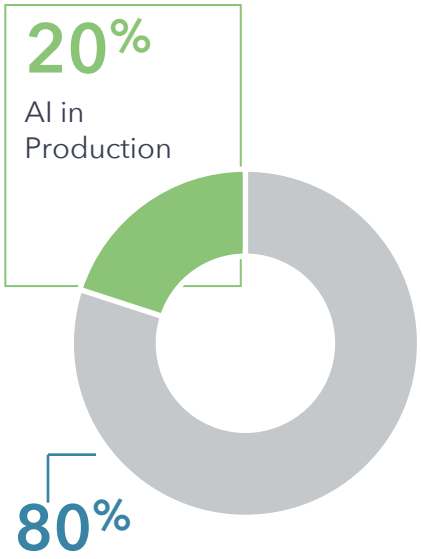
ParallelM

Operationalizing Edge Machine Learning with Apache Spark

ParallelM

Growing AI Investments; Few Deployed at Scale



Survey of 3073 AI-aware C-level Executives

Out of 160 reviewed AI use cases:

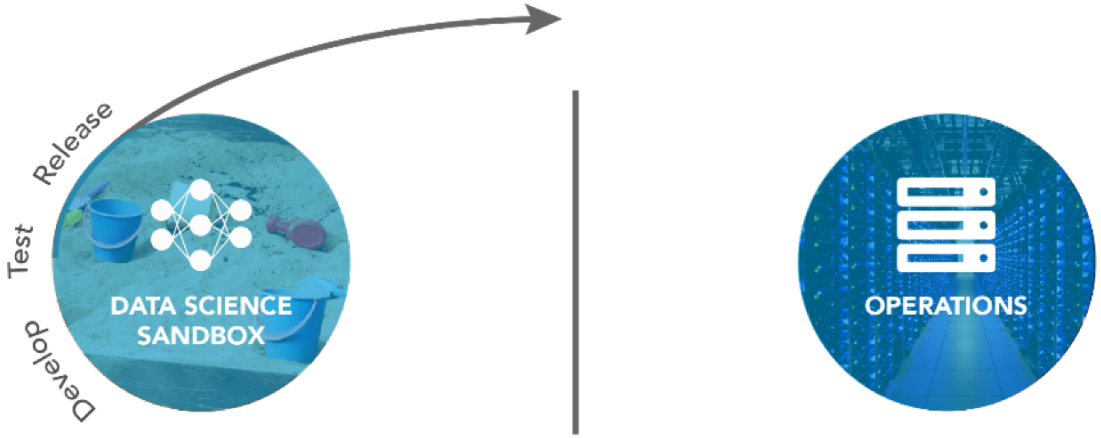
88% did not progress beyond the experimental stage

But successful early AI adopters report:

Profit margins **3–15%** higher than industry average

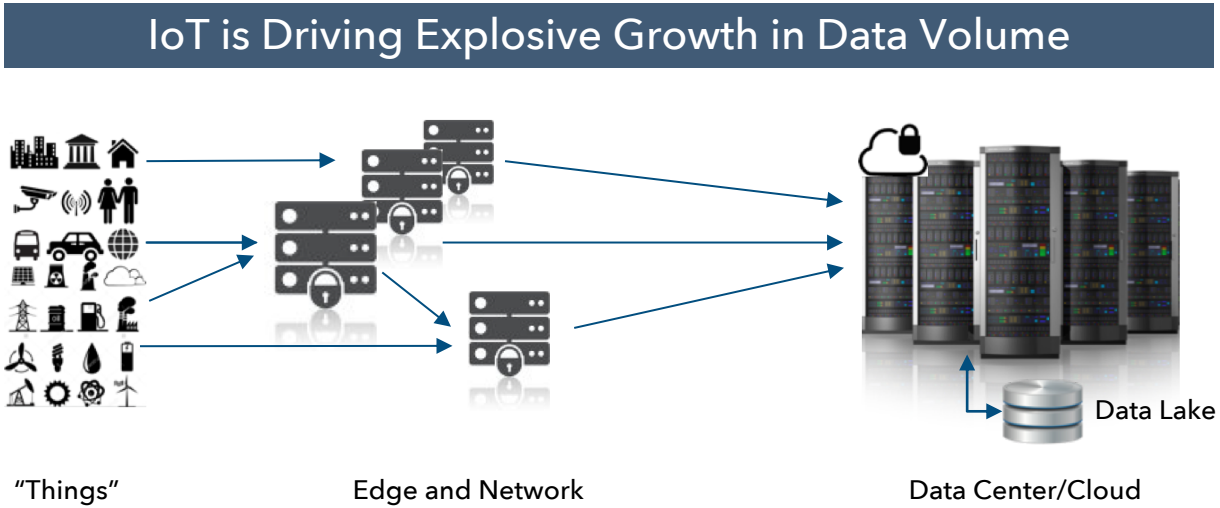
Source: “Artificial Intelligence: The Next Digital Frontier?”, McKinsey Global Institute, June 2017

