Mímir: Building and Deploying an ML Framework for Industrial IoT

Devon Peticolas, Russell Kirmayer, Deepak S. Turaga - Oden Technologies

Outline of this talk

- Introduction to Oden and manufacturing
- System Overview
- Application for Predictive Quality Monitoring
- Future Work

Devon Peticolas

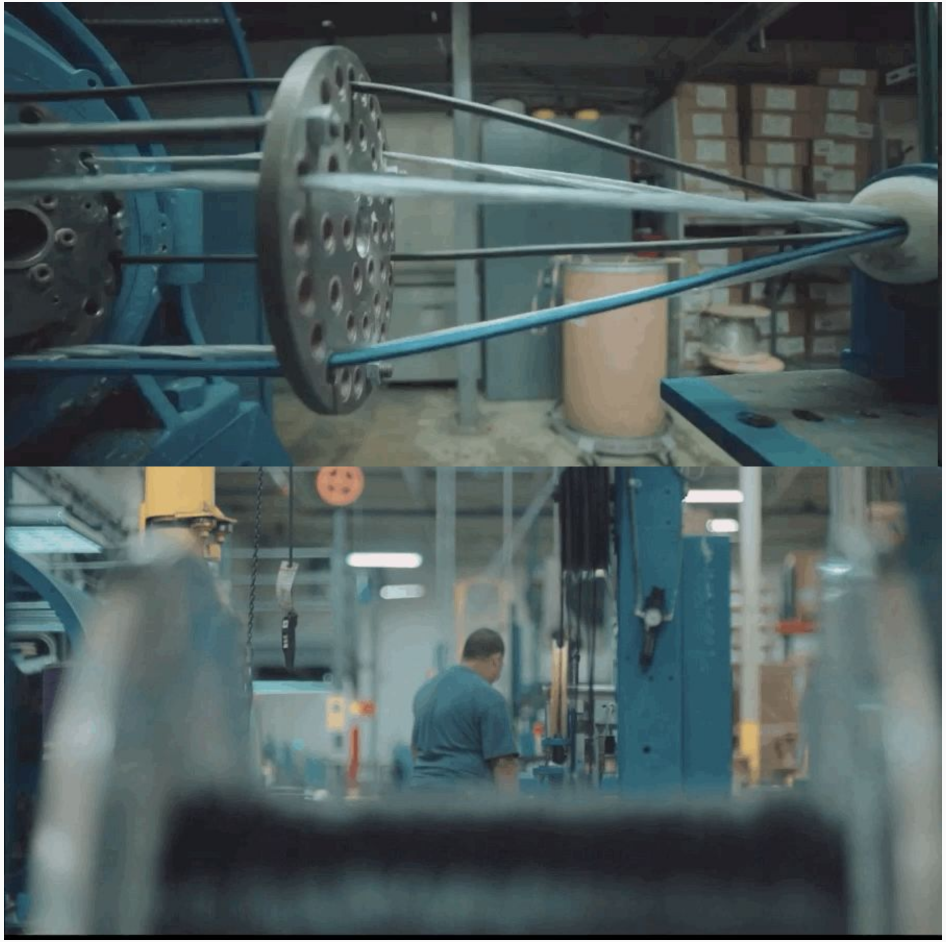
Sr. Data Engineer

Oden's Customers

Medium to large manufacturers in plastics extrusion, injection molding, and metal stamping.

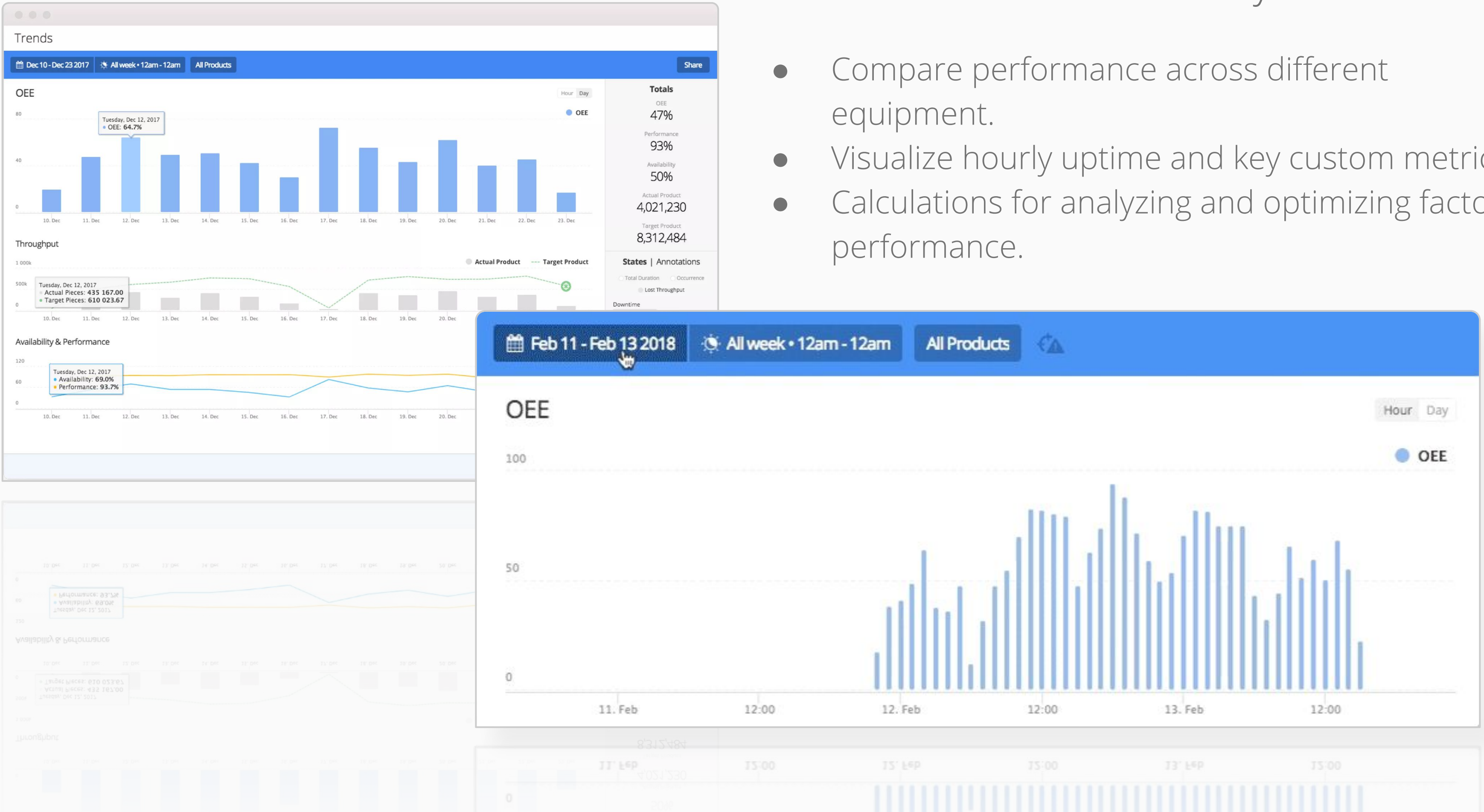
Process and Quality Engineers looking to centralize, analyze, and act on their data.

Need for real-time prediction for better quality control.



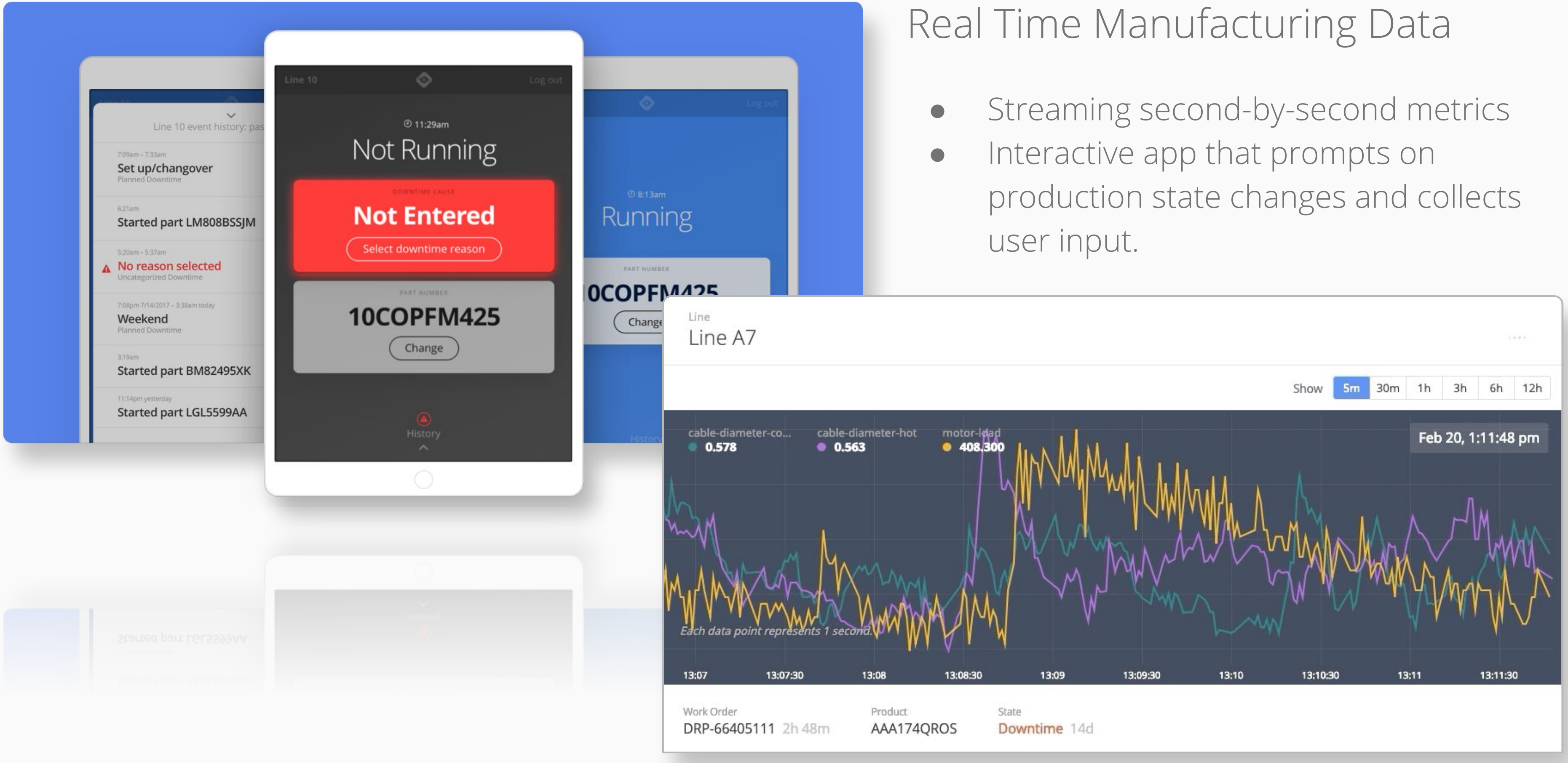
Interactive Time-series Analysis

- Compare performance across different equipment.
- Visualize hourly uptime and key custom metrics.
- Calculations for analyzing and optimizing factory performance.



Real Time Manufacturing Data

- Streaming second-by-second metrics
- Interactive app that prompts on production state changes and collects user input.



Reporting and Alerting

- Daily summaries on key process metrics from continuous intervals of production work.
- Real-time email and text alerts on target violations and concerning trends.

ALERT

Downtime violation on **Line 1**

As of 12:55pm, Line 1 has been in Downtime for more than 15 minutes.

View line

Snooze this alert for: 30m 2h 8h 24h

Powered by Oden Technologies

Is this alert useful? Let us know!

Daily Run Report

Runs completed 9:00am EST February 12, 2019 – 9:00am EST February 13, 2019

Runs sorted by worst Cpk for Cold OD Avg

SWJNG519-LQ8 View run →

Line 10 · | 0 Reels · 06:11 2/12 – 11:42 2/12 · 3h 16m uptime

METRIC	MEAN	STD DEV	TARGET	NON CON*	Cpk
Cold OD Avg	0.403	0.010	0.391 - 0.411	4.235%	**0.274**
Feet per min	274.794	194.059	-	-	-

SWHD72Y-R4 View run →

Line 10 · | 0 Reels · 10:08 2/12 – 12:34 2/13 · 1h 35m uptime

METRIC	MEAN	STD DEV	TARGET	NON CON*	Cpk
Cold OD Avg	0.141	0.002	0.135 - 0.145	0.242%	**0.782**
Feet per min	829.680	492.109	-	-	-

