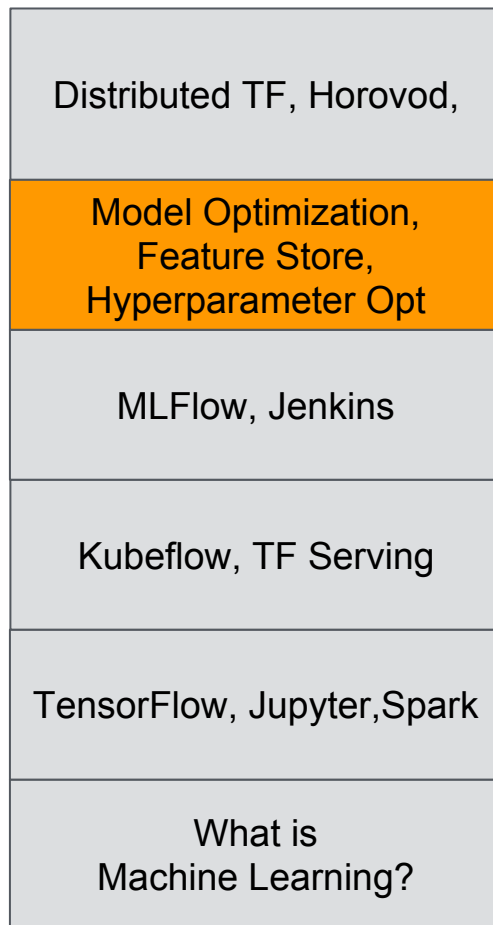
The "Stage View" section displays a table of stage execution times. The average stage times are: Checkout (16s), Train Model(s) (10s), Optimize Model(s) (5s), Test (17ms), Build Serving container (80ms), and Deploy (626ms). The average full run time is approximately 1 minute and 33 seconds.

Stage	Checkout	Train Model(s)	Optimize Model(s)	Test	Build Serving container	Deploy
Average stage times	16s	10s	5s	17ms	80ms	626ms
Build #1 (Oct 01, 08:26)	16s	10s	5s	17ms	80ms	626ms

Below the table, there is a "Permalinks" section with links for "RSS for all" and "RSS for failures".

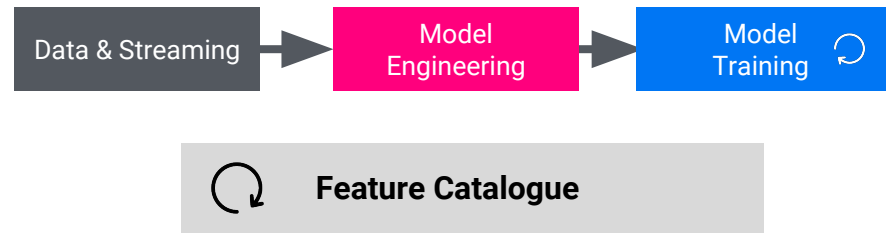
# Model Management

- 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  $\approx$  Preprocessing Cache + Discovery



# Feature Store

## ML Feature Data Warehouse

Reduce the cost of generating and storing the feature data

## Just-in-time Feature Transforms

Allow the research teams to experiment with new features and new feature engineering techniques

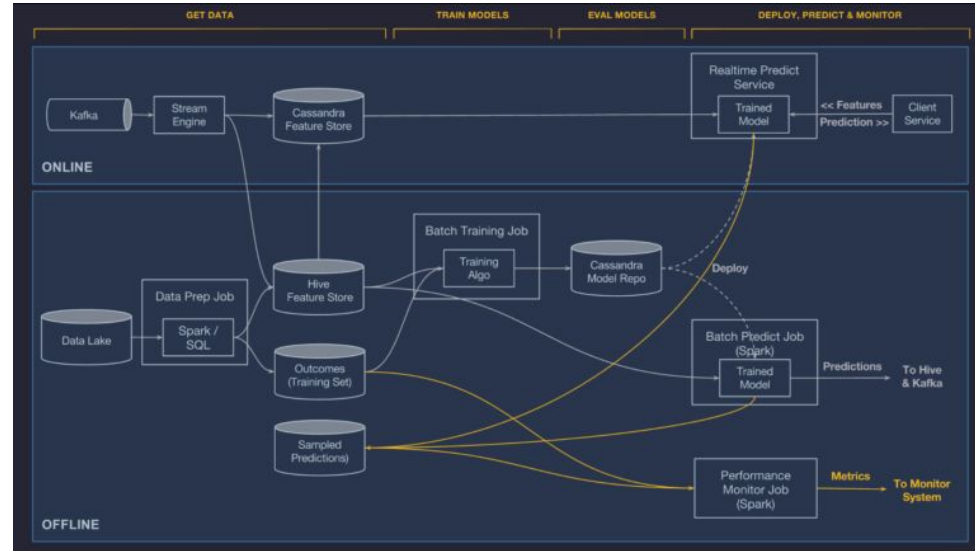


<https://techblog.appnexus.com/lessons-learned-from-building-scalable-machine-learning-pipelines-822acb3412ad>