Challenges of Deploying & Managing ML in Production



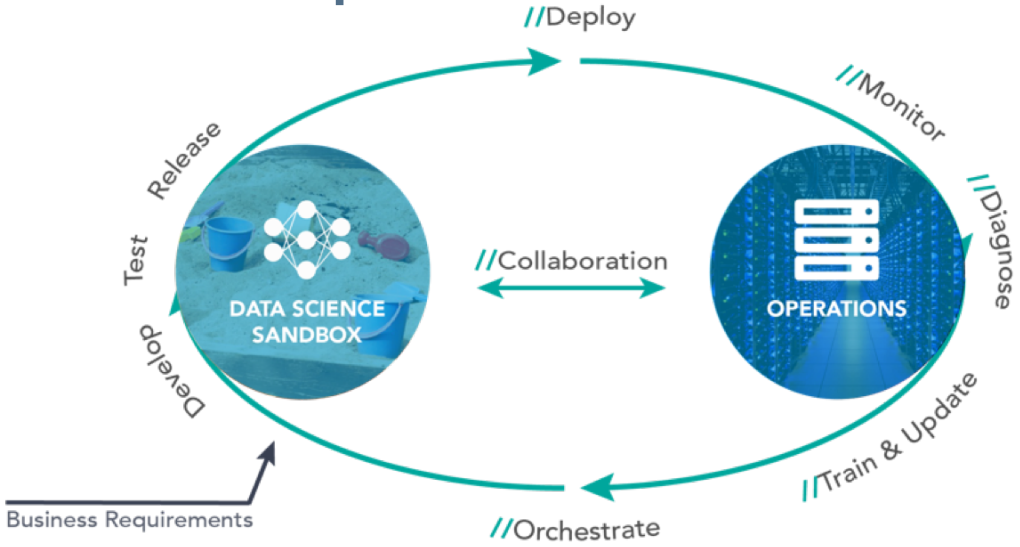
- Diverse focus and expertise of Data Science & Ops teams
- Increased risk from non-deterministic nature of ML
- Current Operations solutions do not address uniqueness of ML Apps

