Hopsworks: Horizontally Scalable ML Pipelines

Dr. Jim Dowling, CEO
Logical Clocks

21 August 2019
Dresden ScaDS
5th International Summer School on Big Data and ML
Dr. Jim Dowling  
CEO

Steffen Grohsschmidt  
Head of Cloud

Theofilos Kakantousis  
COO

Fabio Buso  
Head of Engineering

Prof Seif Hardi  
Chief Scientist

Silicon Valley Office  
470 Ramona St  
Palo Alto  
California, USA

Stockholm Office  
Box 1263,  
Isafjordsgatan 22  
Kista, Sweden

UK Office  
IDEALondon,  
69 Wilson St,  
London, EC2A2BB, UK

Venture Capital Backend (Inventure, Frontline.vc, AI Seed)

For more information, visit us on the Web at: www.logicalclocks.com
If you’re working with big data and Hadoop, **this one paper could repay your investment** in the Morning Paper many times over.... **HopsFS is a huge win.**

- Adrian Colyer, *The Morning Paper*
Evolution of Distributed Filesystems

- **Past**
  - NFS, HDFS: Single DC, Strongly Consistent Metadata

- **Present**
  - S3: Multi-DC, Eventually Consistent Metadata
  - GCS: Multi-DC, Strongly Consistent Metadata
  - HopsFS: POSIX-like Filesystem

- **Object Store**
Why HopsFS?

- Distributed FS
  - Needed for Parallel ML Experiments / Dist Training / FeatureStore
- Provenance/Free-text-search
  - Change Data Capture API
- Performance
  - >1.6m ops/sec over 3 AZes on GCP using Spotify’s Hadoop workload
  - NVMe for small files stored in metadata
- HDFS API
  - TensorFlow/Keras/PySpark/Beam/Flink/PyTorch (Petastorm)
Hopsworks – a platform for Data Intensive AI built on HopsFS
0. Slides:
http://hops.io/dresden.pdf

1. Register for an account at:
www.hops.site

2. Follow the Instructions here:
https://bit.ly/2UEixTr

3. Getting started Videos
Hopworks hides the Complexity of Deep Learning

Hopworks
Feature Store

[Hopsworks REST API]

Data → Model → Prediction

[Adapted from Schulley et Al “Technical Debt of ML”]
Hopsworks

The Platform for Data Intensive AI
- Machine Learning, Deep Learning & Model serving
Hopworks

Datasources

Batch
- Apache Beam
- Apache Spark

Streaming
- Apache Beam
- Apache Spark
- Apache Flink

Feature Store
- Hopworks Feature Store

Distributed ML & DL
- Pip install
- Conda libraries
- Tensorflow
- scikit-learn
- Keras
- Jupyter Notebooks
- Tensorboard
- Kubernetes

Serving

Filesystem and Metadata storage
- HopsFS

Data Preparation & Ingestion

Experimentation & Model Training

Deploy & Productionalize

Applications

API

Dashboards
Hopworks

Orchestration in Airflow

**Batch**
- Apache Beam
- Apache Spark

**Feature Store**
- Hopworks Feature Store

**Distributed ML & DL**
- Pip install Conda libraries
- Tensorflow
- scikit-learn
- Keras
- Jupyter Notebooks
- Tensorboard

**Serving**
- Kubernetes

**Streaming**
- Apache Beam
- Apache Spark
- Apache Flink

**Monitoring**
- Spark Streaming

**Filesystem and Metadata storage**
- HopsFS

**Encrypt everything**
- TSL/SSL encrypted calls between services with X.509 certificates

**Secure Collaboration**
- Multi-Tenancy with Project-based collaboration and resource management

**Data Lake Support**
- Integrates with your existing Data Lake or acts as your Data Lake

Logical Clocks
What is Hopsworks?

**Elasticity & Performance**
- **Feature Store**
  - Data warehouse for ML
- **Distributed Deep Learning**
  - Faster with more GPUs
- **HopsFS**
  - NVMe speed with Big Data
- **Horizontally Scalable**
  - Ingestion, DataPrep, Training, Serving

**Development & Operations**
- **Notebooks for Development**
  - First-class Python Support
- **Version Everything**
  - Code, Infrastructure, Data
- **Model Serving on Kubernetes**
  - TF Serving, MLeap, SkLearn
- **End-to-End ML Pipelines**
  - Orchestrated by Airflow