# 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

## Featuretools

An open source python framework for automated feature engineering

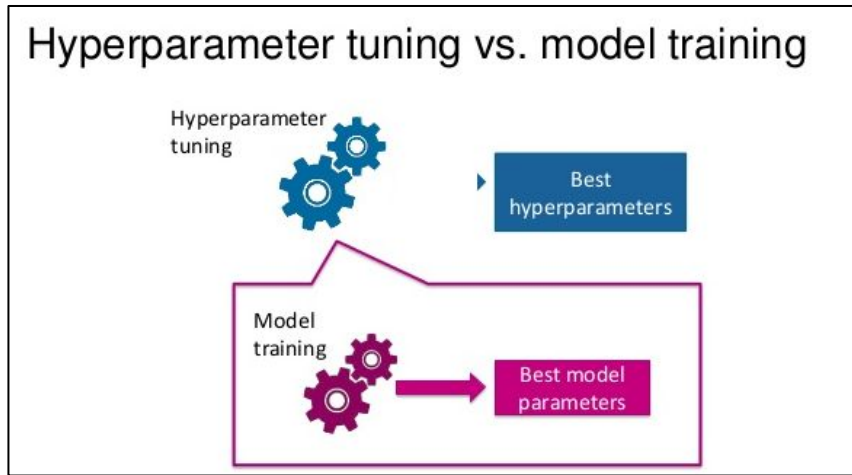
LET'S GET STARTED

★ Star 2,279

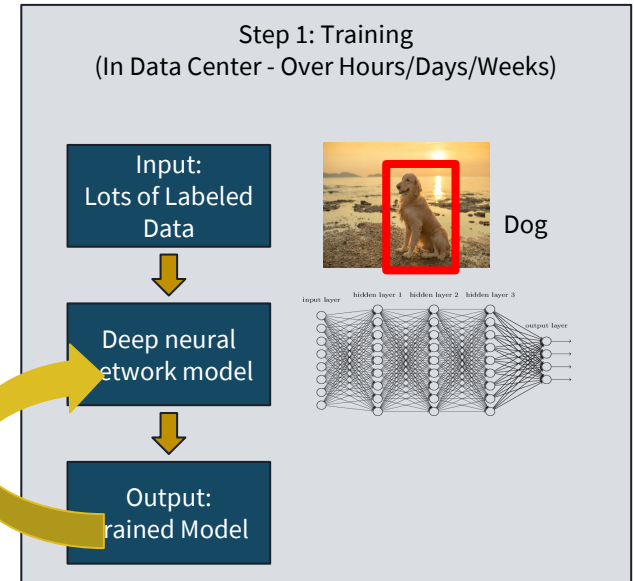
<https://www.featuretools.com/>



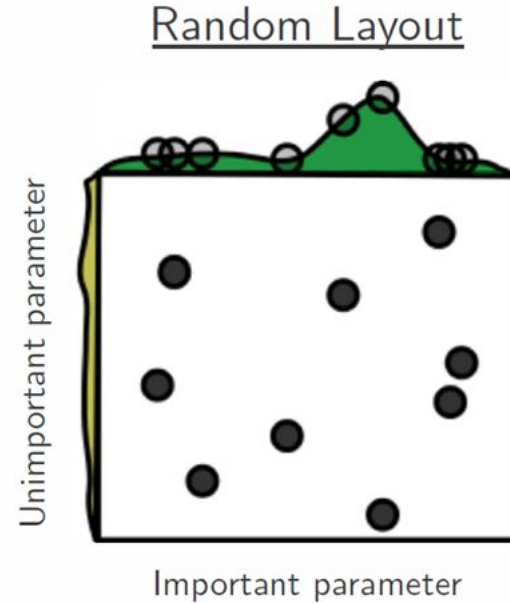
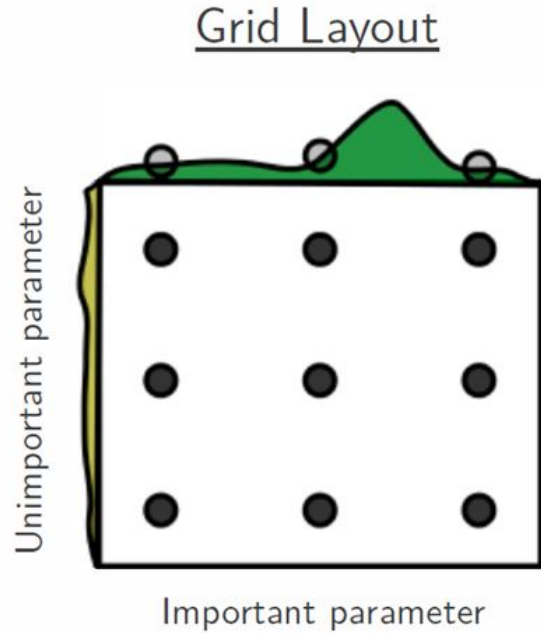
# Hyperparameter Optimization



- Networks Shape
- Learning Rate
- ...



# Hyperparameter Search



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

# Dask-ML for Hyperparameter Search

```
# 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)
```

# 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

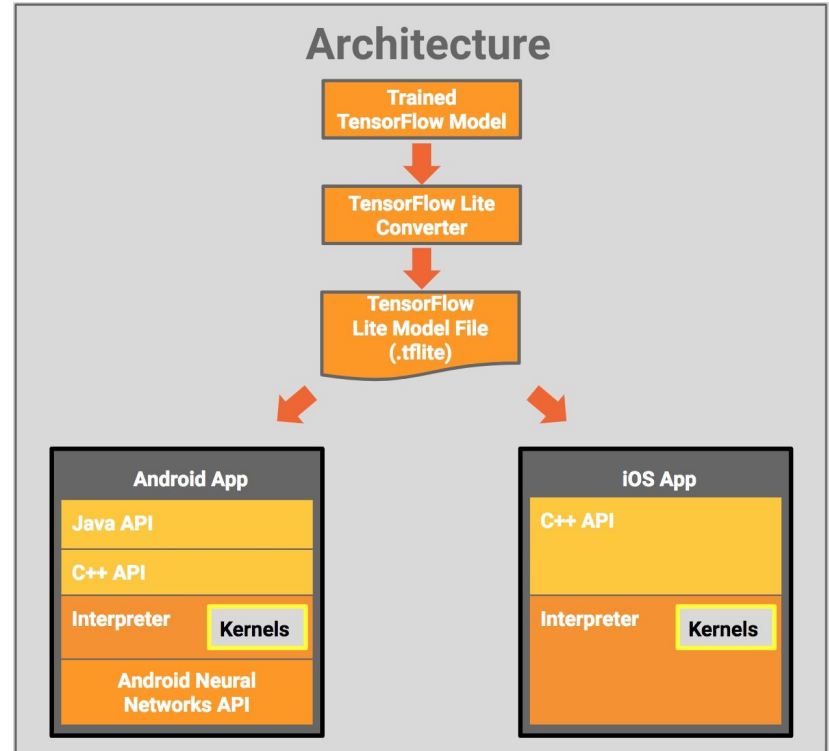
```
import dask.dataframe as dd
df = dd.read_parquet('...')
data = df[['age', 'income', 'married']]
labels = df['outcome']

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

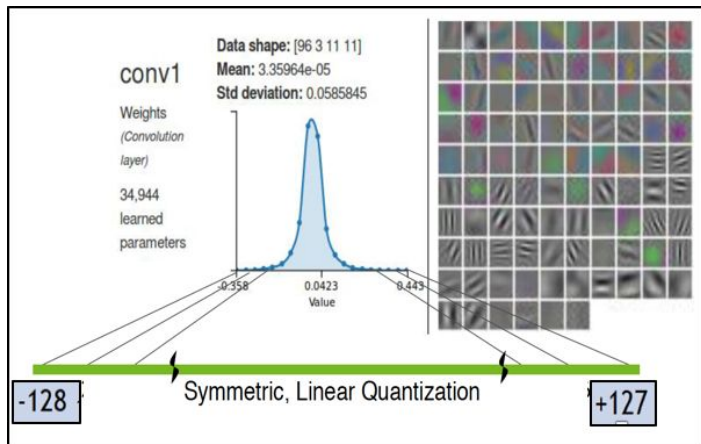
- Different Serving Environments
  - Mobile
  - GPU
  - CPU
- Small/Fast model without losing too much performance
- 500 KB models....



# Model Optimization