Challenges of Edge/Distributed Topologies



- Varied resources at each level
- Scale, heterogeneity, disconnected operation

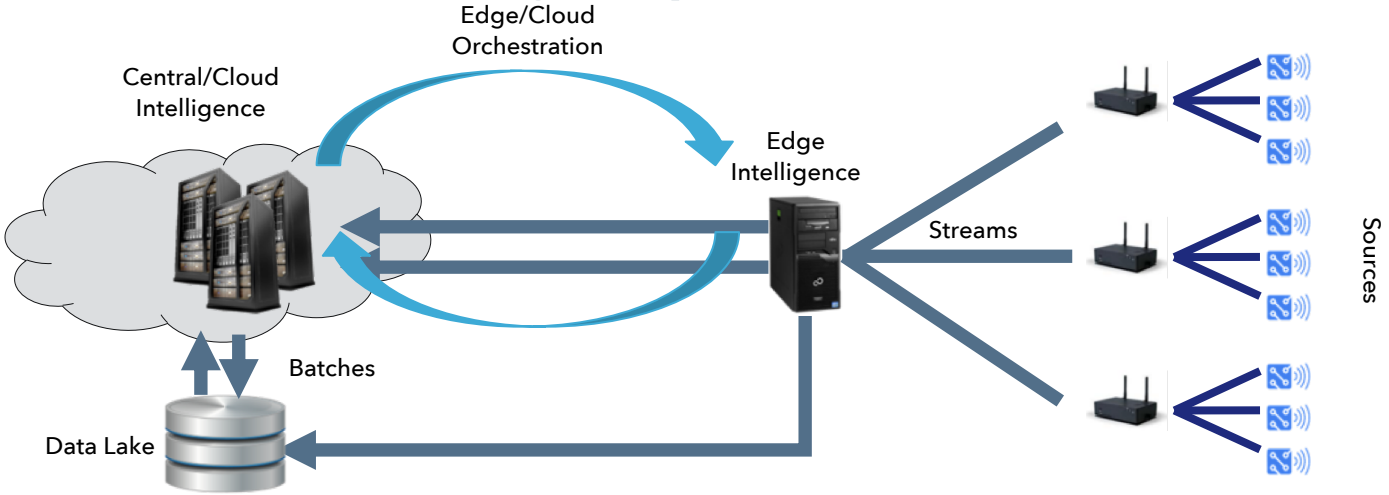
What We Need For Operational ML



- Accelerate deployment & facilitate collaboration between Data & Ops teams
- Monitor validity of ML predictions, diagnose data and ML performance issues
- Orchestrate training, update, and configuration of ML pipelines across distributed, heterogeneous infrastructure with tracking