**Governance & Compliance**
- **Secure Multi-Tenancy**
  - Project-based restricted access
- **Encryption At-Rest, In-Motion**
  - TLS/SSL everywhere
- **AI-Asset Governance**
  - Models, experiments, data, GPUs
- **Data/Model/Feature Lineage**
  - Discover/track dependencies
Machine Learning Pipelines
TensorFlow Extended (TFX) ML Pipeline

https://www.tensorflow.org/tfx
End-to-End ML Pipelines

Raw Data
Event Data
Data Lake

Data Ingest → Data Prep → Train → Serve

Distributed Storage

Online Monitor
Horizontally Scalable End-to-End ML Pipelines

Data Pipelines → Feature Store → Machine Learning Experiments (Hyperparam Optimization, Ablation Studies) → Data Parallel Training → Model Serving

Share, Collaborate, Align

>100X Productivity

Data Team

Data Scientist

>100X Productivity

Machine Learning Experiments

Horizontally Scalability needed at each Stage of the Pipeline
End-to-End ML Pipelines in Hopsworks

- Data Ingest
- Data Prep
- Train
- Serve

Raw Data

Event Data

Data Lake

Amazon S3

CloudData

Online Monitor
End-to-End ML Pipelines in Hopsworks
TensorFlow Extended Components in Hopsworks
Apache Airflow to Orchestrate ML Pipelines
Apache Airflow to Orchestrate ML Pipelines

- Jobs REST API

Hopsworks Jobs: PySpark, Spark, Flink, Beam/Flink
ML Pipelines with a Feature Store
ML Pipelines with a Feature Store
These should be based on the same feature engineering code.
It’s not always trivial to ensure features are engineered consistently between training and inference.
Feature Store

Training

Feature Store

Put

Get

Features + Labels → Model

Inference

Get

Features + Model → Labels
Feature Store

Training

- Features + Labels → Model

Put

Feature Store

Get

Online or Offline Features? On-Demand or Cached Features?

Inference

Features + Model → Labels

Get

Batch App

Online App
Hopsworks Feature Store

**Data Engineer**
- Add/remove features, access control, feature data validation.

**External DB Feature Defn**
- "select .."

**Pandas or PySpark DataFrame**

**Feature Mgmt**
- Statistics
- Access Control
- Feature CRUD
- Feature Data Ingestion
- Data Validation

**Storage**
- MySQL Cluster (Metadata, Online Features)
- Apache Hive Columnar DB (Offline Features)
- HopsFS

**Access**
- Discovery
- Models
- Online Features
- Time Travel
- Offline Features

**Data Scientist**
- Discover features, create training data, save models, read online/offline/on-demand features, historical feature values.

**Online Apps**

**Batch Apps**

**Training Data (S3, HDFS)**

**JDBC (SAS, R, etc)**

**Discover features, create training data, save models, read online/offline/on-demand features, historical feature values.**

**Models**

**Batch Apps**

**Training Data (S3, HDFS)**

**JDBC (SAS, R, etc)**

**Discover features, create training data, save models, read online/offline/on-demand features, historical feature values.**
Features, FeatureGroups, and Train/Test Datasets are versioned
Register a Feature Group with the Feature Store

titanic_df =  # Spark or Pandas Dataframe

# Do feature engineering on ‘titanic_data’

# Register Dataframe as FeatureGroup
featurestore.create_featuregroup(titanic_df,
"titanic_data_passengers“)
Create Training Datasets using the Feature Store

```python
sample_data = featurestore.get_features(["name", "Pclass", "Sex", "balance"])  
featurestore.create_training_dataset(sample_data, "titanic_training_dataset", data_format="tfrecords", training_dataset_version=1)

# Use the training dataset
dataset_dir = featurestore.get_training_dataset_dataset_path("titanic_training_dataset")
s = featurestore.get_training_dataset_tf_record_schema("titanic_training_dataset")
```
What is all this talk about metadata and provenance?
Explicit vs Implicit Provenance

- **Explicit**: TFX, MLFlow
  - Wrap existing code in components that execute a stage in the pipeline
  - Interceptors in components inject metadata to a metadata store as data flows through the pipeline
  - Store metadata about artefacts and executions

- **Implicit**: Hopworks with ePipe*

  ![Diagram showing data flow and metadata storage]

  - DataPrep
  - Experiment
  - Train
  - /Training Datasets
  - /Experiments
  - /Models
  - HopsFS
  - ePipe: ChangeDataCapture API
  - Elasticsearch