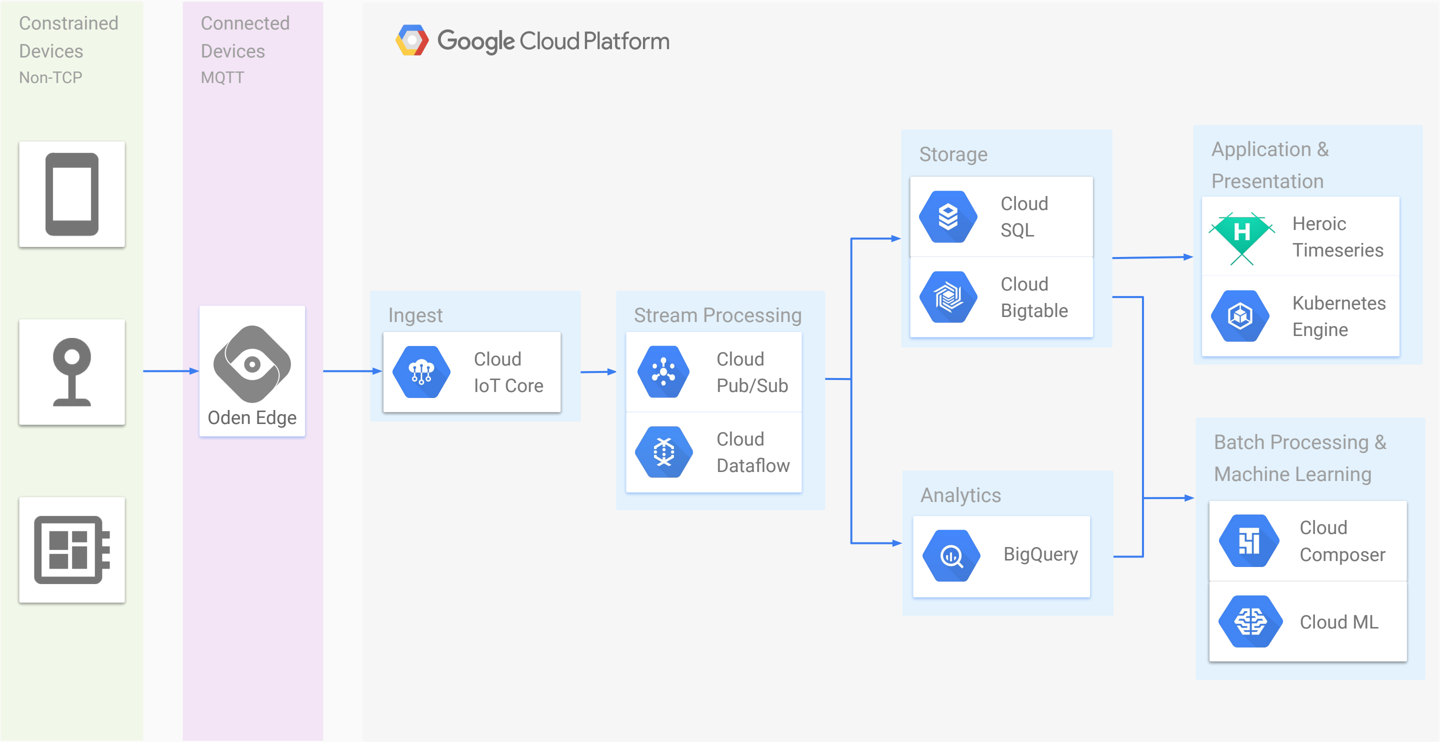
System Overview

Technology - Hardware

Oden Hardware

- Linux devices that connect via standard industrial protocols over serial and ethernet and speak of MQTT
- On-prem servers that a subset "edge" version of the Oden platform and speak to devices and modern PLCs via MQTT
- Connect to our services in the cloud via wired, wifi, or cellular networks.

Technology - Architecture



System Overview

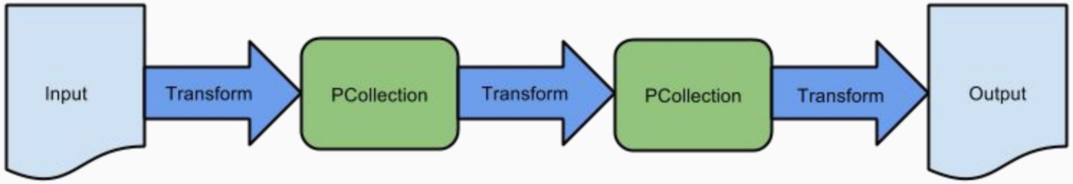
- Infrastructure for optimized data capture
- Multi-resolution storage of manufacturing data
- Workflow for ML model training and validation
- Network-partition resliant deployment of ML to cloud and factory

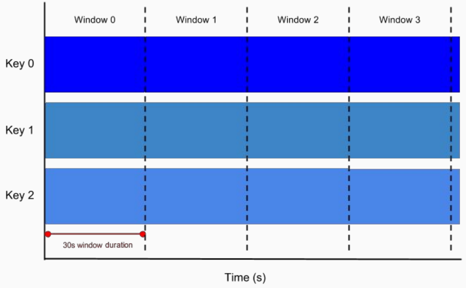
Batch and Streaming Data Handling - Apache Beam

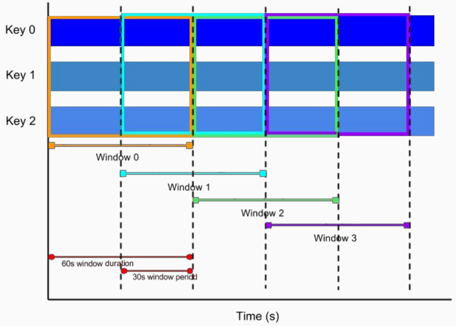
- Open Source
- Unified programming model for **stream and batch** processing
- **Portable** to multiple “runners” (Dataflow and Apache Flink)
- Java and Python SDKs

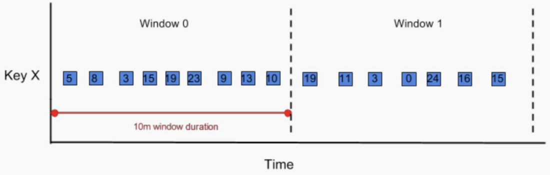
Unified Programming Model

- Unifies inputs, outputs, and intermediate state as **PCollections** (bounded or unbounded) linked by **transforms** built into a **pipeline**.
- Supports streaming joins, group-bys, stepping and sliding windows, and global state.
- Offers fine-grained tooling around handling late data.
- Can be executed over both batch and streaming data.









8.4.1.1. Accumulating mode

If our trigger is set to accumulating mode, the trigger emits the following values each time it fires. Keep in mind that the trigger fires every time three elements arrive:

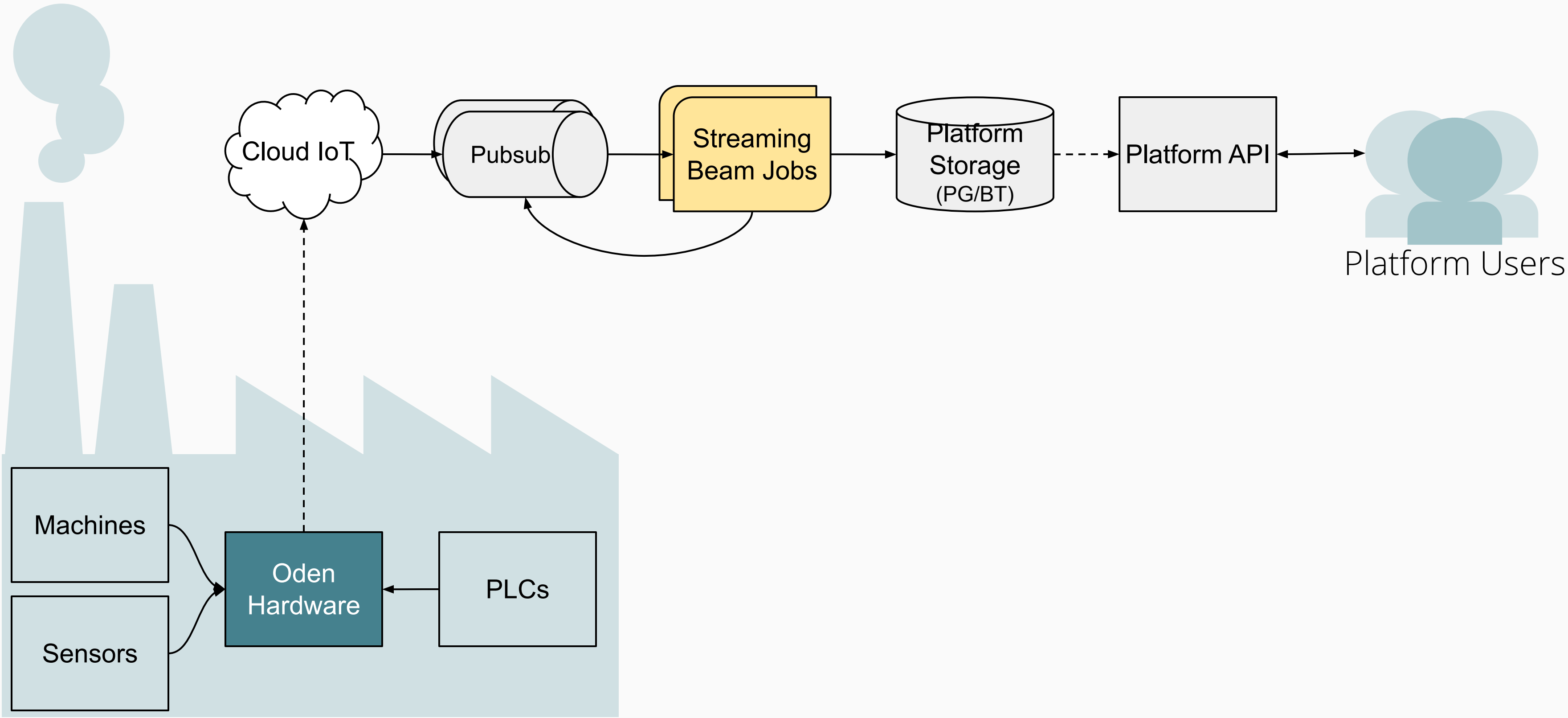
```
First trigger firing:  [5, 8, 3]
Second trigger firing: [5, 8, 3, 15, 19, 23]
Third trigger firing:  [5, 8, 3, 15, 19, 23, 9, 13, 10]
```

8.4.1.2. Discarding mode

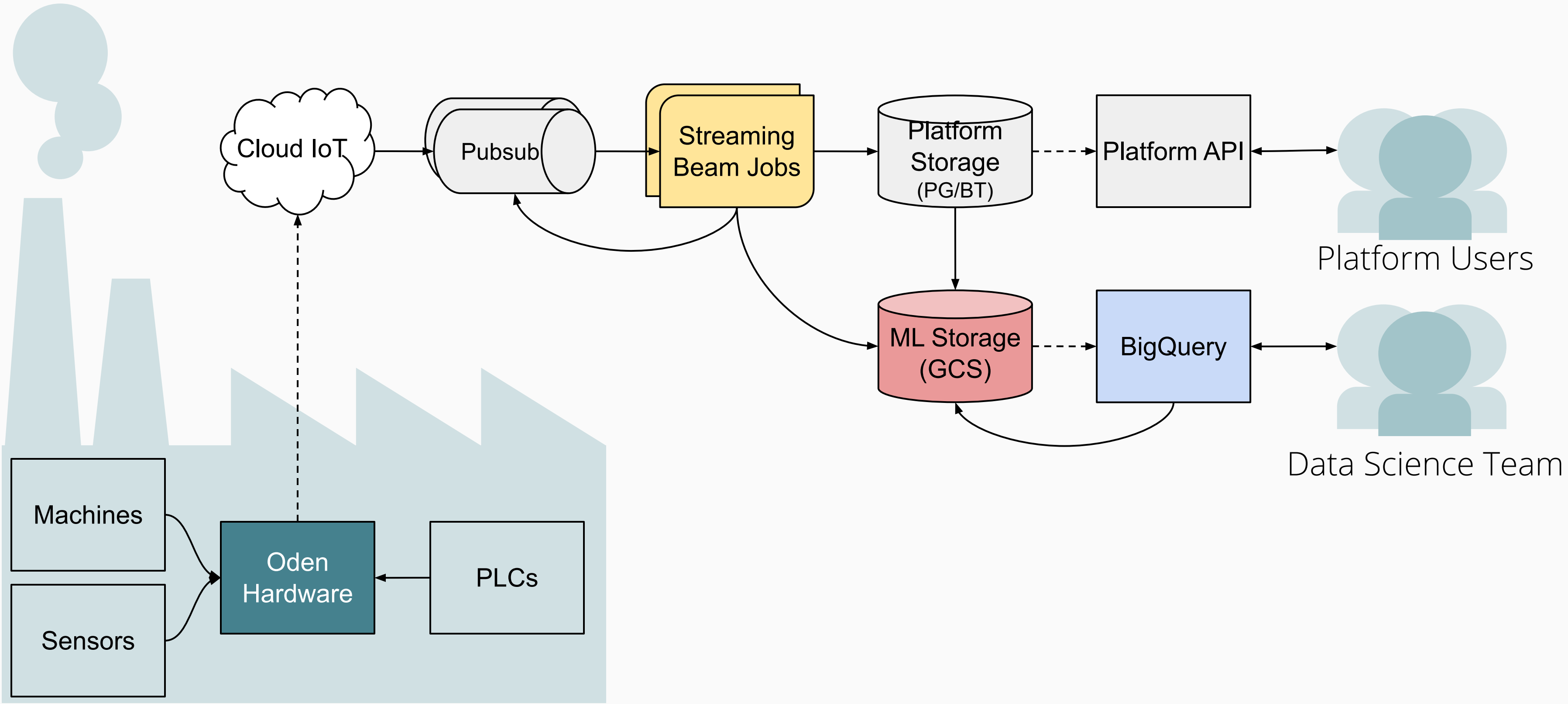
If our trigger is set to discarding mode, the trigger emits the following values on each firing:

```
First trigger firing:  [5, 8, 3]
Second trigger firing:           [15, 19, 23]
Third trigger firing:                         [9, 13, 10]
```