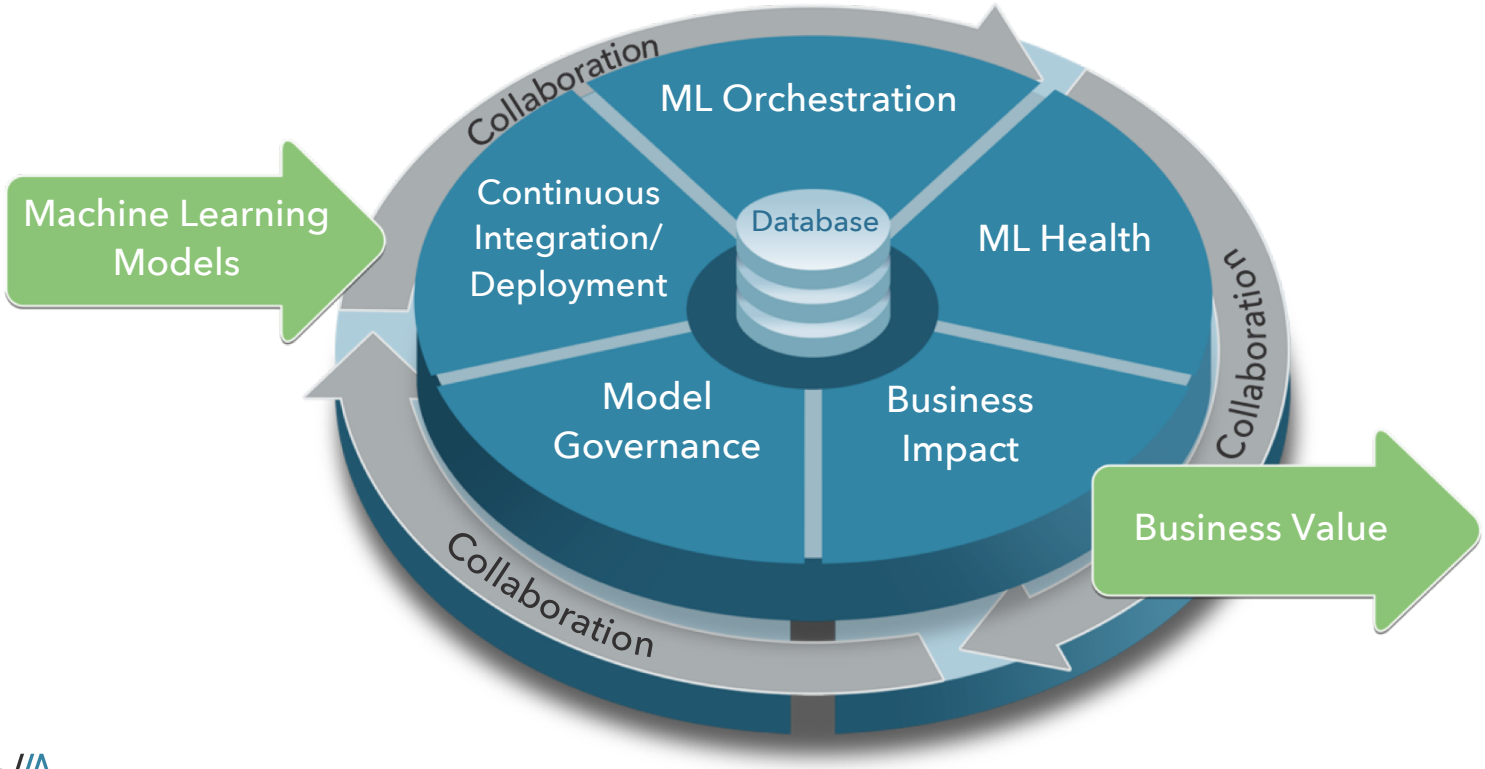
Parallel//\

What We Need For Edge Operational ML

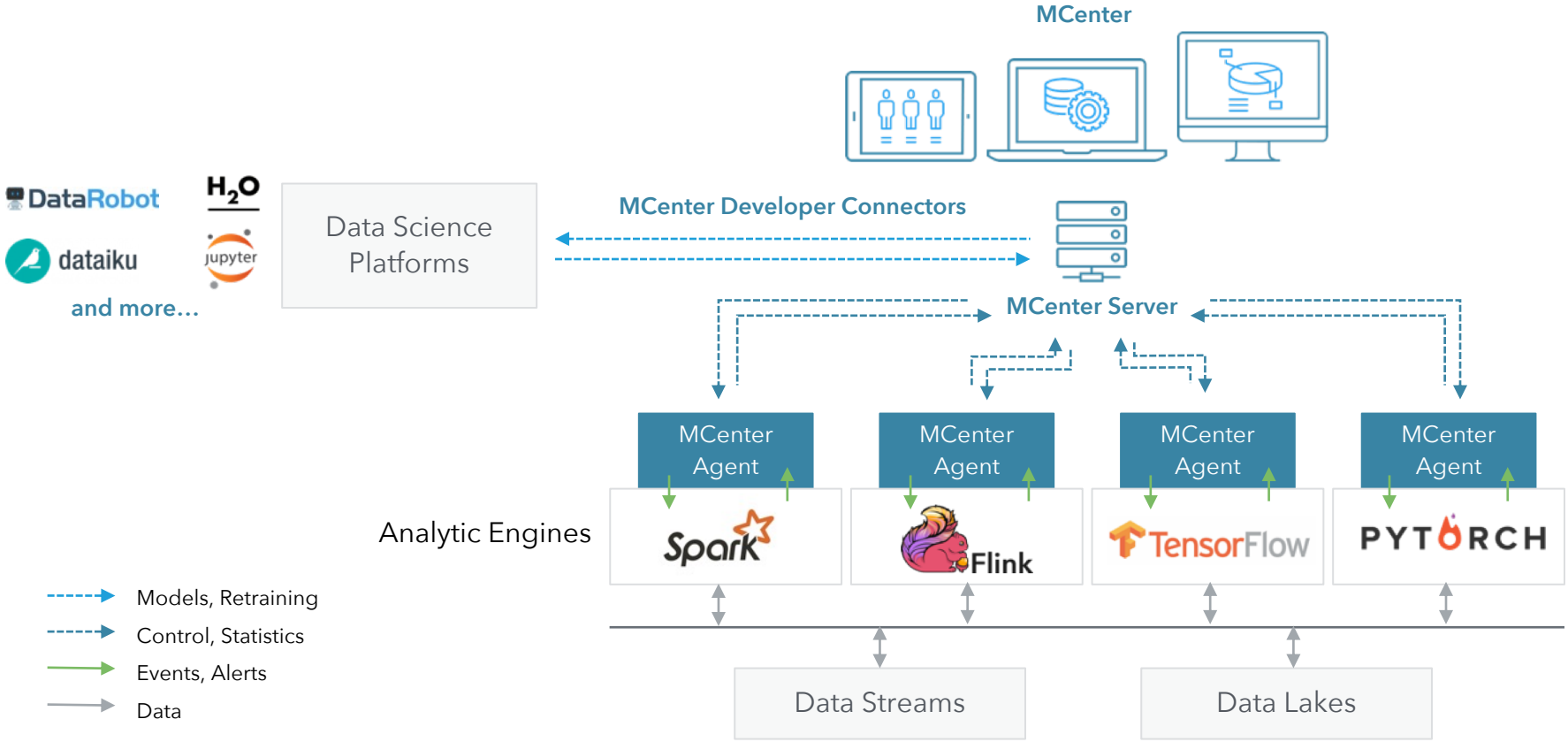


- Distribute analytics processing to the optimal point for each use case
- Flexible management framework enables:
  - Secure centralized and/or local learning, prediction, or combined learning/prediction
  - Granular monitoring and control of model update policies
- Support multi-layer topologies to achieve maximum scale while accommodating low bandwidth or unreliable connectivity