Provenance in Hopsworks

● Implicit Provenance
  ○ Applications read/write files to HopsFS – infer artefacts from pathname conventions and xattrs

● Explicit Provenance in Hops API
  ○ Executions:
    hops.experiment.grid_search(...), hops.experiment.collective_allreduce(...)
  ○ FeatureStore:
    hops.featurestore.create_training_dataset(...)
  ○ Saving and deploying models:
    hops.serving.export(export_path, “model_name", 1)
    hops.serving.deploy(“.../model.pb", destination)
/Experiments

- Executions add entries in /Experiments:
  - experiment.launch(…)
  - experiment.grid_search(…)
  - experiment.collective_allreduce(…)
  - experiment.lagom(…)

- /Experiments contains:
  - logs (application, tensorboard)
  - executed notebook file
  - conda environment used
  - checkpoints

/Projects/MyProj
  └ Experiments
    └ <app_id>
      └ <type>
        ├─ checkpoints
        ├─ tensorboard_logs
        └ logfile
          └ versioned_resources
            └ notebook.ipynb
            └ conda_env.yml
<table>
<thead>
<tr>
<th>Name</th>
<th>Metric</th>
<th>User</th>
<th>Start</th>
<th>End</th>
<th>State</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>mnist serving example</td>
<td></td>
<td>Admin Admin</td>
<td>2019-08-02T16:06:34.514</td>
<td>2019-08-02T16:06:57.554</td>
<td>FINISHED</td>
<td></td>
</tr>
<tr>
<td>fashion mnist grid search</td>
<td>0.7054688</td>
<td>Admin Admin</td>
<td>2019-08-02T15:12:12.674</td>
<td>2019-08-02T15:18:02.916</td>
<td>FINISHED</td>
<td></td>
</tr>
<tr>
<td>fashion mnist autoML</td>
<td>0.7359375</td>
<td>Admin Admin</td>
<td>2019-08-02T15:56:48.977</td>
<td>2019-08-02T16:04:34.243</td>
<td>FINISHED</td>
<td></td>
</tr>
</tbody>
</table>
### Individual Experiment Overview

#### Experiment - fashion mnist autoML - Detailed Information

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>application_1564752881619_0006_1</td>
</tr>
<tr>
<td>Name</td>
<td>fashion mnist autoML</td>
</tr>
<tr>
<td>Metric</td>
<td>0.7359375</td>
</tr>
<tr>
<td>User</td>
<td>Admin Admin</td>
</tr>
<tr>
<td>Started</td>
<td>2019-08-02T15:56:48.977</td>
</tr>
<tr>
<td>Finished</td>
<td>2019-08-02T16:04:34.243</td>
</tr>
<tr>
<td>ExperimentType</td>
<td>PARALLEL_EXPERIMENTS</td>
</tr>
<tr>
<td>Function</td>
<td>differential_evolution</td>
</tr>
<tr>
<td>Direction</td>
<td>max</td>
</tr>
<tr>
<td>Optimization key</td>
<td>accuracy</td>
</tr>
<tr>
<td>State</td>
<td>FINISHED</td>
</tr>
</tbody>
</table>

#### Hyperparameters

<table>
<thead>
<tr>
<th>learning_rate</th>
<th>dropout</th>
<th>accuracy</th>
<th>loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.001</td>
<td>0.7674655085132838</td>
<td>0.7359375</td>
<td>0.8713156</td>
</tr>
<tr>
<td>0.001</td>
<td>0.8</td>
<td>0.725</td>
<td>0.91851103</td>
</tr>
<tr>
<td>0.001</td>
<td>0.7217845218609694</td>
<td>0.71171874</td>
<td>0.884253</td>
</tr>
<tr>
<td>0.001</td>
<td>0.7777123179779686</td>
<td>0.70703125</td>
<td>0.9578301</td>
</tr>
<tr>
<td>0.001</td>
<td>0.555320974435649</td>
<td>0.69609374</td>
<td>0.78994834</td>
</tr>
<tr>
<td>0.001</td>
<td>0.353423249822699</td>
<td>0.690625</td>
<td>0.81624717</td>
</tr>
<tr>
<td>0.0001</td>
<td>0.3462998450903566</td>
<td>0.67265624</td>
<td>1.5414852</td>
</tr>
<tr>
<td>0.0001</td>
<td>0.7548718246518114</td>
<td>0.6171875</td>
<td>1.832791</td>
</tr>
<tr>
<td>0.0001</td>
<td>0.514763392515677</td>
<td>0.6070312</td>
<td>1.666979</td>
</tr>
<tr>
<td>0.0001</td>
<td>0.7395016104547842</td>
<td>0.5960938</td>
<td>1.9024662</td>
</tr>
<tr>
<td>0.0001</td>
<td>0.8</td>
<td>0.58984375</td>
<td>1.9473743</td>
</tr>
<tr>
<td>0.0001</td>
<td>0.7674655085132838</td>
<td>0.57109374</td>
<td>1.8521827</td>
</tr>
</tbody>
</table>