```
transform_graph \  
  --in_graph=unoptimized_cpu_graph.pb \  
  --out_graph=optimized_cpu_graph.pb \  
  --inputs='x_observed:0' \  
  --outputs='Add:0' \  
  --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'
```

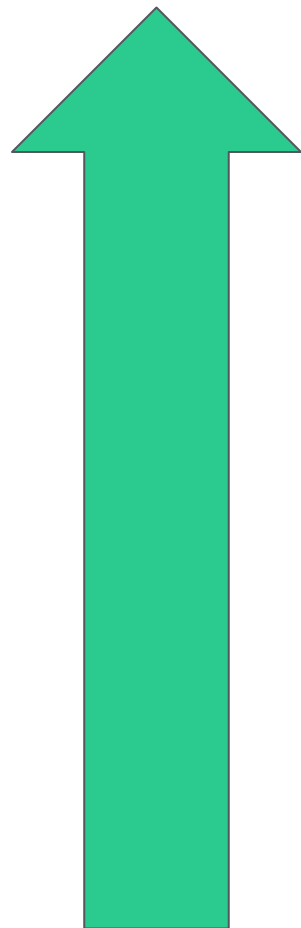
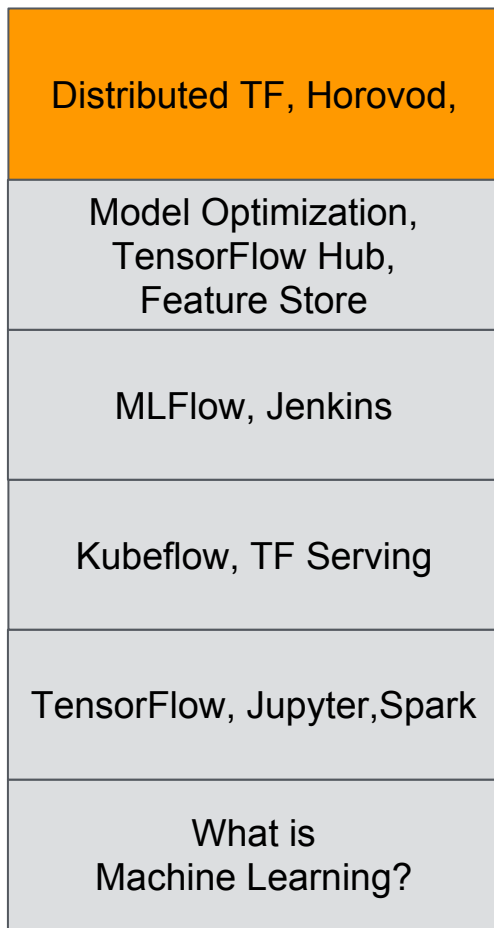
# Model Optimization



	Dynamic Range	Min Pos Value
FP32	$-3.4 \times 10^{38} \sim +3.4 \times 10^{38}$	$1.4 \times 10^{-45}$
FP16	-65504 ~ +65504	$5.96 \times 10^{-8}$
INT8	<b>-128 ~ +127</b>	1

# Distributed TensorFlow

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

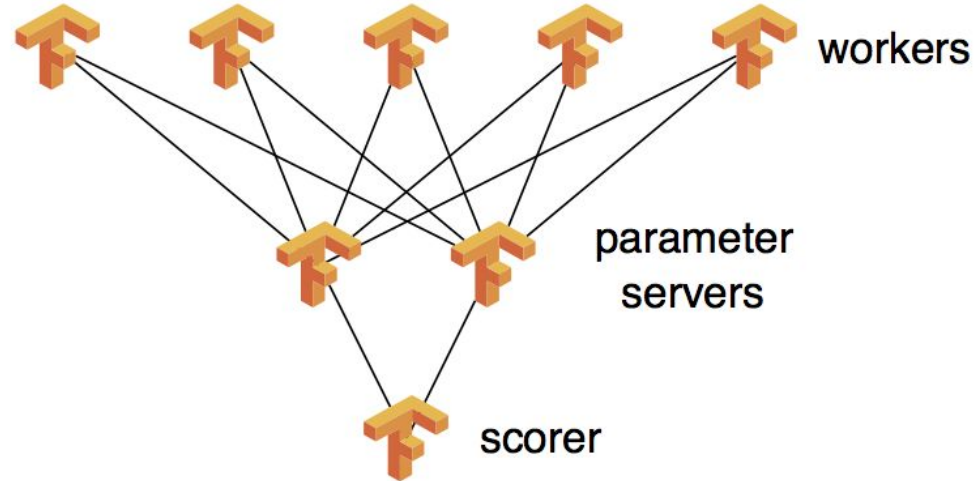




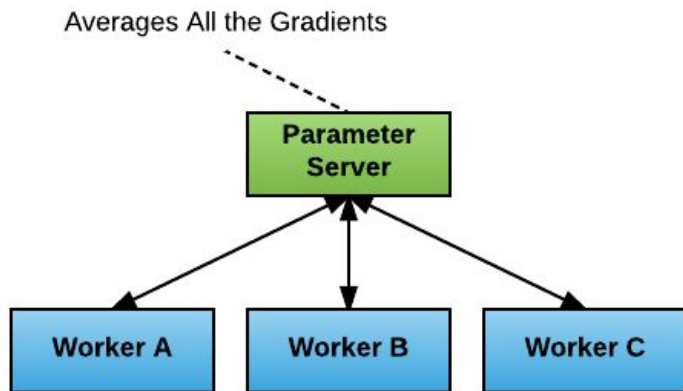
# TensorFlow Overview



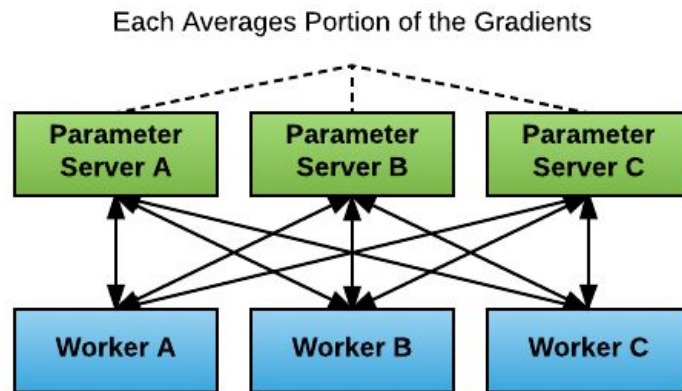
# Challenge: Distributed TensorFlow



# Challenge: Distributed TensorFlow

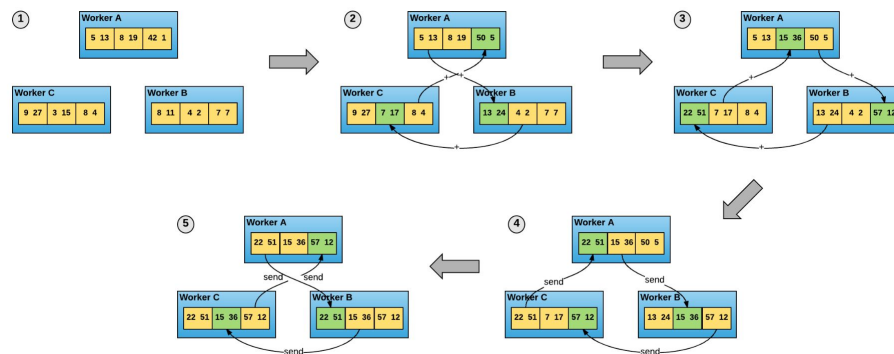


or



# 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 ;)



# 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.

# Challenge: “Libraries”

- Different Frameworks
- Existing architectures
- Pretrained models