MLOps – Managing the full Production ML Lifecycle

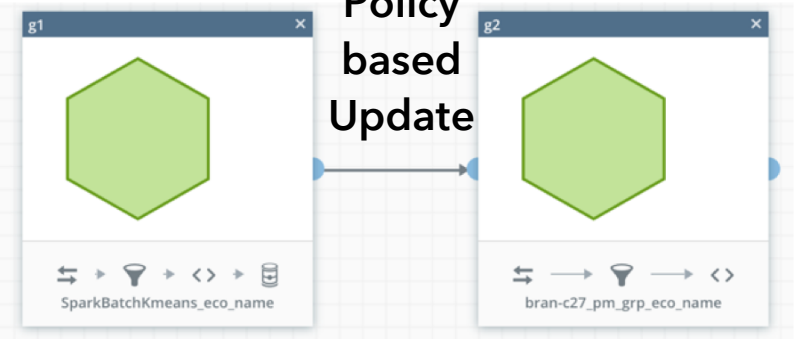


Our Approach



Operational Abstraction

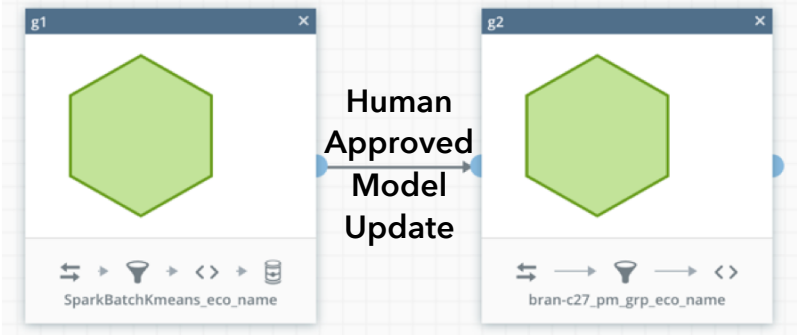
- Link pipelines (training and inference) via an ION (Intelligence Overlay Network)
- Basically a Directed Graph representation with allowance for cycles
- Pipelines are DAGs within each engine
- Distributed execution over heterogeneous engines, programming languages and geographies



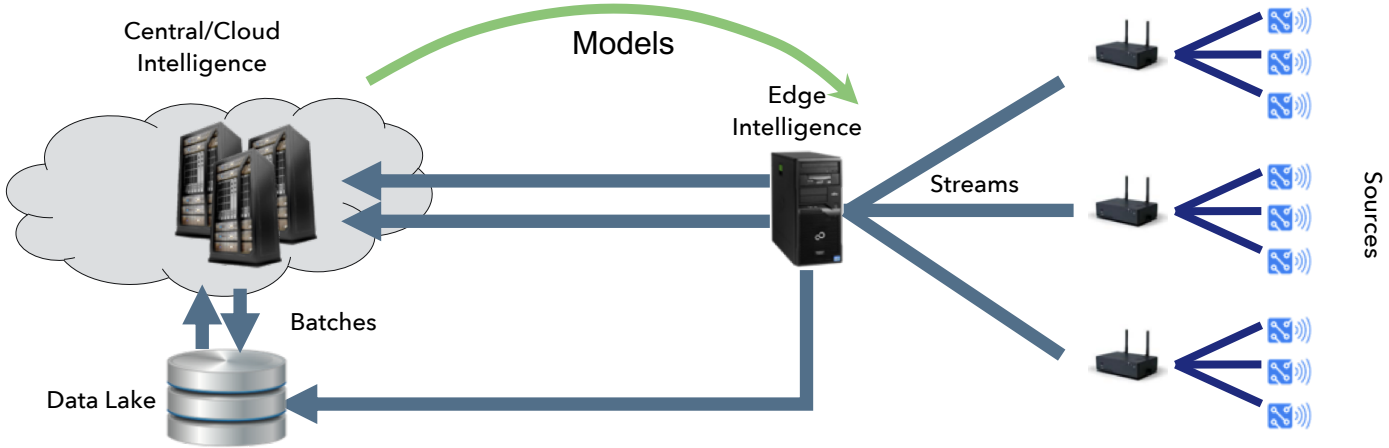
Example – KMeans Batch Training Plus Streaming Inference Anomaly Detection