#### Input files

- autoML_fashion_mnist.ipynb
Models

- Named/versioned model management for: TensorFlow/Keras, Scikit Learn
- A Models dataset can be securely shared with other projects or the whole cluster
- The provenance API returns the conda.yml and execution used to train a given model
Model Serving

- TensorFlow Serving, Scikit-Learn
- Serve models on Kubernetes
Jupyter Notebooks as Jobs in Airflow Pipelines
Principles of Development with Notebooks

- No throwaway code
- Code can be run either in Notebooks or as Jobs (in Pipelines)
- Notebooks/jobs should be parameterizable
- No external configuration for program logic
  - HPARAMs are part of the program
- Core training loop should be the same for all stages of development
  - Data exploration, hparam tuning, dist training,
Problem in ML Pipeline Development Process?

Explore Data, Train model → Hyperparam Opt. → Distributed Training

Notebook → Python + YML in Git

port/rewrite code

Iteration is hard/impossible/a-bad-idea
Notebooks offer more than Prototyping

- Familiar web-based development environment
- Interactive development/debugging
- Reporting platform for ML applications (Papermill by Netflix)
- Parameterizable as part of Pipelines (Papermill by Netflix)

Disclaimer: Notebooks are not for everyone
ML Pipelines of Jupyter Notebooks with Airflow

Dataprep Pipeline

Feature Engineering → Feature Store

Training and Deployment Pipeline

Select Features, File Format → Experiment, Train Model → Validate & Deploy Model
PySpark Notebooks as Jobs in ML Pipelines
Running TensorFlow/Keras/PyTorch Apps in PySpark

Warning: micro-exposure to PySpark may cure you of distributed programming phobia
End-to-End ML Pipelines in Hopsworks
PySpark makes it easier to write TensorFlow/Keras/PyTorch code that can either be run on a single GPU or scale to run on lots of GPUs for Parallel Experiments or Distributed Training.
Need Distributed Filesystem for Coordination

- Training/Test Datasets
- Model checkpoints, Trained Models
- Experiment run data
- Provenance data
- Application logs
def executor():
    print("Hello from GPU")

from hops import experiment
experiment.launch(executor)
PySpark – Hello World

In [ ]:

def executor():
    print("Hello from GPU")

In [ ]:

from hops import experiment
experiment.launch(executor)
Leave code unchanged, but configure 4 Executors

```
print("Hello from GPU")
print("Hello from GPU")
print("Hello from GPU")
print("Hello from GPU")
```

**Driver**

- **Hours to shutdown**: 6
- **Driver memory (MB)**: 2048
- **Distribution strategy**: COLLECTIVE_ALL_REDUCE
- **Workers**: 4
- **Executor memory (MB)**: 4096
- **Number GPUs per worker**: 1
Driver with 4 Executors

In [ ]:
```python
def executor():
    print("Hello from GPU")
```

In [ ]:
```python
def executor():
    print("Hello from GPU")
```

In [ ]:
```python
def executor():
    print("Hello from GPU")
```

In [ ]:
```python
def executor():
    print("Hello from GPU")
```

In [ ]:
```python
from hops import experiment
experiment.launch(executor)
```
Same/Replica Conda Environment on all Executors

```python
In [ ]:
def executor():
    print(“Hello from GPU”)
conda env

In [ ]:
def executor():
    print(“Hello from GPU”)
conda env

In [ ]:
def executor():
    print(“Hello from GPU”)
conda env

In [ ]:
def executor():
    print(“Hello from GPU”)
conda env

In [ ]:
from hops import experiment
experiment.launch(executor)
conda env
```
# A Conda Environment Per Project in Hopsworks