```
import tensorflow as tf
import tensorflow_hub as hub

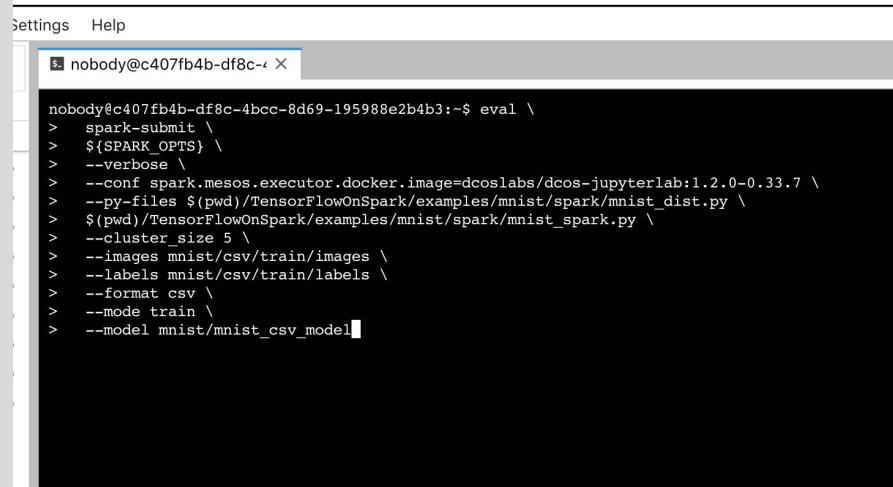
with tf.Graph().as_default():
    embed = hub.Module("https://tfhub.dev/google/nlm-en-dim128-with-normalization/1")
    embeddings = embed(["A long sentence.", "single-word", "http://example.com"])