An Example ION to Resource Mapping

Every Tuesday at 10AM - Cloud



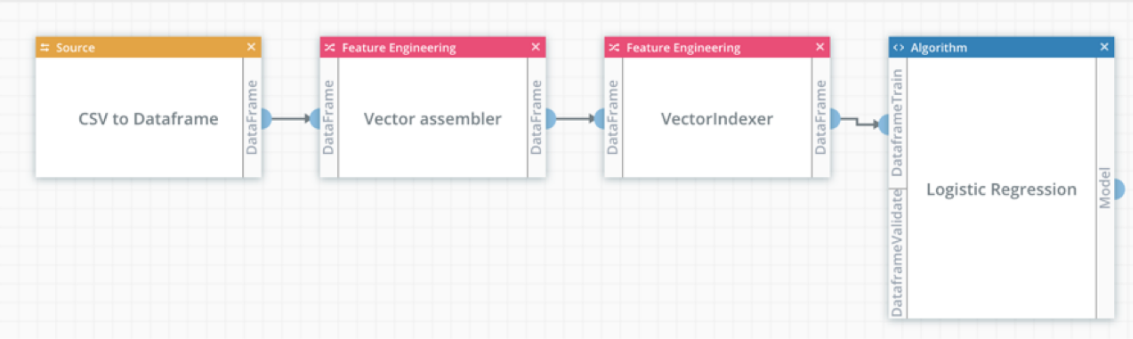
Every 5 min - Edge



Pipeline Examples

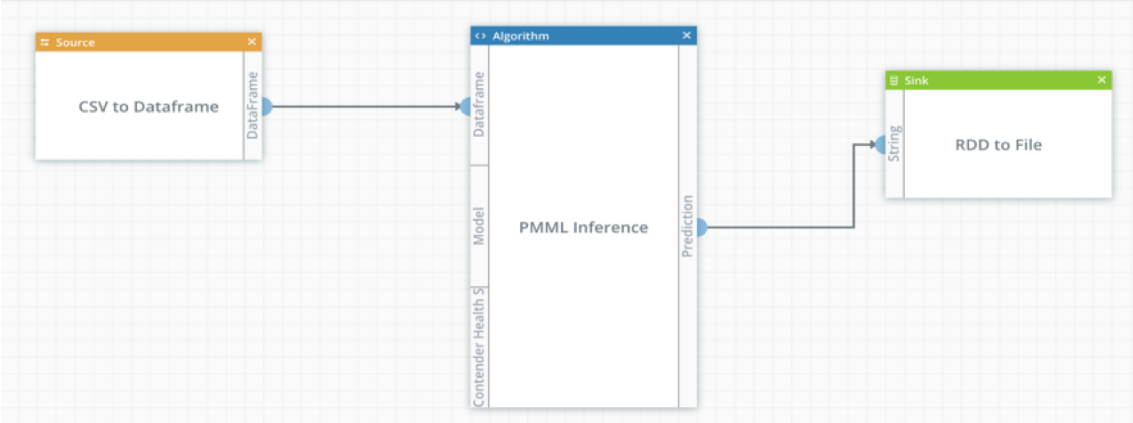
Training Pipeline

(SparkML)

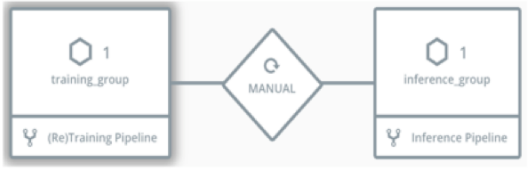


Inference Pipeline

(SparkML)



Instrument, Upload, Orchestrate, Monitor



JSON

```
{
 "workflow": [
   {
     "groupId": "1342",
     "pipelineId": "42154",
     "pipelineType": "Training",
     "id": "1",
     "pipelineMode": "offline",
     "schedule": "* * 0/12 ? * *",
     "templateId": "32",
     . . .
   },
   {. . .}
 ],
 "modelPolicy": "Manual Approval",
 "name": "Fraud Detection",
 "profileId": "19"
}
```