## Uninstall/Upgrade Python Libraries

<table>
<thead>
<tr>
<th>Url</th>
<th>Library</th>
<th>Version</th>
<th>Package Manager</th>
<th>MachineType</th>
<th>Status</th>
<th>User-Installed</th>
</tr>
</thead>
<tbody>
<tr>
<td>PyPi</td>
<td>hopsfacets</td>
<td>0.0.3</td>
<td>PIP</td>
<td>ALL</td>
<td>SUCCESS</td>
<td>Uninstall</td>
</tr>
<tr>
<td>PyPi</td>
<td>pandas</td>
<td>0.23.1</td>
<td>PIP</td>
<td>ALL</td>
<td>SUCCESS</td>
<td>Uninstall</td>
</tr>
<tr>
<td>PyPi</td>
<td>mmlspark</td>
<td>0.13</td>
<td>PIP</td>
<td>ALL</td>
<td>SUCCESS</td>
<td>Uninstall</td>
</tr>
<tr>
<td>PyPi</td>
<td>numpy</td>
<td>1.15.3</td>
<td>PIP</td>
<td>ALL</td>
<td>SUCCESS</td>
<td>Uninstall</td>
</tr>
<tr>
<td>PyPi</td>
<td>hops</td>
<td>0.6.0.1</td>
<td>PIP</td>
<td>ALL</td>
<td>SUCCESS</td>
<td>Uninstall</td>
</tr>
<tr>
<td>PyPi</td>
<td>pydoop</td>
<td>2.0a3</td>
<td>PIP</td>
<td>ALL</td>
<td>SUCCESS</td>
<td>Pre-installed</td>
</tr>
<tr>
<td>PyPi</td>
<td>tensorboard</td>
<td>1.11.0</td>
<td>PIP</td>
<td>ALL</td>
<td>SUCCESS</td>
<td>Pre-installed</td>
</tr>
<tr>
<td>PyPi</td>
<td>tensorflow</td>
<td>1.11.0</td>
<td>PIP</td>
<td>CPU</td>
<td>SUCCESS</td>
<td>Pre-installed</td>
</tr>
<tr>
<td>PyPi</td>
<td>tensorflow-gpu</td>
<td>1.11.0</td>
<td>PIP</td>
<td>GPU</td>
<td>SUCCESS</td>
<td>Pre-installed</td>
</tr>
</tbody>
</table>
Use Pip or Conda to install Python libraries

Install Python libraries using pip in Anaconda environment

Python Version is 3.6

Installation mode

- All machines
- CPU machines
- GPU machines

Pillow-PIL

Version: 0.1dev

Not Installed

Install
def train():
    # Separate shard of dataset per worker
    # create Estimator w/ DistribStrategy
    # as CollectiveAllReduce
    # train model, evaluate
    return loss

# Driver code below here
# builds TF_CONFIG and shares to workers
from hops import experiment
experiment.collective_allreduce(train)

More details: http://github.com/logicalclocks/hops-examples
def train(dropout):
    # Same dataset for all workers
    # create model and optimizer
    # add this worker’s value of dropout
    # train model and evaluate
    return loss

# Driver code below here
from hops import experiment
args={"dropout":[0.1, 0.4, 0.8]}
experiment.grid_search(train, args)

More details: http//github.com/logicalclocks/hops-examples
def train(dropout):
    # Same dataset for all workers
    # create model and optimizer
    optimizer.apply(dropout)
    # train model and evaluate
    return loss

from hops import experiment
args = {"dropout": "0.1-0.8"}
experiment.diff_ev(train, args)

More details: http://github.com/logicalclocks/hops-examples
Wasted Compute!

N spark tasks → N eval metrics

Driver/Parameter Server

Waiting... wasted compute
Parallel ML Experiments with PySpark and Maggy
Iterative Model Development

<table>
<thead>
<tr>
<th>name</th>
<th>PClass</th>
<th>Sex</th>
<th>survive</th>
</tr>
</thead>
</table>

Problem Definition
Data Preparation
Model Selection
Hyperparameters

Model Training

Dataset
Machine Learning Model
Optimizer

Evaluate

Repeat if needed
Maggy: Unified Hparam Opt & Ablation Programming

Machine Learning System

Evaluate

Ablation Study Controller

New Hyperparameter Values
New Dataset/Model-Architecture

Synchronous or Asynchronous Trials
Directed or Undirected Search
User-Defined Search/Optimizers
Maggy – Async, Parallel ML experiments using PySpark