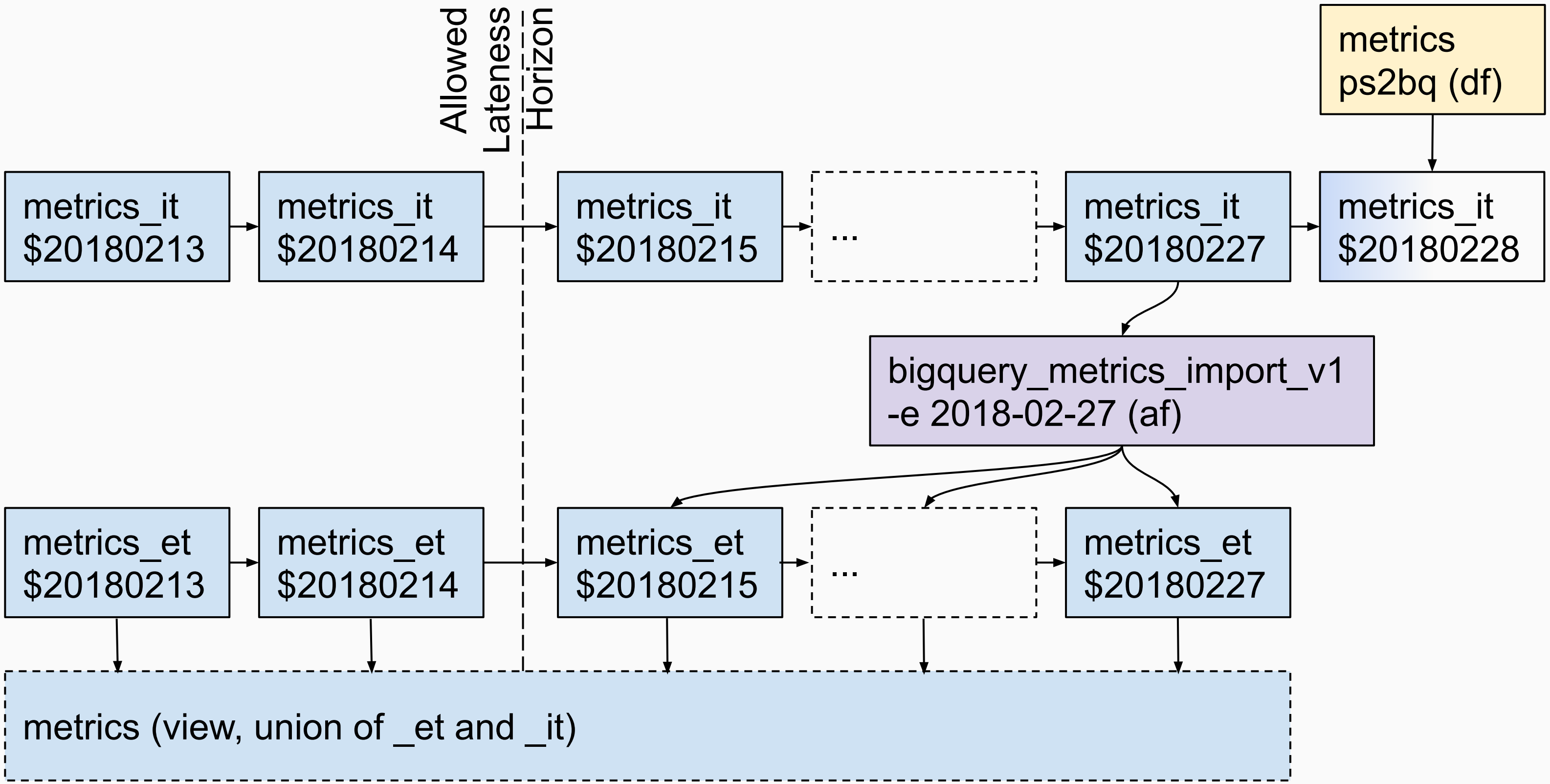
Oden's Streaming Data Pipeline



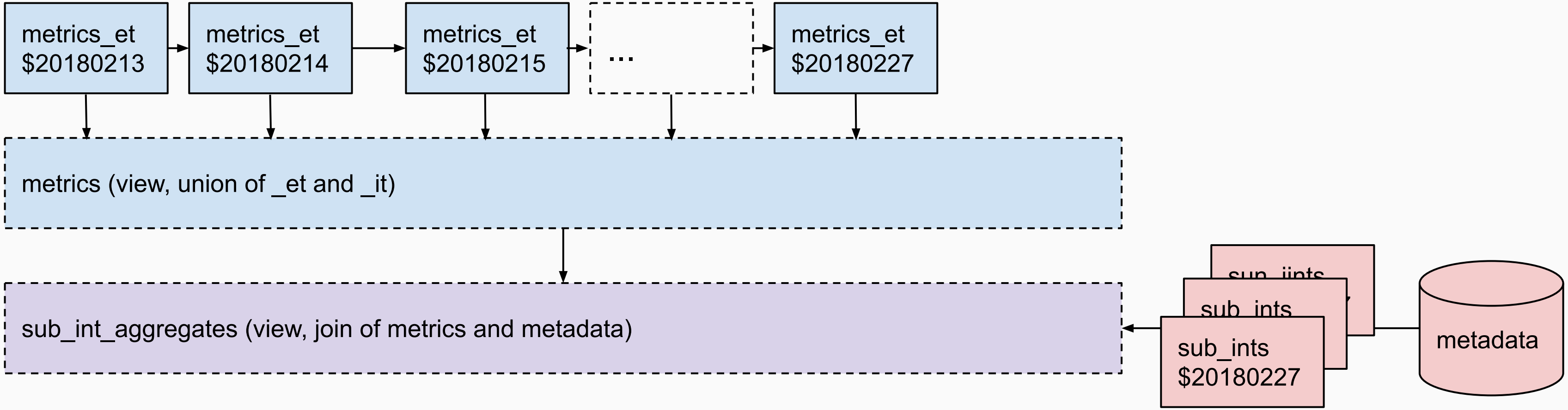
Data Storage Optimization



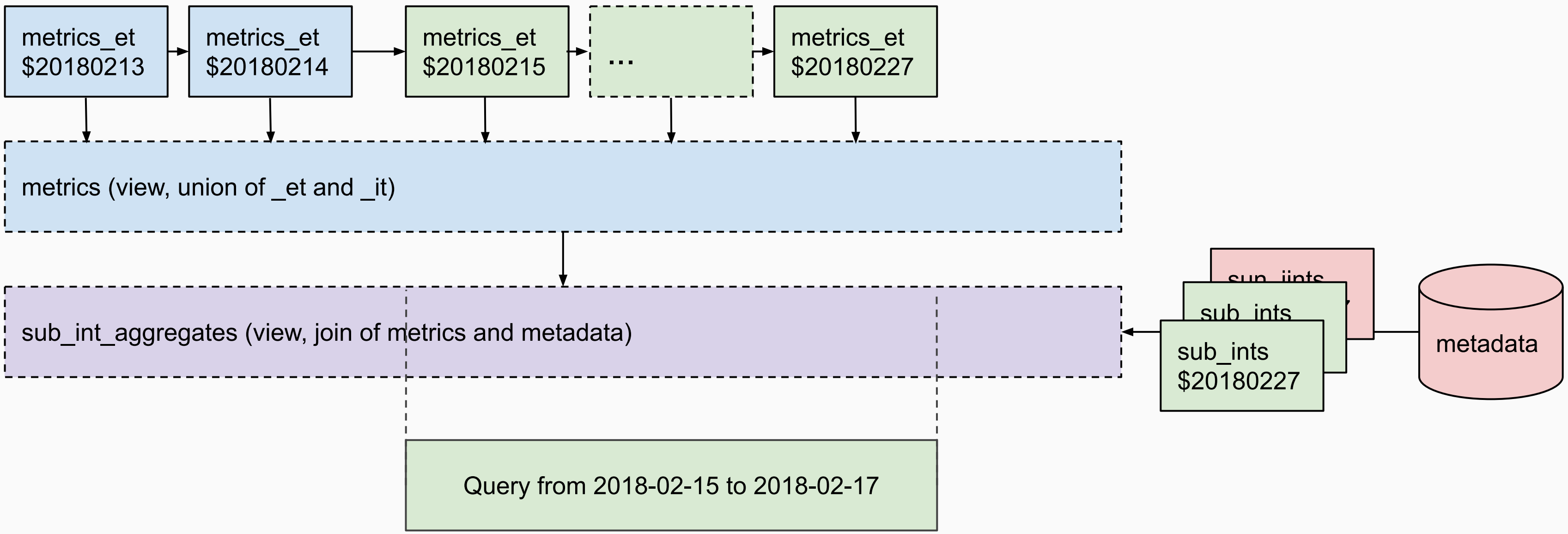
Data Storage Optimization



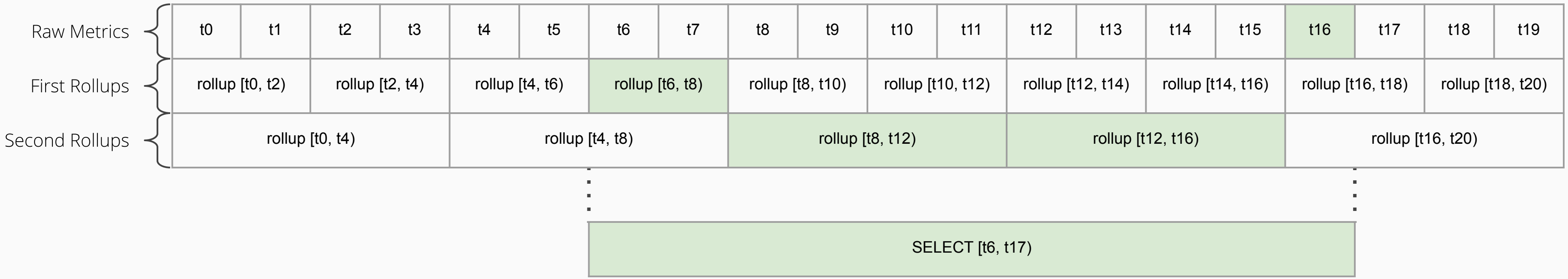
Data Storage Optimization



Data Storage Optimization



Multi-resolution Rollup Aggregates



Rollup non-overlapping windows of metrics using *associative aggregates.*

- Count
- Sum
- Min, Max
- Sum2 - sum of x squared

$(x * y) * z = x * (y * z)$ *for all* x, y, z *in* S

$f(A \cup B) = g(f(A), f(B))$

$\mathrm{sum}(A \cup B) = \mathrm{sum}(A) + \mathrm{sum}(B)$

$\mathrm{count}(A \cup B) = \mathrm{count}(A) + \mathrm{count}(B)$

$\max(A \cup B) = \max(\max(A), \max(B))$

$\mathrm{sum2}(A \cup B) = \mathrm{sum2}(A \cup B) + \mathrm{sum2}(A \cup B)$

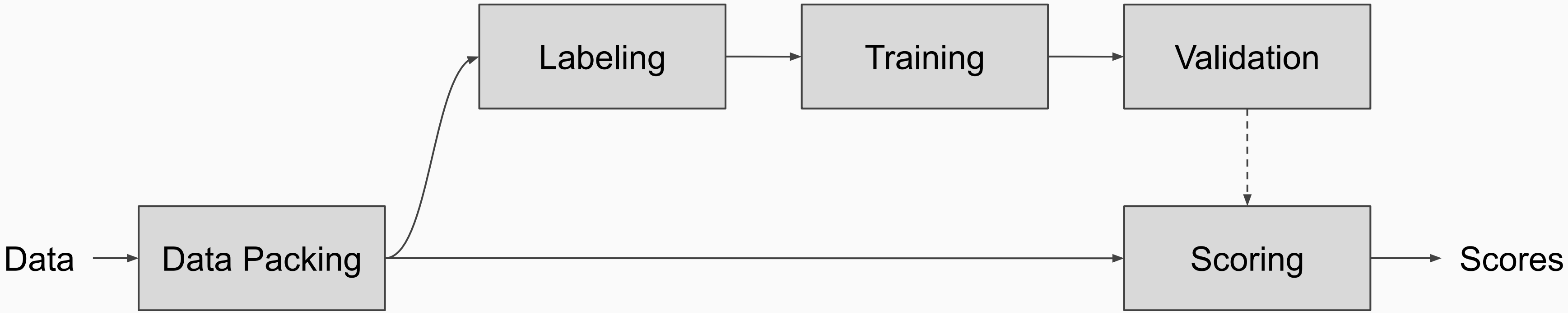
$mean(A \cup B) = sum(A \cup B) / count(A \cup B)$

$stddev(A \cup B) = 1/(count(A \cup B)*(count(A \cup B)-1))*(sum2(A \cup B)-sum(A \cup B)\text{^}2)$

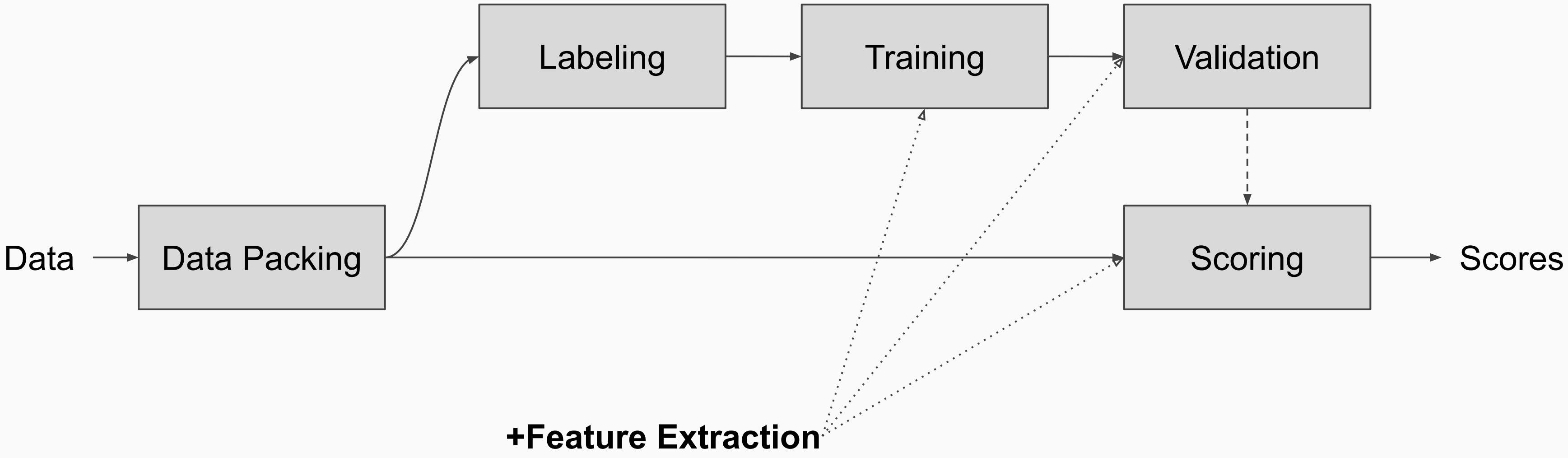
Composable ML Workflows

- **Data Packing** - Collecting all metrics and metadata needed to compose the feature set for one training or scoring example.
- **Labeling** - Assigning a label to each data pack.
- **Feature Extraction** - Extracting features from the data packs for training or scoring.
- **Model Training** - Building a model against training data, the features + labels
- **Model Validation** - Validating the model against held out data
- **Model Scoring** - Applying the model to new data