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

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

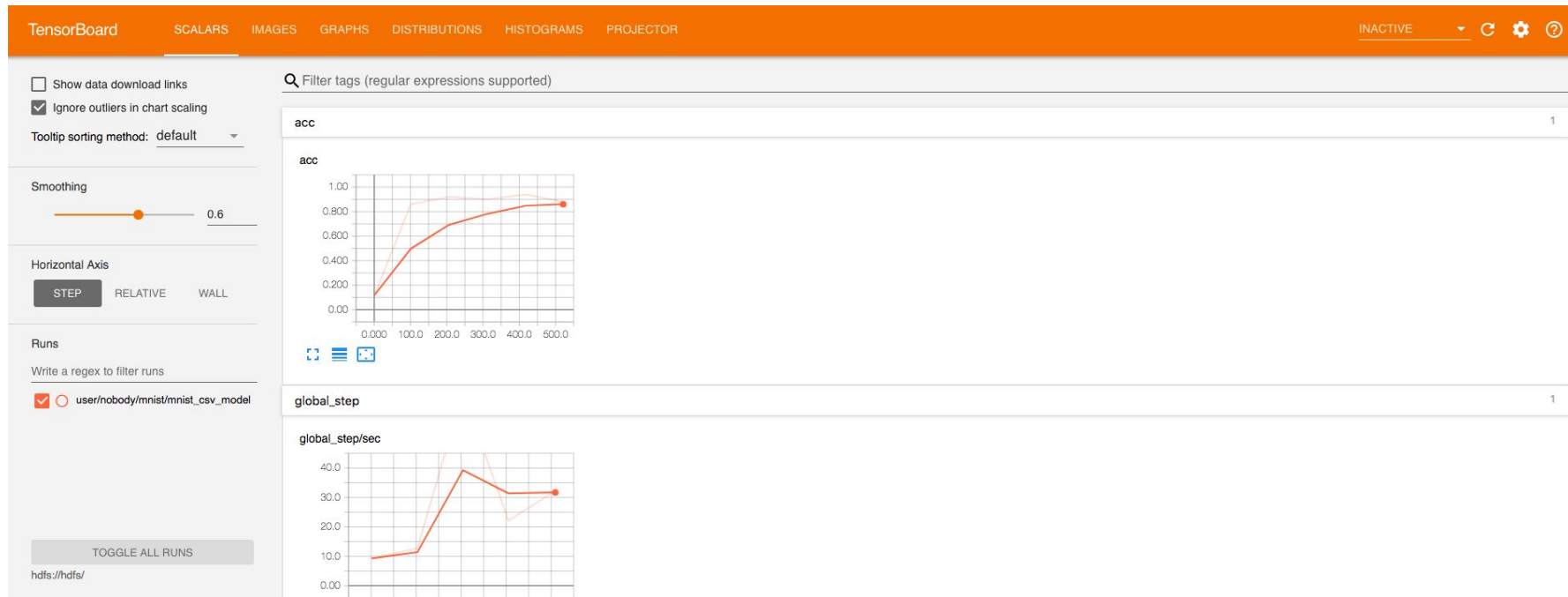
# Lab 4: TensorFlow

```
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
```



```
Settings Help  
nobody@c407fb4b-df8c-4bcc-8d69-195988e2b4b3:~$ 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



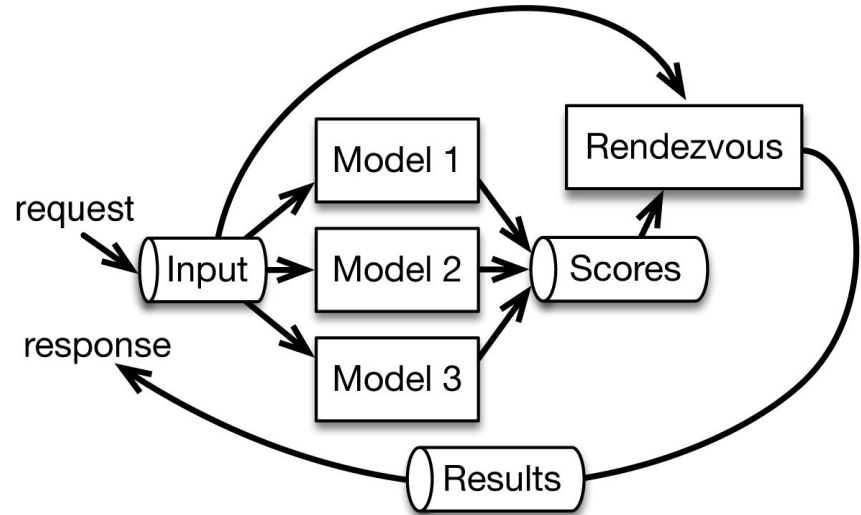
## Further Reading/Watching



Distributed Deep Learning with Apache Spark and TensorFlow  
Jim Dowling (Logical Clocks AB)

# Challenge: Testing

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



# Challenge: Debugging

The screenshot displays the TensorBoard Debugger interface, which is currently inactive. The interface is divided into several sections:

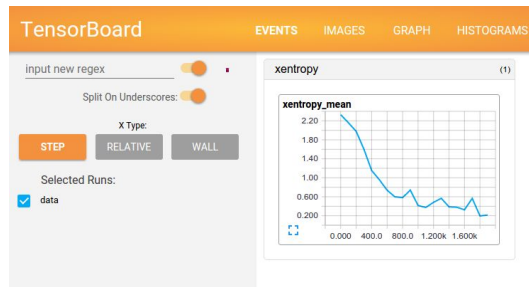
- Node List:** Shows a list of nodes for the current session. The selected node is `/job:localhost/replica:0/task:0/device:CPU:0`. It includes checkboxes for `Add`, `Const`, `Mul`, and `Variable`.