- Experiment-Driven, interactive development of ML applications
- Parallel Experimentation
  - Hyperparameter Optimization
  - Ablation Studies
  - Leave-one-Feature-out
- Interactive Debugging
  - Experiment/executor logs shown both in Jupyter notebooks and logging
Interactive Debugging: Print Executor Logs in Jupyter
SparkSession available as 'spark'.
You are running maggy on Hopworks.

```
In [2]: def train(kernel, pool, dropout, reporter):
    batch_size = 512
    num_classes = 10
    epochs = 1
    img_rows, img_cols = 28, 28
    (x_train, y_train), (x_test, y_test) = mnist.load_data()
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)
    x_train = x_train.astype('float32')
    x_test = x_test.astype('float32')
    x_train /= 255
    x_test /= 255
    y_train = keras.utils.to_categorical(y_train, num_classes)
    y_test = keras.utils.to_categorical(y_test, num_classes)
    model = Sequential()
    model.add(Conv2D(32, kernel_size=(kernel, kernel), activation='relu', input_shape=input_shape))
    model.add(MaxPooling2D(pool_size=(pool, pool)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(dropout))
    model.add(Dense(num_classes, activation='softmax'))
    opt = keras.optimizers.Adamax(1.0)
    model.compile(loss=keras.losses.categorical_crossentropy, optimizer=opt, metrics=['accuracy'])
    callbacks = [keras.callbacks.ModelCheckpoint('model.h5')]  
    model.fit(x_train, y_train, batch_size=batch_size, callbacks=callbacks, epochs=epochs, validation_data=(x_test, y_test))
    score = model.evaluate(x_test, y_test, verbose=0)
    print('Accuracy:', score[1])
    return score[1]

In [4]: sp = Searchespace(kernel=('INTEGER', [2, 8]), pool=('INTEGER', [2, 8]), dropout=('DOUBLE', [0.01, 0.99]))
experiment.log('train', sp, 'randomsearch', direction='max', num_trials=15, name='MNIST', es_interval=300, es_min=5)
```

Maggy experiment 0% 0/15 [02:00<?], trials/early stopped=0, best metric=n.a.

0: Train on 60000 samples, validate on 10000 samples
1: Train on 60000 samples, validate on 10000 samples
2: Train on 60000 samples, validate on 20000 samples
3: Train on 60000 samples, validate on 10000 samples
That was Hopsworks

**Efficiency & Performance**
- Feature Store
  - Data warehouse for ML
- Distributed Deep Learning
  - Faster with more GPUs
- HopsFS
  - NVMe speed with Big Data
- Horizontally Scalable
  - Ingestion, DataPrep, Training, Serving

**Development & Operations**
- Development Environment
  - First-class Python Support
- Version Everything
  - Code, Infrastructure, Data
- Model Serving on Kubernetes
  - TF Serving, SkLearn
- End-to-End ML Pipelines
  - Orchestrated by Airflow

**Security & Governance**
- Secure Multi-Tenancy
  - Project-based restricted access
- Encryption At-Rest, In-Motion
  - TLS/SSL everywhere
- AI-Asset Governance
  - Models, experiments, data, GPUs
- Data/Model/Feature Lineage
  - Discover/track dependencies
Hopworks: Horizontally Scalable ML Pipelines in Python

- **Raw Data**
  - Ingest
  - Data Prep
  - Feature Store

- **Event Data**
  - Flink
  - HopsFS

- **Experiment/Train**
  - Spark
  - Feature Store

- **Deploy**
  - Serving
  - Monitor

- **Airflow**
  - Monitor
  - logs

- **Metadata Store**
  - News
  - Spark Streaming

- **Other Tools**
  - Amazon S3
  - Cloudera
  - Kafka

- **Python Libraries**
  - HopsFS
  - Feature Store
Hopsworks 1.0 coming soon

Hopsworks-1.0
- Beam 2.14.0
- Flink 1.8.1
- Spark 2.4.3
- TensorFlow 1.14.0
- TFX 0.13
- TensorFlow Model Analysis 0.13.2
- PyTorch 1.1
- ROCm 2.6, Cuda 10.X
Acknowledgements and References

Slides and Diagrams from colleagues:
● Maggy: Moritz Meister and Sina Sheikholeslami
● Feature Store: Kim Hammar
● Beam/Flink on Hopsworks: Theofilos Kakantousis

References
● Hopsworks Demo, SysML 2019.
Thank you!

470 Ramona St
Palo Alto
https://www.logicalclocks.com
Register for a free account at
www.hops.site

Twitter
@logicalclocks
@hopswork

GitHub
https://github.com/logicalclocks/hopsworks
https://github.com/hopshadoop/hops