(a) ION Template

Inference Pipeline

(Re) Training Pipeline

Policy

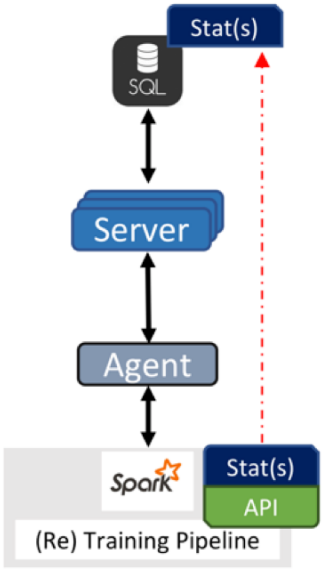
PySpark

```
  5 import pyspark
  6 from mlops import MLAnalytics as mla

145 # Train & Evaluate
146 modelRandForest=Pipeline(stages=fullPipe).fit(input_train)
147 predictions = modelRF.transform(input_test)
160 rmse_eval= RegressionEvaluator(…).evaluate(predictions)
161 mla.stat("RMSE", rmse_eval, st.TIME_SERIES
180 mlt = MultiLineGraph().name("G1").labels([…]).data([…])
181 mla.stat(mlt)

230 # Table
231 for j in range (0,len(maxDepthRange)):
232     col_name.append(str(maxDepthRange[j]))
233 tbl = Table().name("HyperParams Depth").cols(col_name)
234 tbl.add_row("RMSE:", ["%.2f" % x for x in rmse_array])
235 tbl.add_row("R2:", […])    pm.stat(tbl)
236 mla.stat(tbl)
```

(b) Code w/ API Instrumentation



(c) Pipeline at Runtime

Integrating with Analytics Engines (Spark)

Job Management

- Via *SparkLauncher*: A library to control launching, monitoring and terminating jobs
  - PM Agent communicates with Spark through this library for job management (also uses Java API to launch child processes)