- Source Code:** Displays the source code for the selected node, which is `tdp_demo.py`. The code includes imports for TensorFlow and TensorFlow Debug, and defines variables `a`, `b`, `c`, `x`, and `y`. It also shows the session configuration and the execution of `tf.global_variables_initializer()`.
- Runtime Graphs:** Shows the runtime graph for the selected node. The graph includes nodes for `Variable_1`, `Variable`, and `Add`.
- Tensor Value Overview:** Provides a table of tensor values for the selected node. The table has columns for Tensor, Count, DType, Shape, and Value. The values are: `Variable:0` (Count: 1, DType: float32, Shape: [], Value: 4), `Variable/read:0` (Count: 1, DType: float32, Shape: [], Value: 4), `Const:0` (Count: 1, DType: float32, Shape: [], Value: 10), and `Variable_1:0` (Count: 1, DType: float32, Shape: [], Value: 2).
- Session Runs:** Shows a table of session runs. The table has columns for Feeds, Fetches, Targets, #(Devices), and Count. The runs are: `init` (Fetches: 1, Targets: 1, #(Devices): 1, Count: 1) and `Add:0` (Fetches: 1, Targets: 1, #(Devices): 1, Count: 1).
- Health Pill:** A visual indicator of the session's health, showing a blue pill and a legend for `NaN`, `0`, and `+∞`.