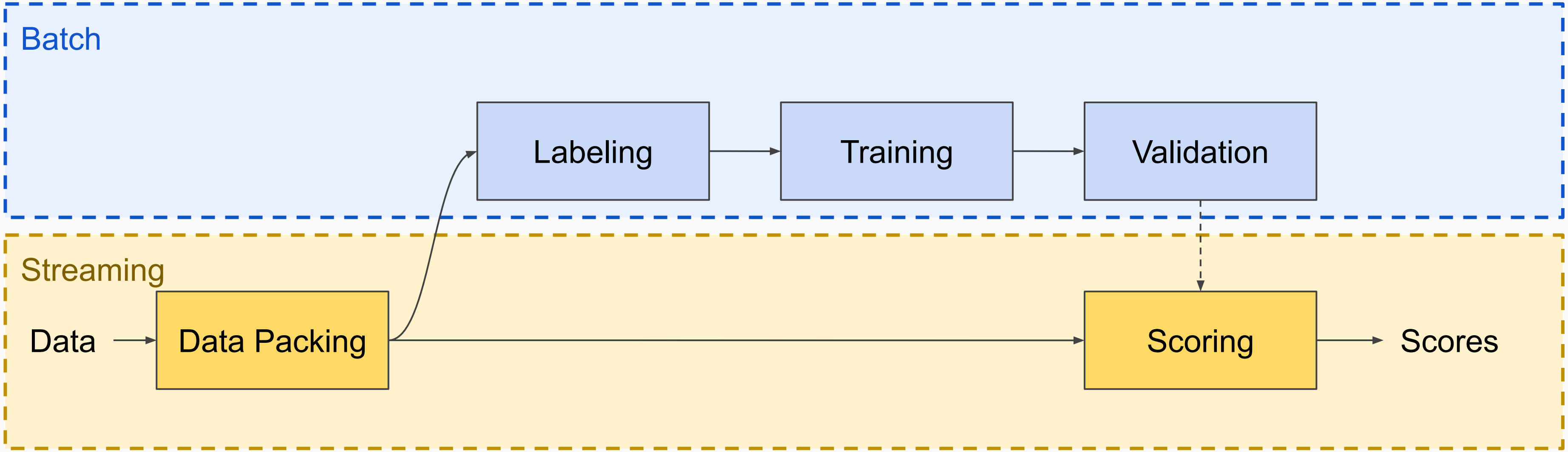
Composable ML Workflows



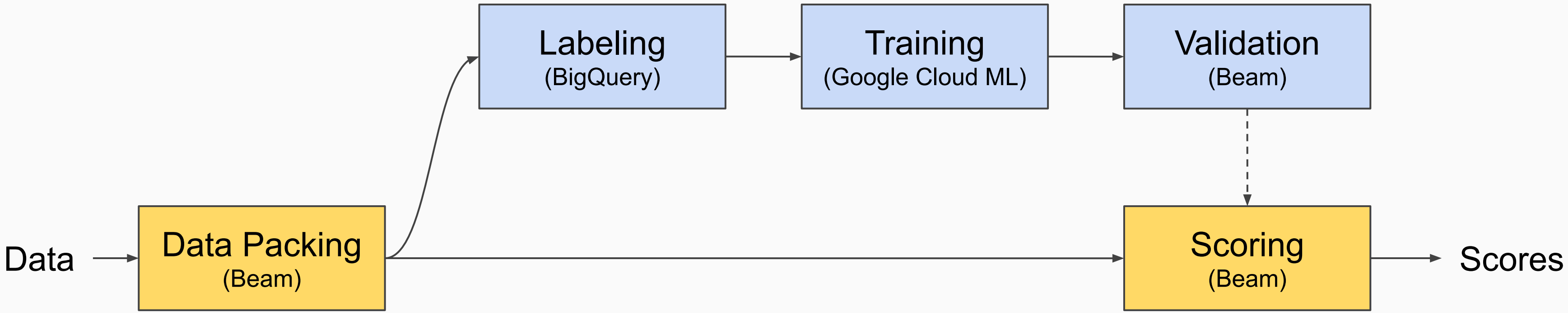
Composable ML Workflows



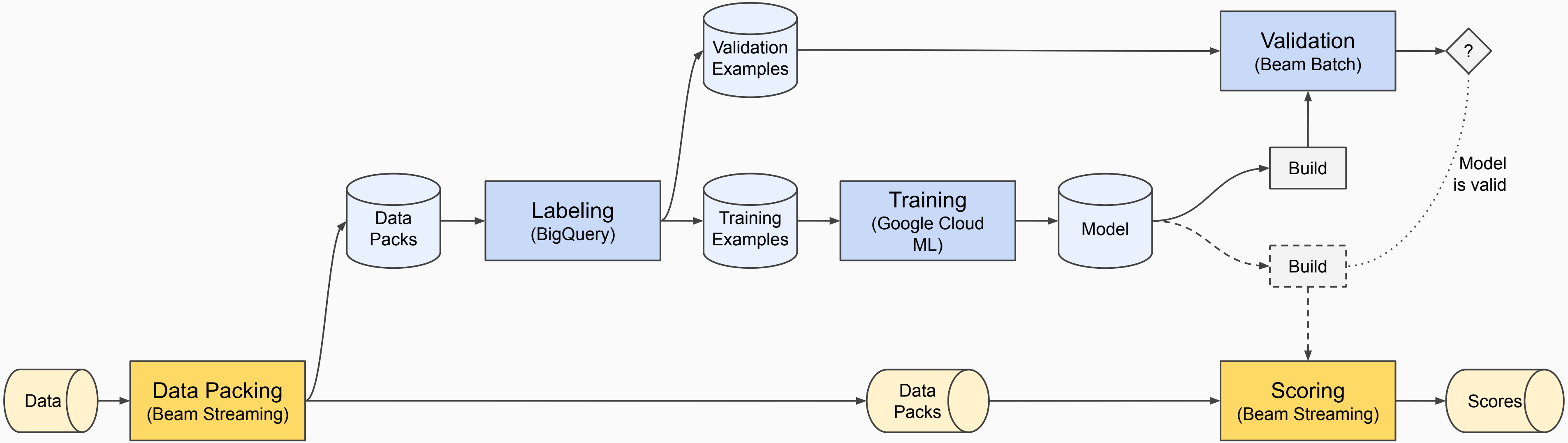
Composable ML Workflows



Composable ML Workflows



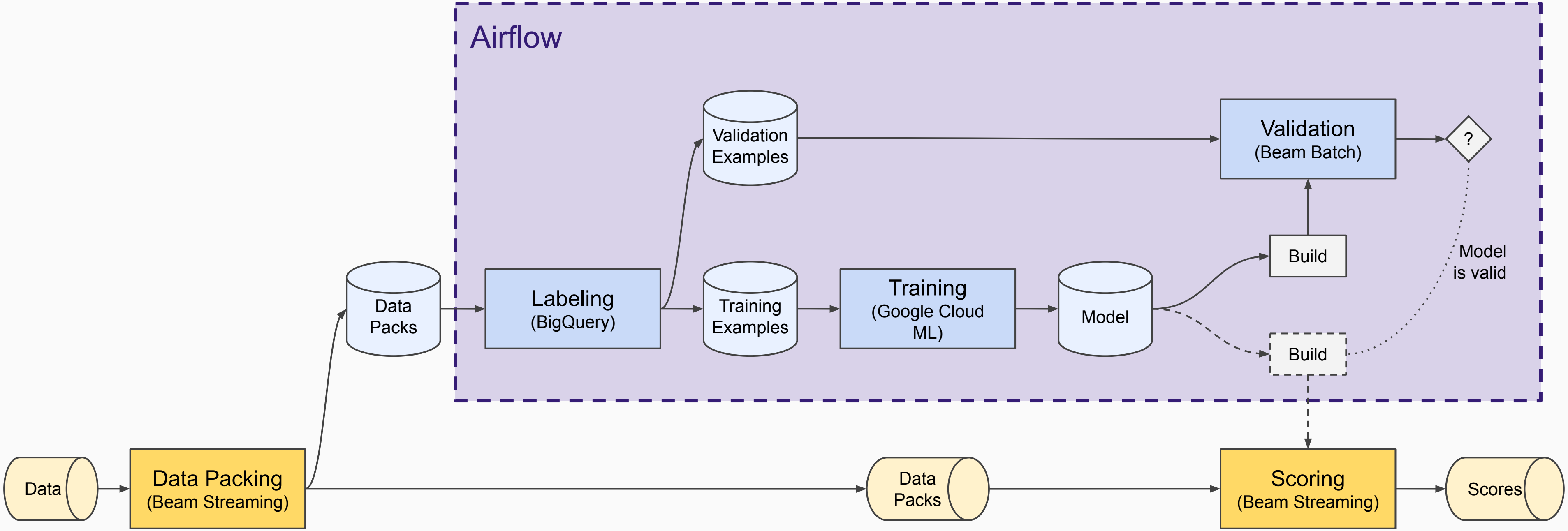
Composable ML Workflows



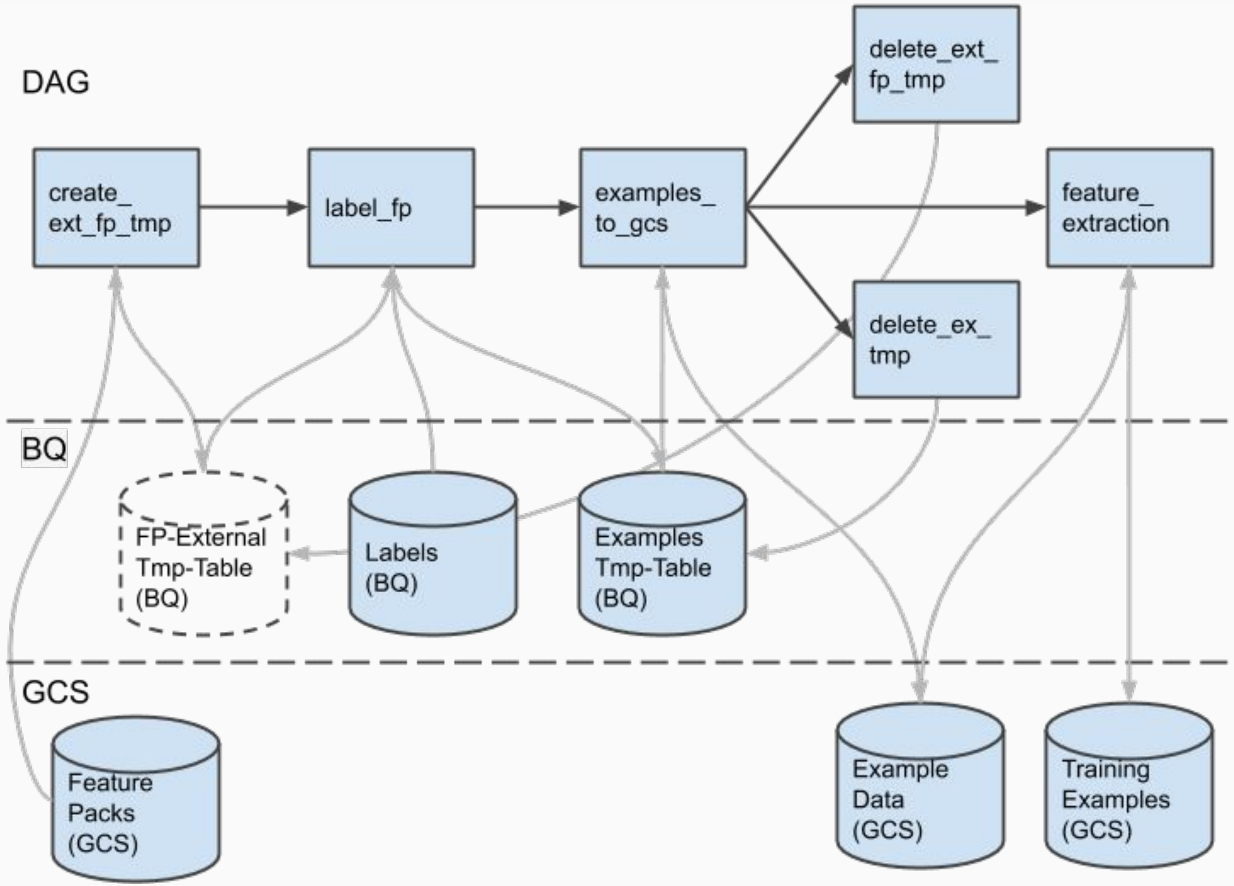
Workflow Orchestration

- Labeling, Training, and validation are orchestrated using Apache Beam in DAGs
- Each step (or task) within the DAG is remotely managed process
- Input and output of each step (or task) is a table or set of files