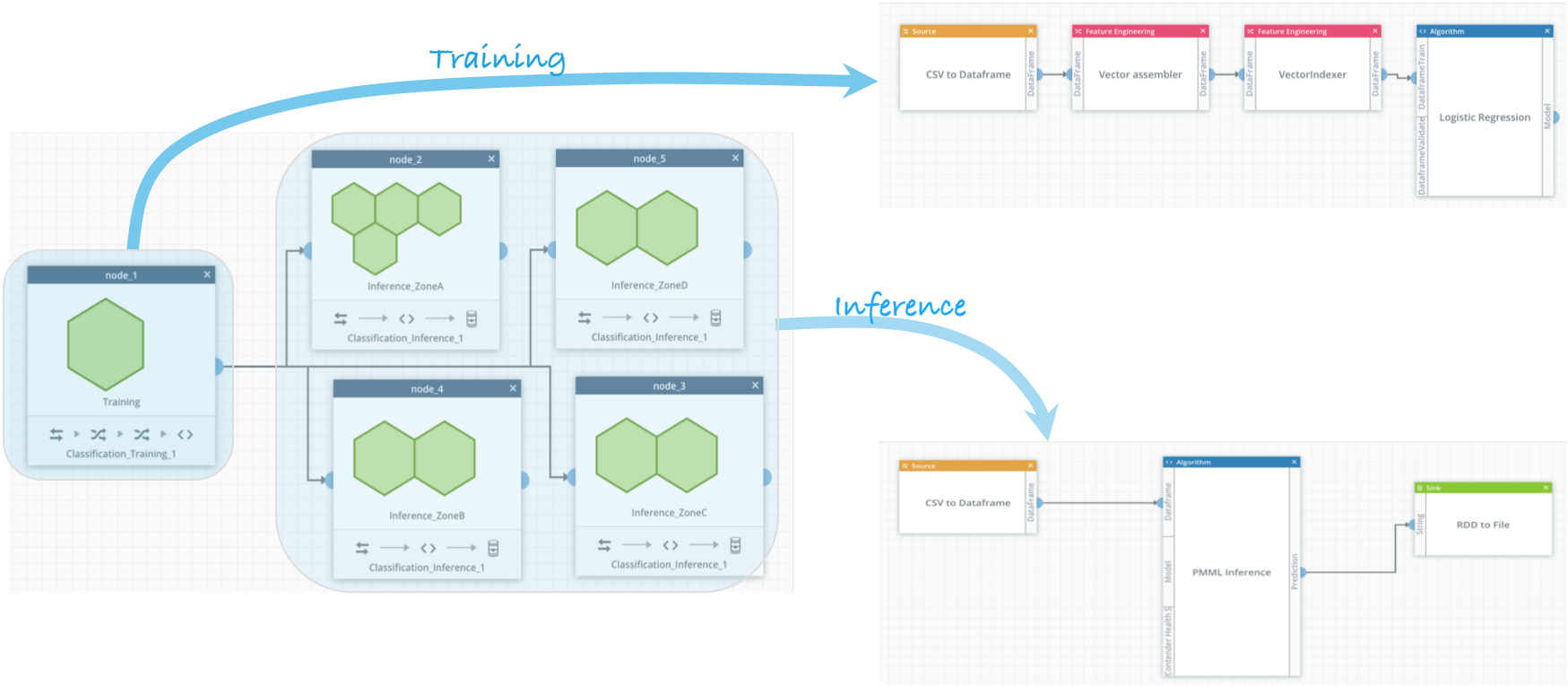
Statistics

- Via *SparkListener*: A Spark-driver callback service
- *SparkListener* taps into all accumulators which, is one of the popular ways to expose statistics
- PM agent communicates with the Spark driver and exposes statistics via a REST endpoint

ML Health / Model collection and updates

- PM Agent delivers and receives health events, health objects and models via sockets from custom PM components in the ML Pipeline

Demo Description



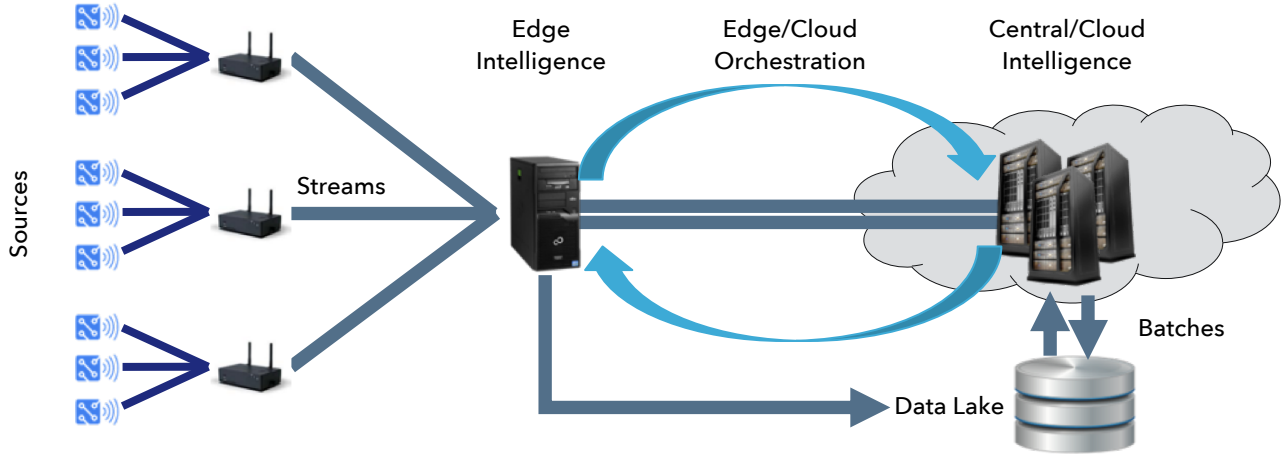
Thank You!

Nisha Talagala
nisha.talagala@parallelm.com

Vinay Sridhar

vinay.sridhar@parallelm.com

What We Need For Edge Operational ML



- Distribute analytics processing to the optimal point for each use case
- Flexible management framework enables:
  - Secure centralized and/or local learning, prediction, or combined learning/prediction
  - Granular monitoring and control of model update policies
- Support multi-layer topologies to achieve maximum scale while accommodating low bandwidth or unreliable connectivity

Integrating with Analytics Engines (TensorFlow)

Job Management

- TensorFlow Python programs run as standalone applications
- Standard process control mechanisms based on the OS is used to monitor and control TensorFlow programs

Statistics Collection

- PM Agent parses contents via *TensorBoard* log files to extract meaningful statistics and events that data scientists added

ML Health / Model collection

- Generation of models and health objects is recorded on a shared medium

An Example ION