[https://www.tensorflow.org/programmers\\_guide/debugger](https://www.tensorflow.org/programmers_guide/debugger)

# Challenge: Monitoring

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

- TensorBoard



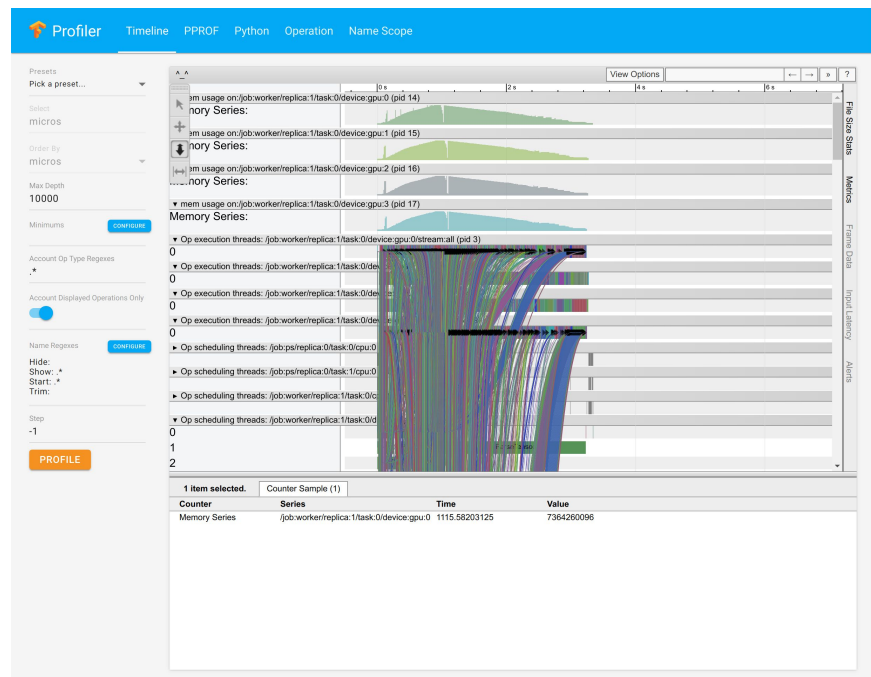
- Traditional Cluster Monitoring Tool

# Lab 5: TensorBoard

To CSV mnist

# Profiling

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

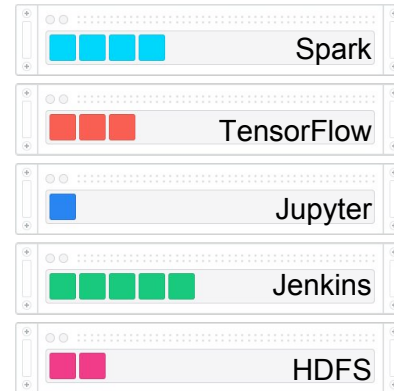


## Lab 6 (Optional): TFDebug

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

# Challenge: Resource and Service Management

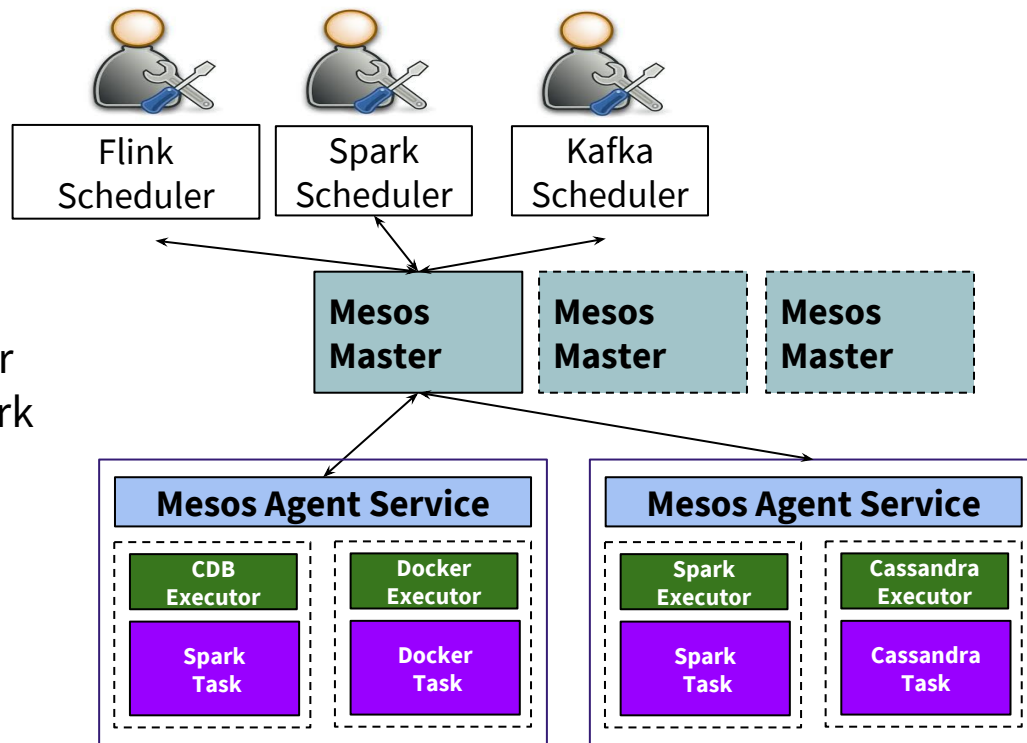
- Different Distributed Systems
  - Deployment
  - Updates
  - Failure Recovery
  - Scaling
- Resource Efficiency
  - Multiple VM per Service?



**Typical Datacenter**  
siloed, over-provisioned servers,  
low utilization



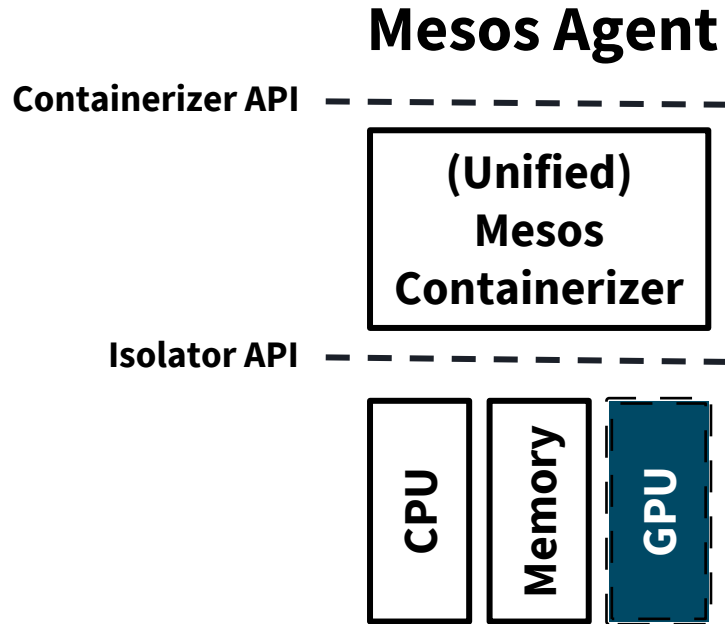
# Apache Mesos



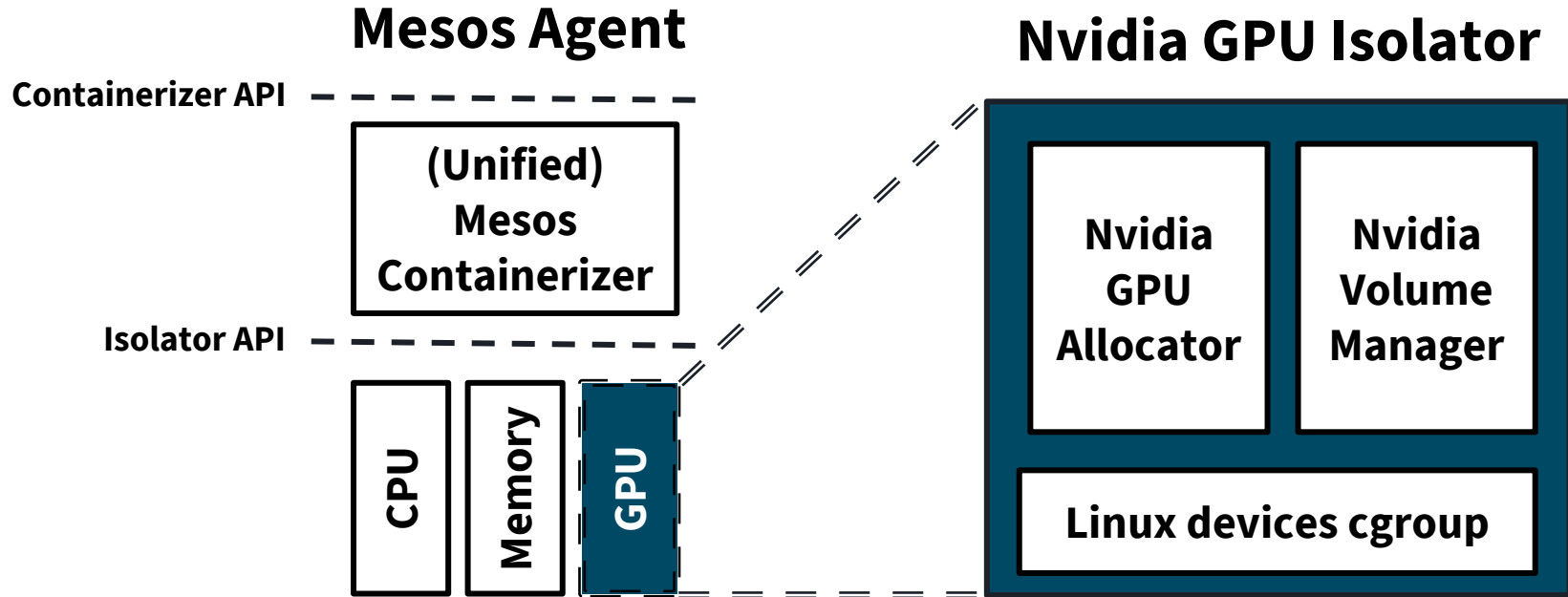
## Two-level Scheduling

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

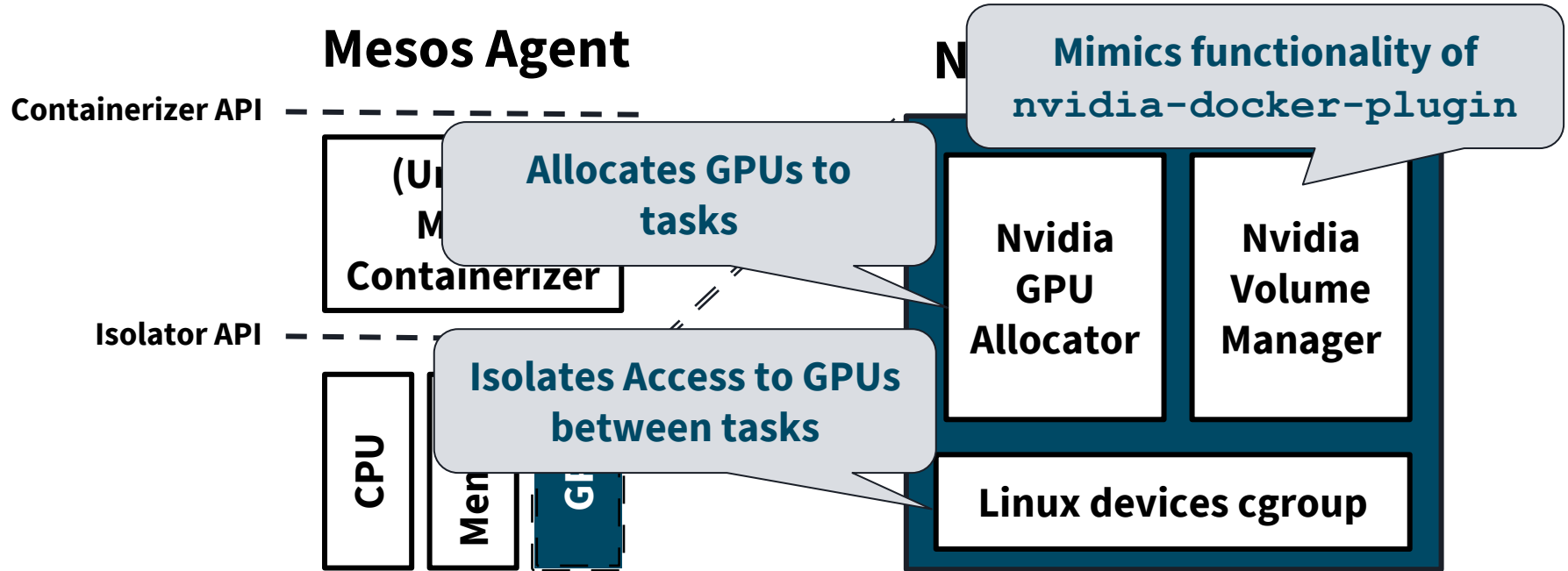
# Example: GPU Isolation



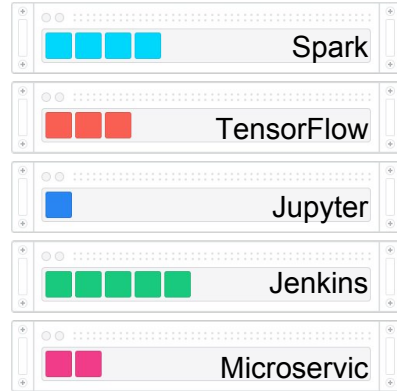