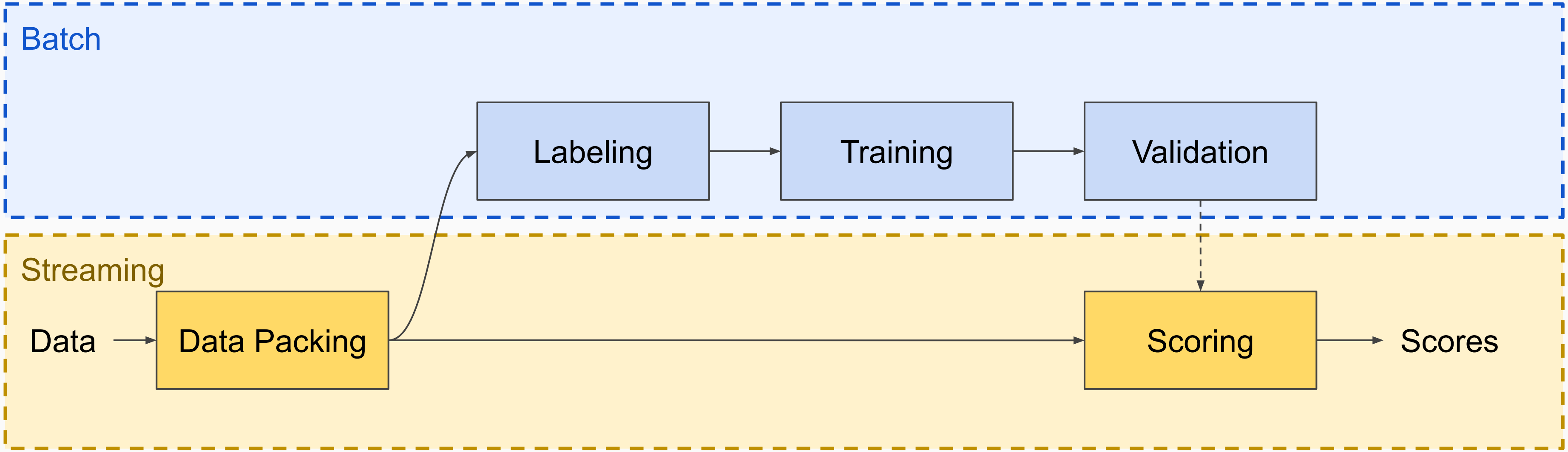
Composable ML Workflows



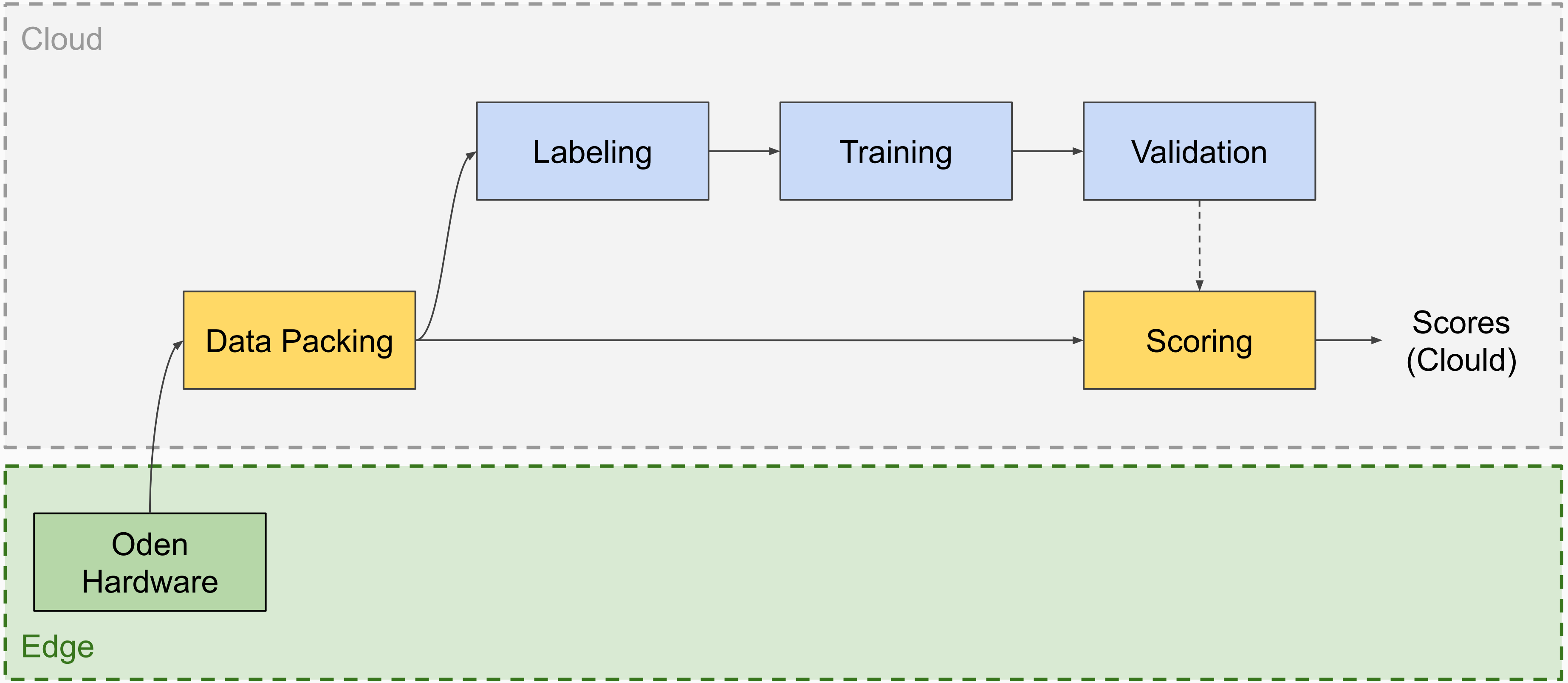
Workflow Orchestration



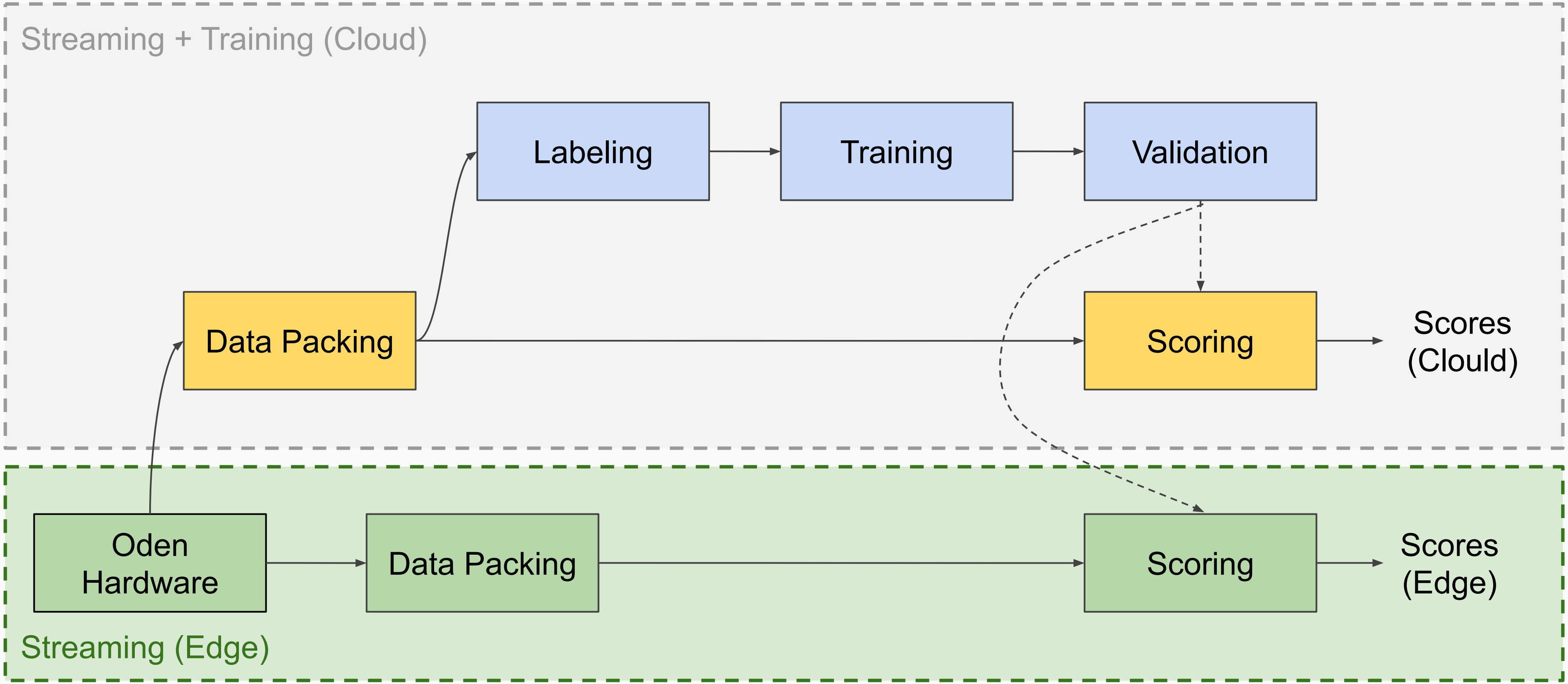
Composable ML Workflows



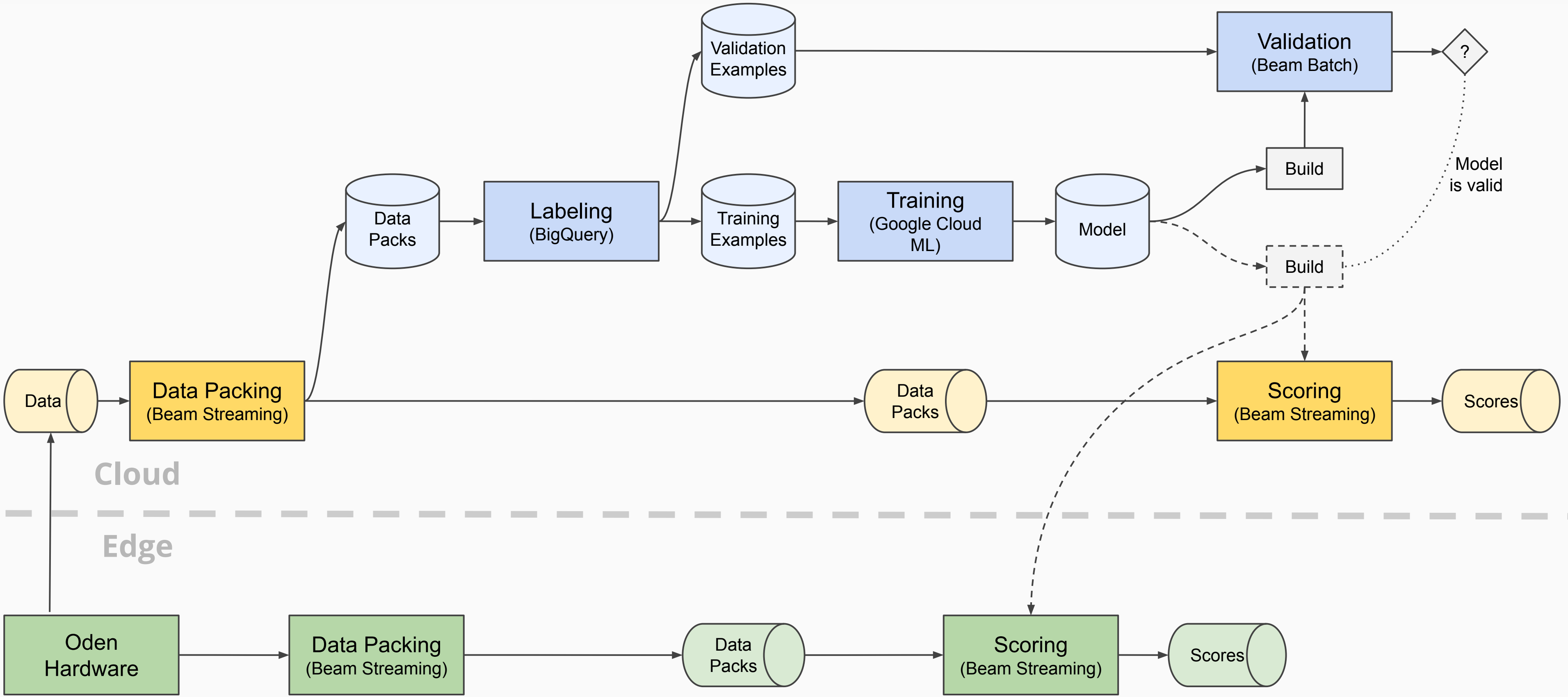
Model Scoring in Cloud and Edge



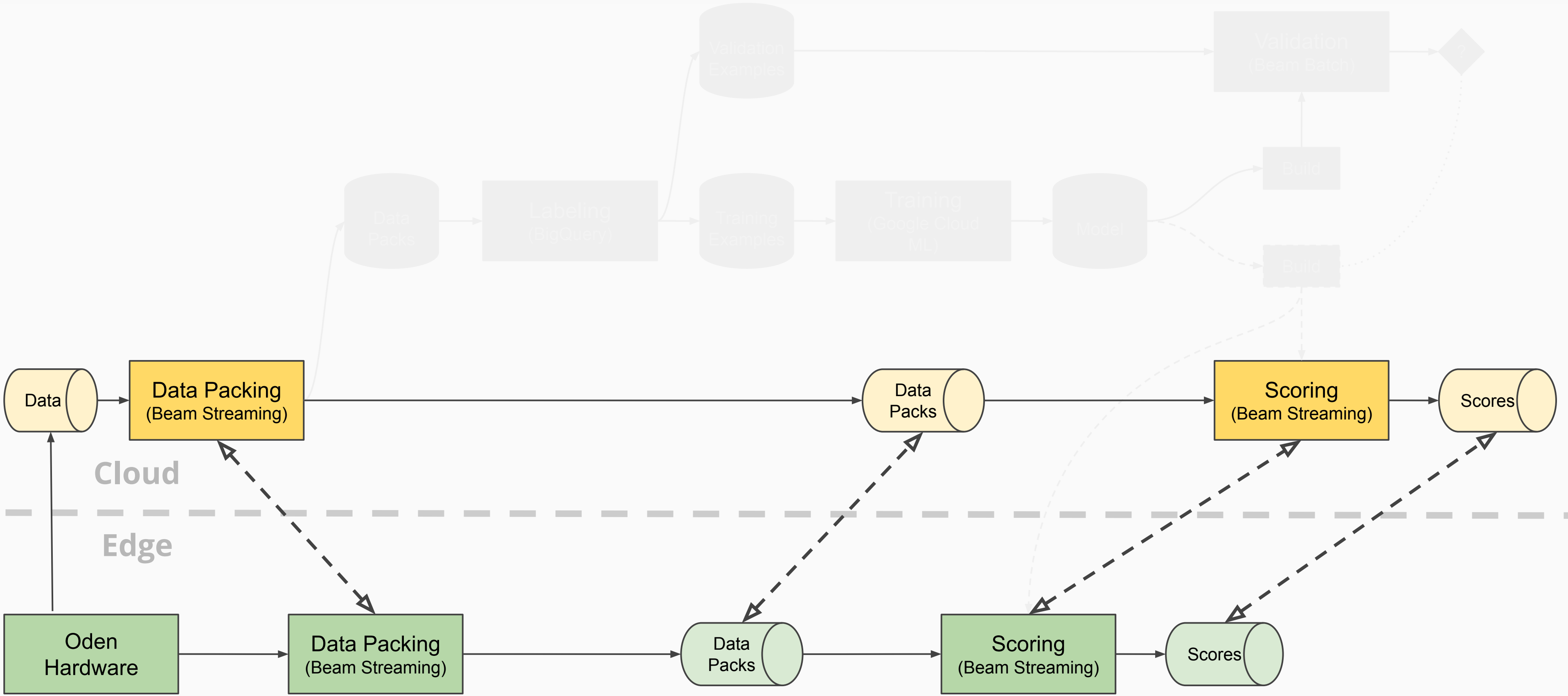
Model Scoring in Cloud and Edge



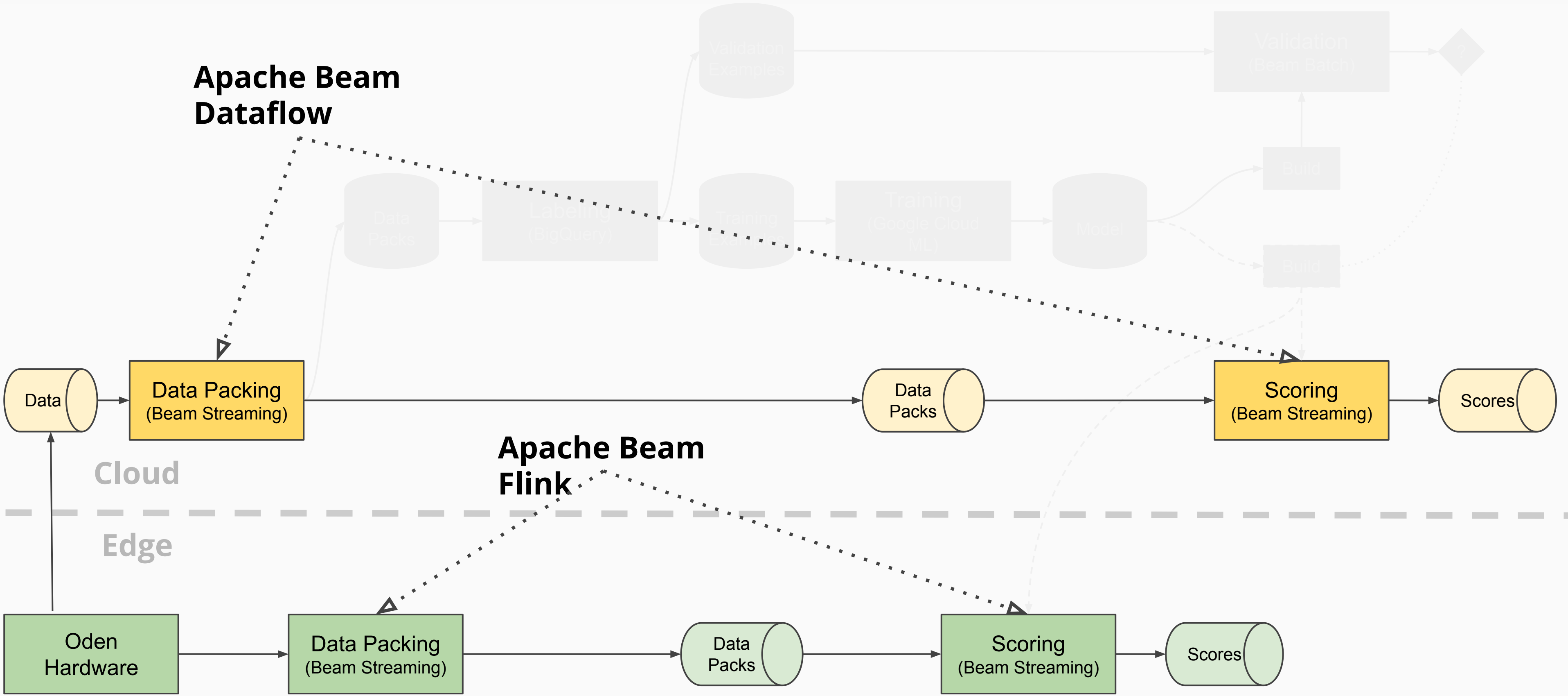
Composable ML Workflows



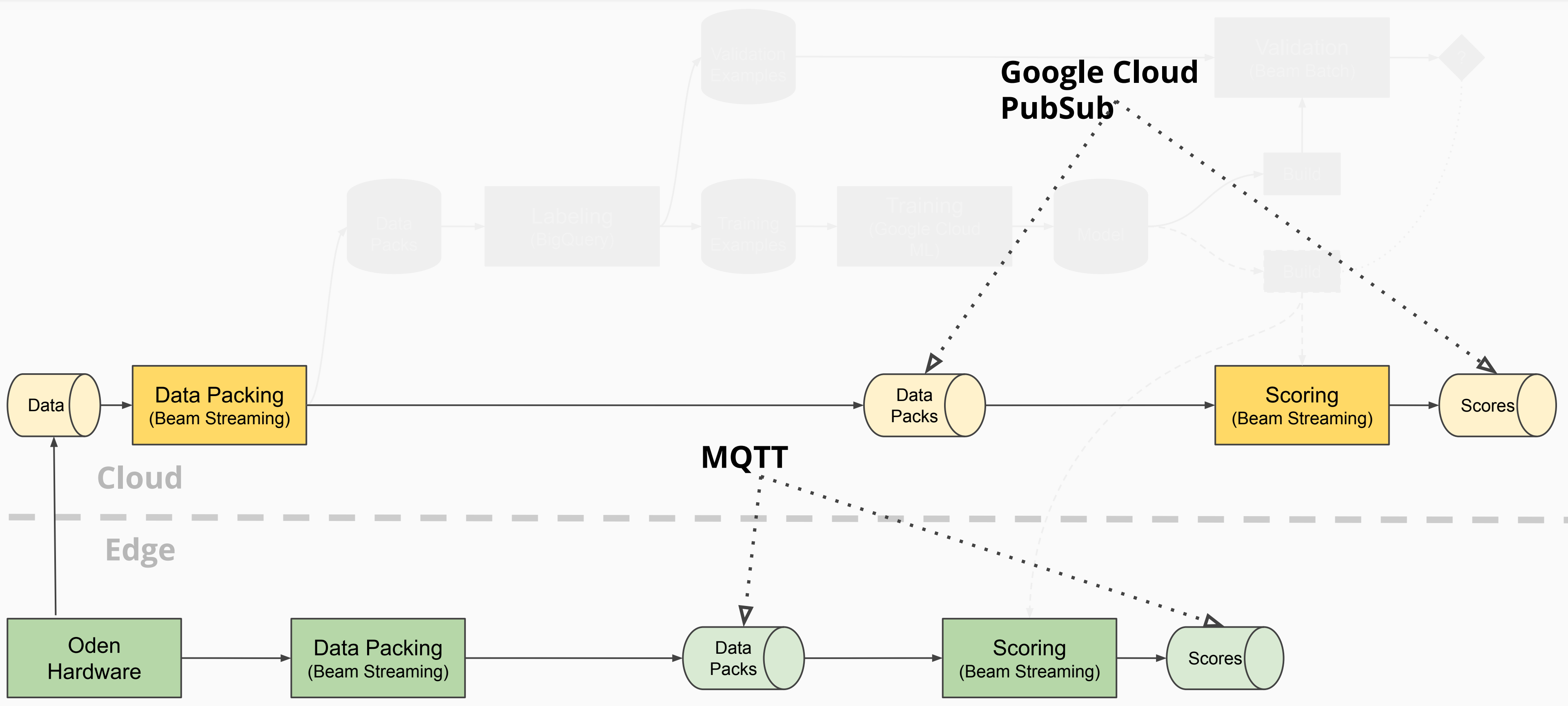