# Example: GPU Isolation



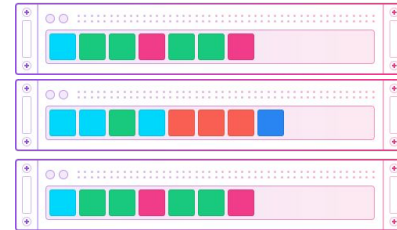
# Example: GPU Isolation



# Resource Management

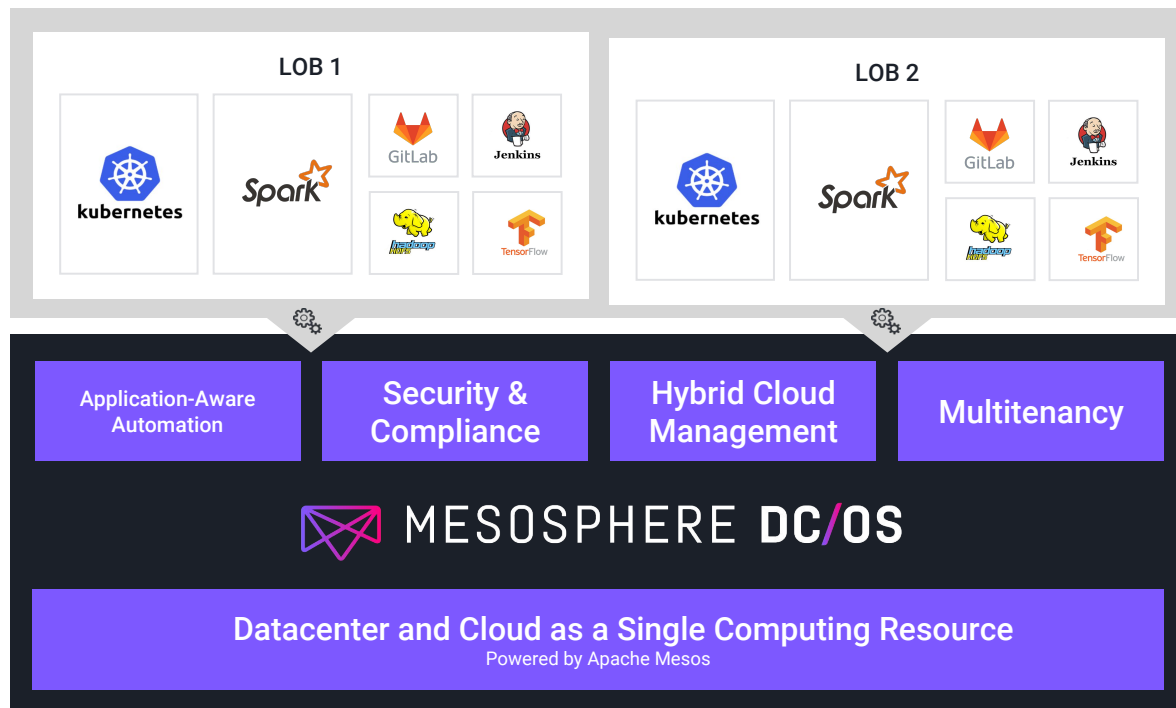


**Typical Datacenter**  
e  
siloed, over-provisioned servers,  
low utilization

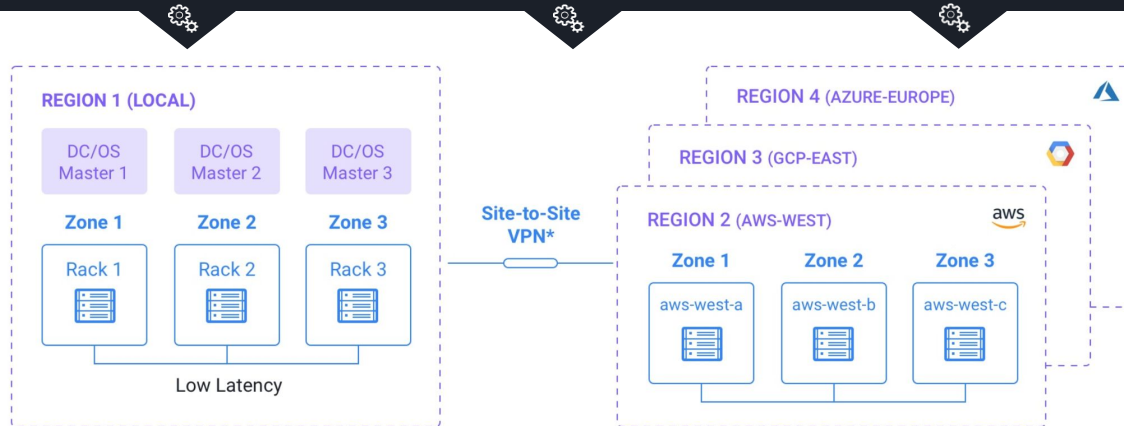


**DC/OS**  
automated schedulers, workload multiplexing onto the same  
machines

# Service Orchestration



# Resource Management



Monitoring & Operations



DATADOG

TensorBoard

Data & Streaming

Model Engineering

Model Training

Model Management

Model Serving

Distributed Data Storage and Streaming

Data Preparation and Analysis

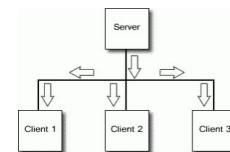
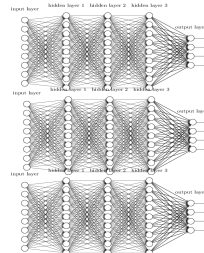
Distributed Training using Machine Learning Frameworks

Storage of trained Models and Metadata

Use trained Model for Inference



PYTORCH



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



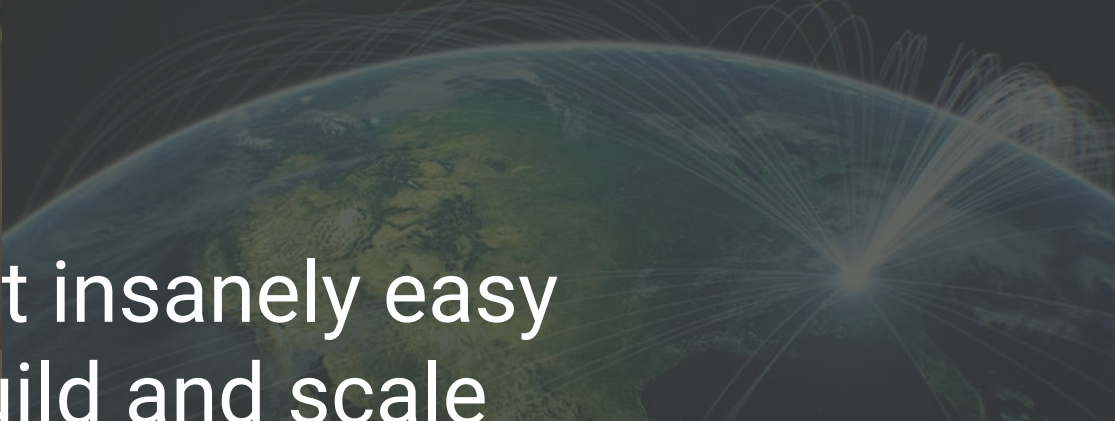

/dcos  
/dcos/examples  
/dcos/demos

THANK YOU!

# ANY QUESTIONS?



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



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



MESOSPHERE