Composable ML Workflows



Composable ML Workflows



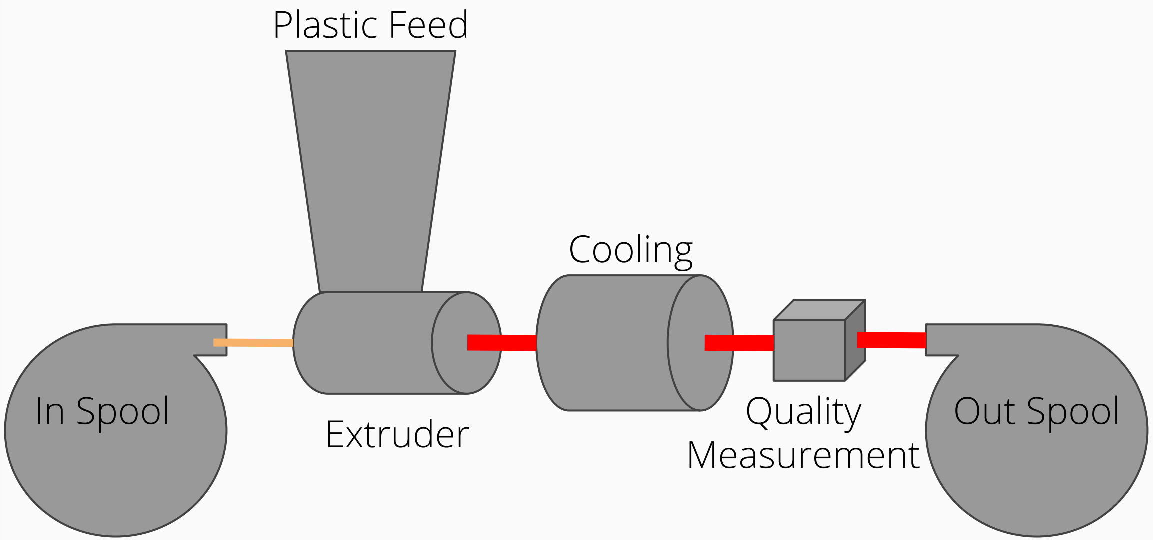
Composable ML Workflows



Application: Predictive Quality

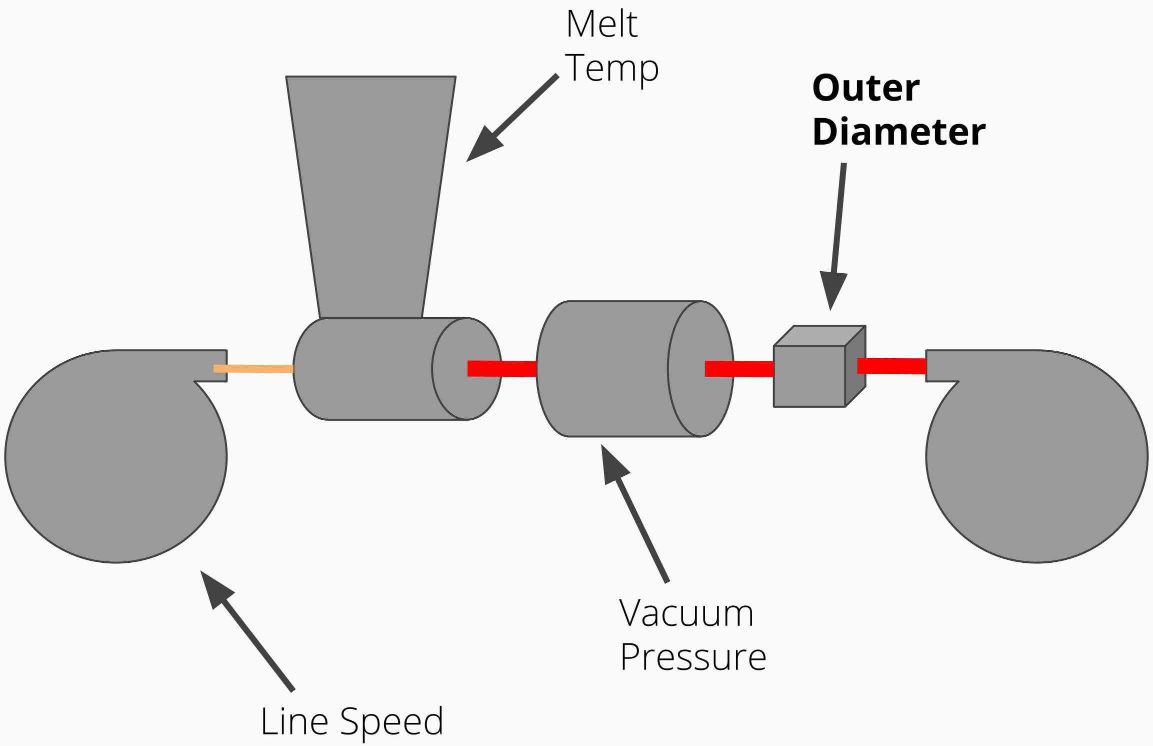
Cable Manufacturing

- Copper is pulled from an in-spool into an extruder.
- Plastic is melted over the copper to make wire.
- Wire is cooled.
- Wire is pulled into an out-spool.
- A laser measures the diameter of the wire to monitor its closeness to spec.



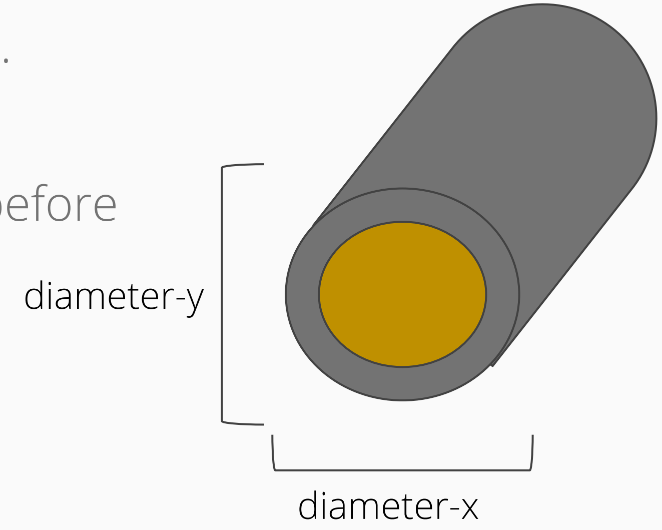
Cable Manufacturing

- Copper is pulled from an in-spool into an extruder.
- Plastic is melted over the copper to make wire.
- Wire is cooled.
- Wire is pulled into an out-spool.
- A laser measures the diameter of the wire to monitor its closeness to spec.



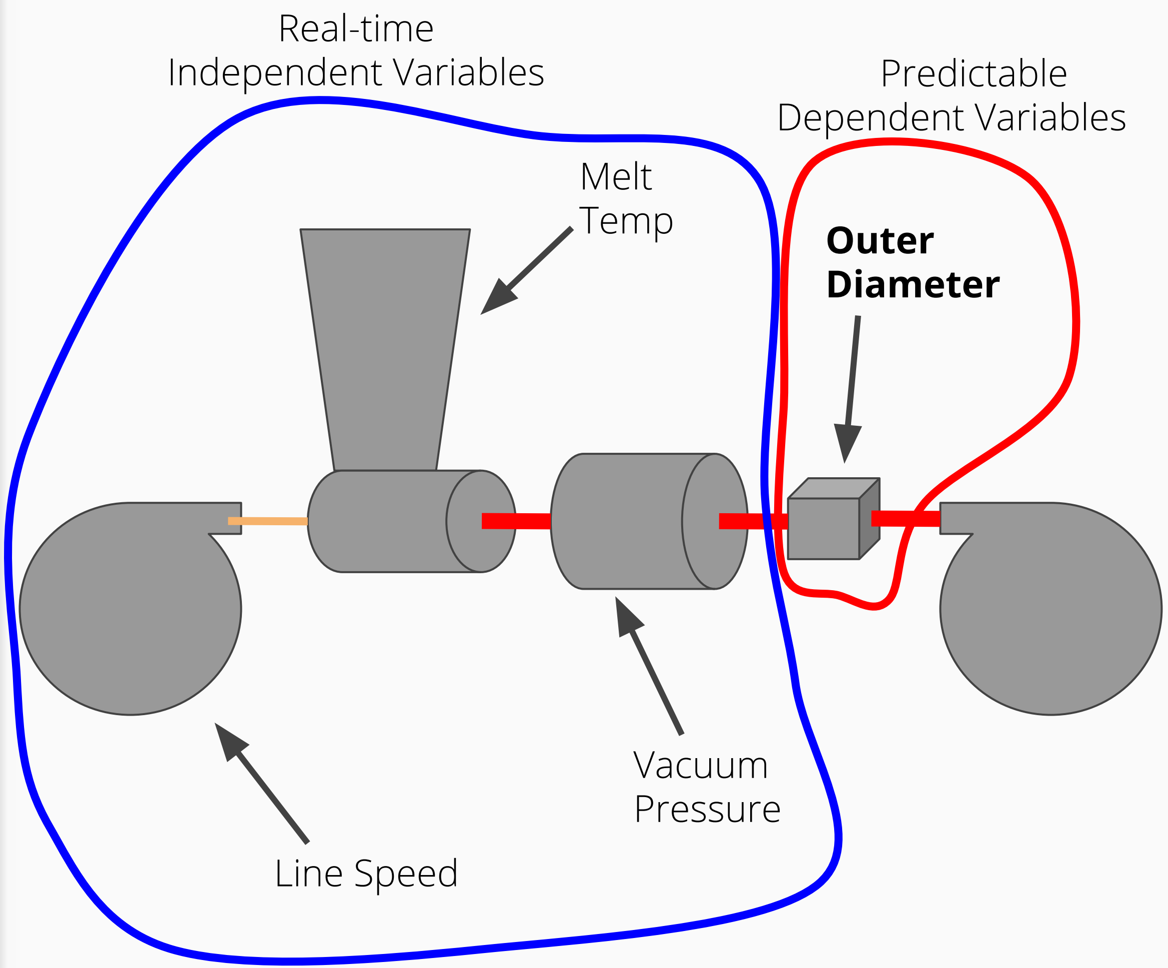
Predictive Quality

- Quality of cable coating is determined by measuring the *average diameter* and ensure it lies between an upper and lower limit [U, L].
- Information lag from measurement system means hundreds of feet can be produced before identifying an issue.
- Predictive alerting of quality issues must be tollerant to network partitons from cloud.

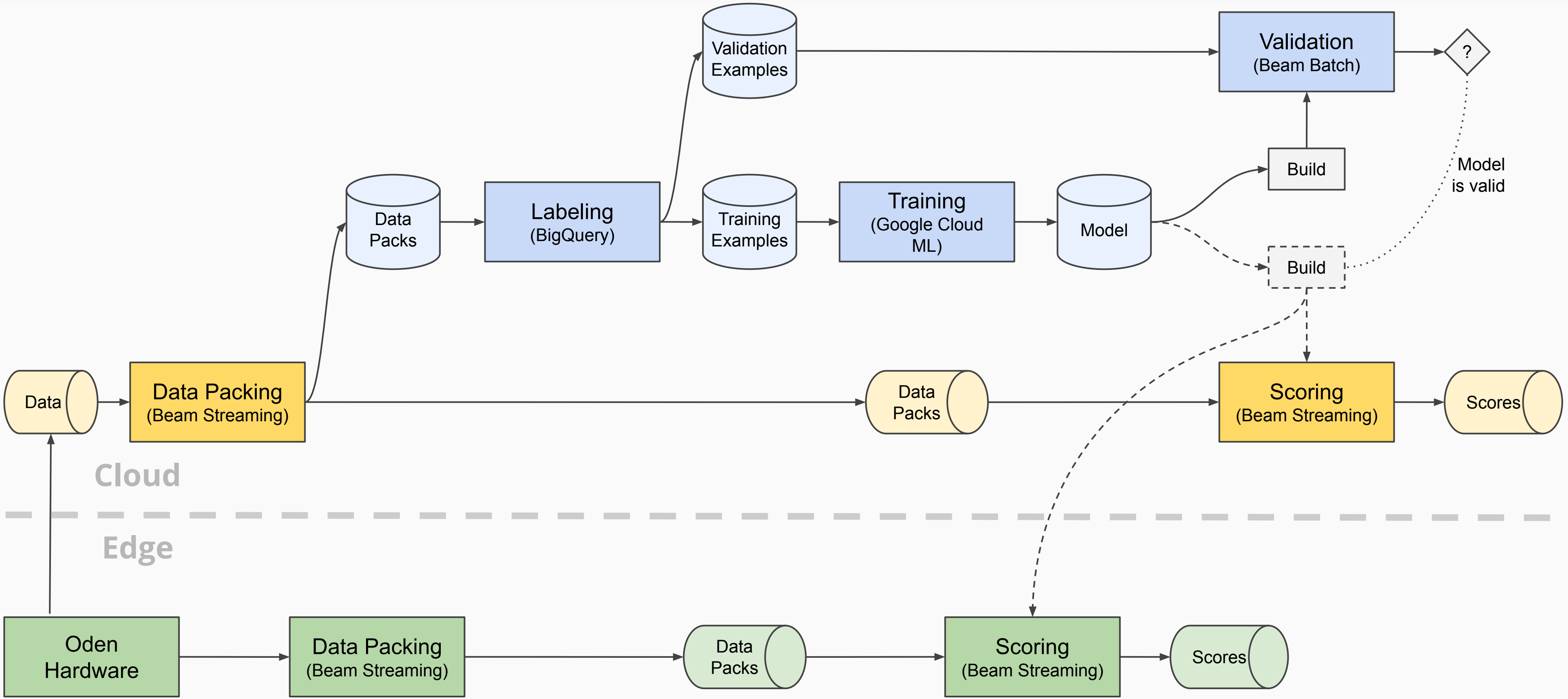


Prediction

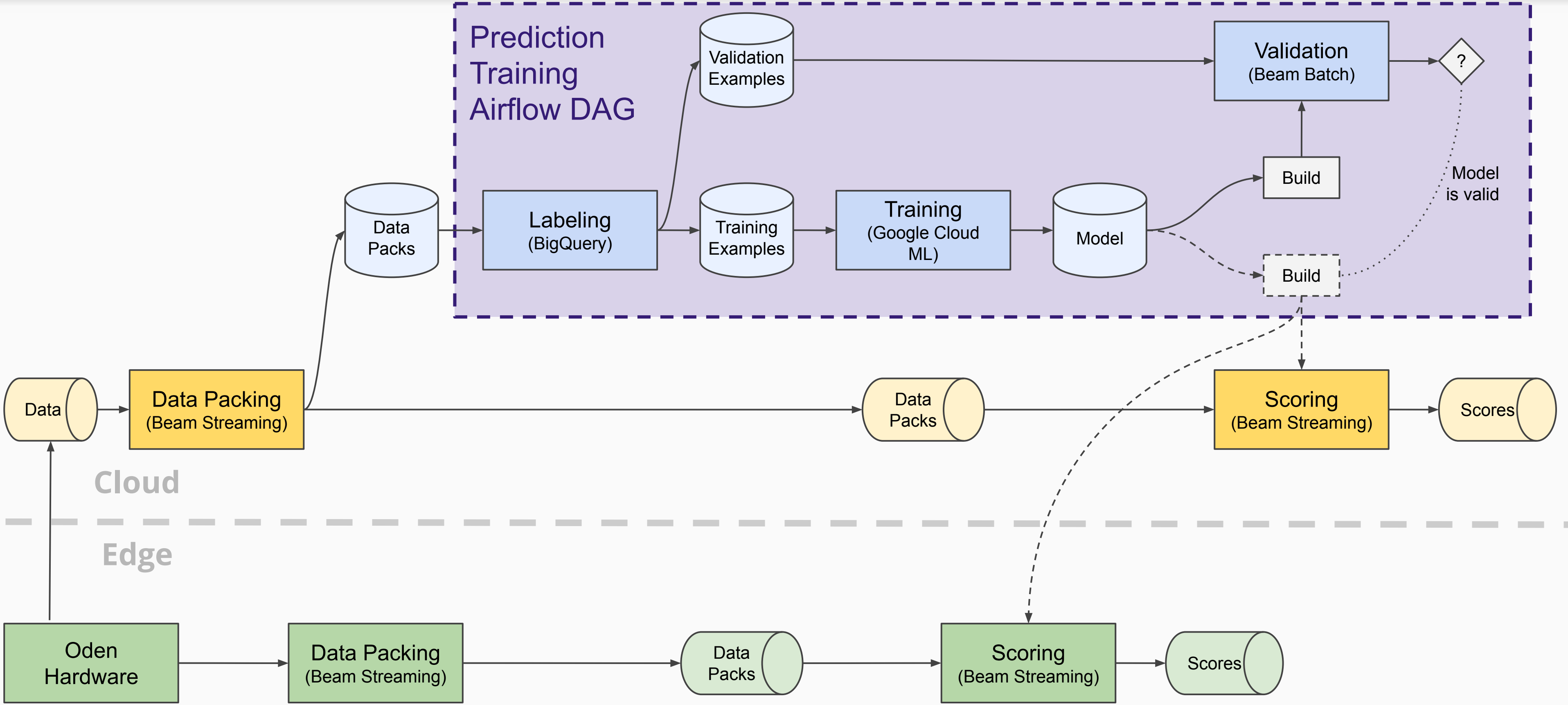
- In real-time we stream controllable metrics such as line-speed, screw-rpm, temp-die, temp-flange and pressure
- In a lagging window we collect previous values for cold-od and pre-cooled diameter hot-od.
- **We predict cold-od 5 minutes into the future and use that to estimate quality in real time.**



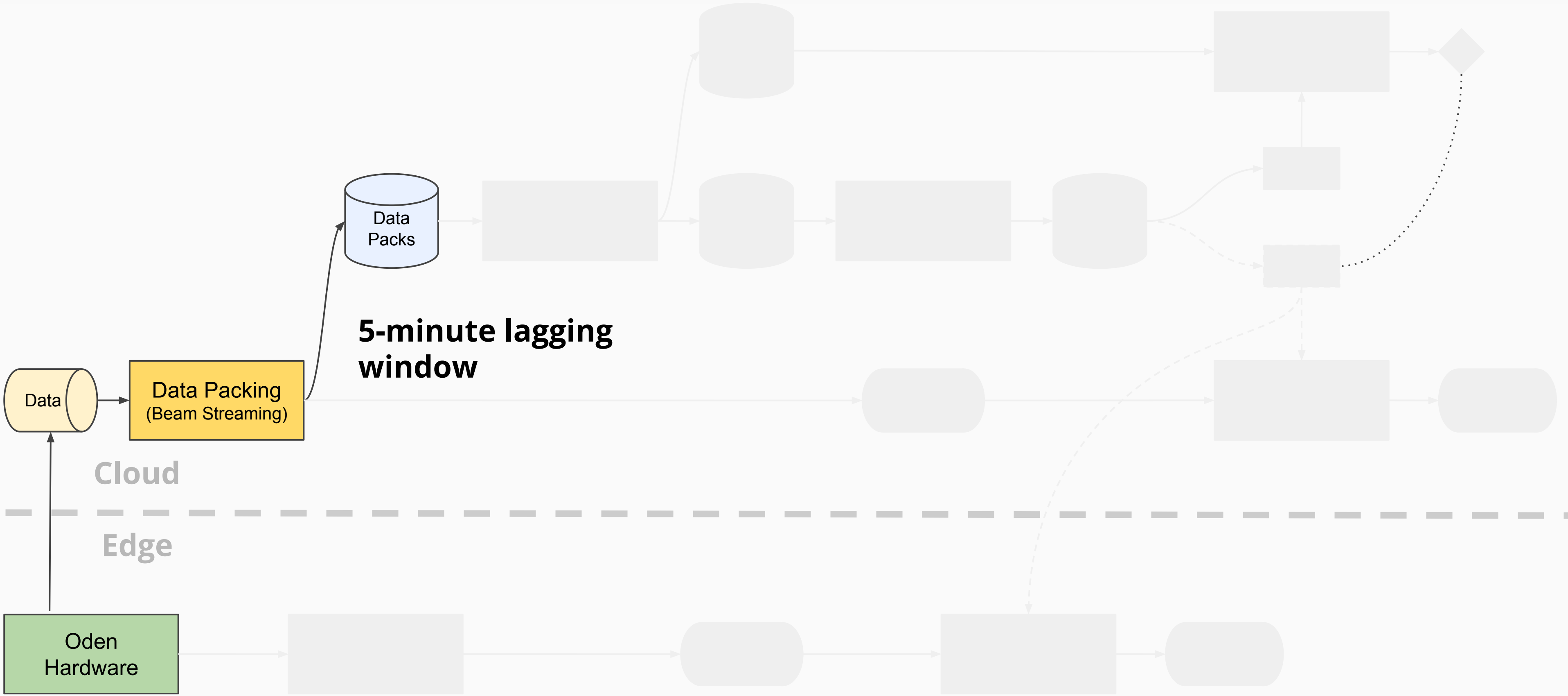
Application: Predictive Quality



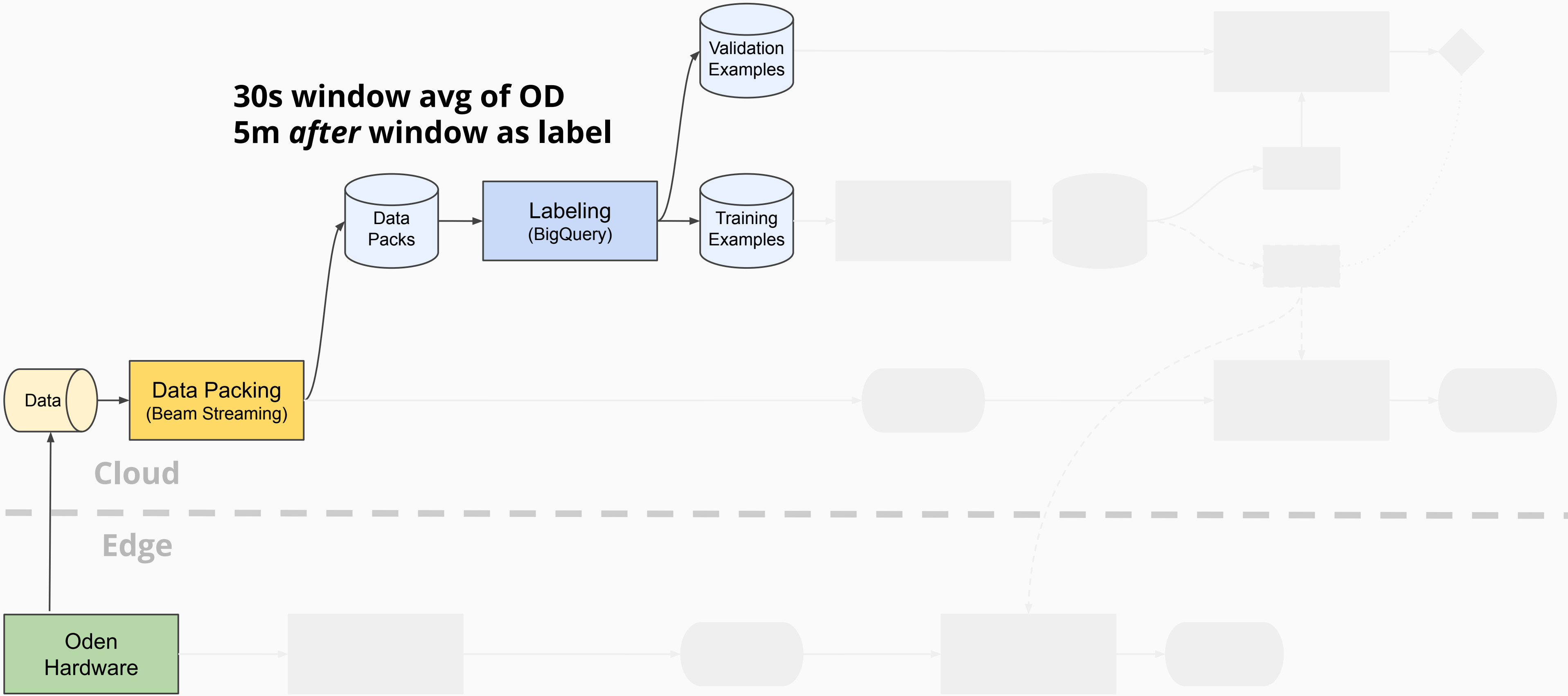
Application: Predictive Quality



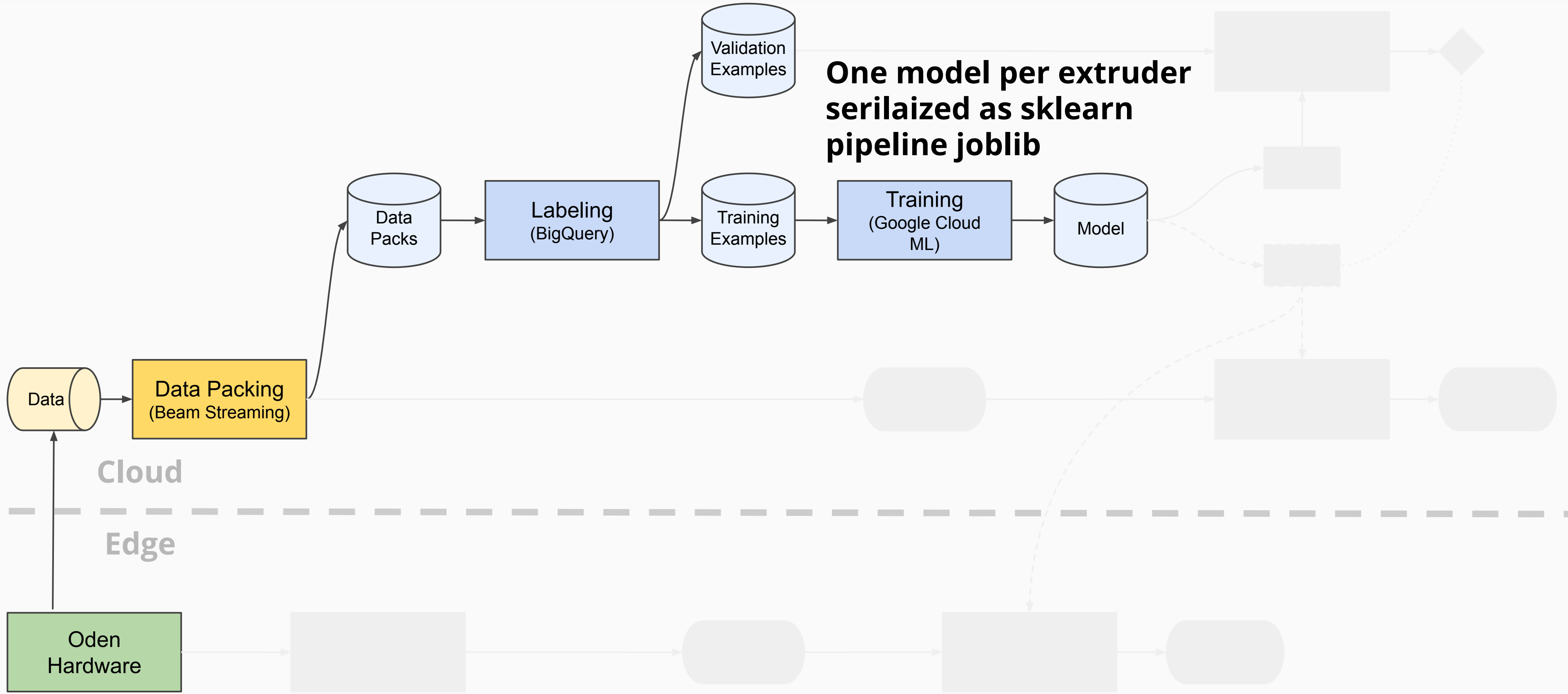
Application: Predictive Quality - Data Packing (For Training)



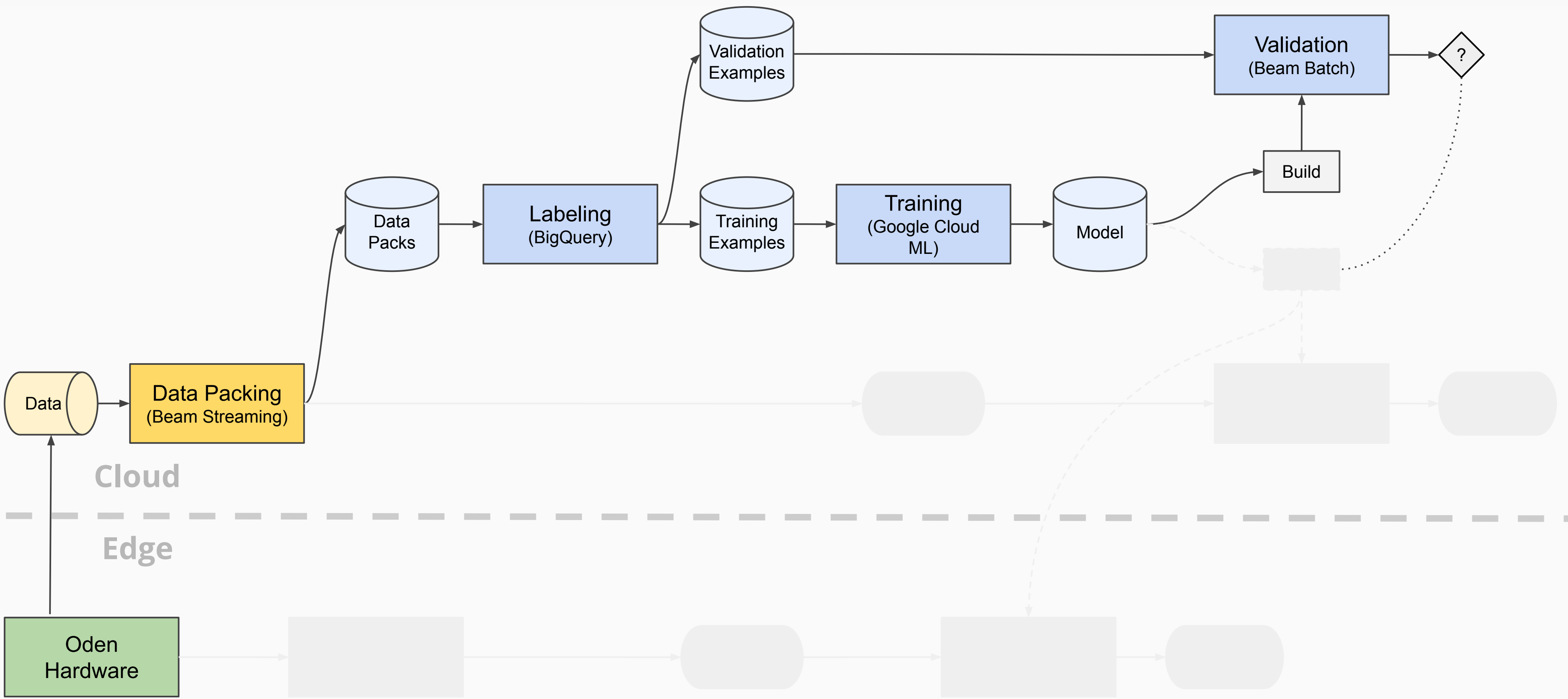
Application: Predictive Quality - Labeling



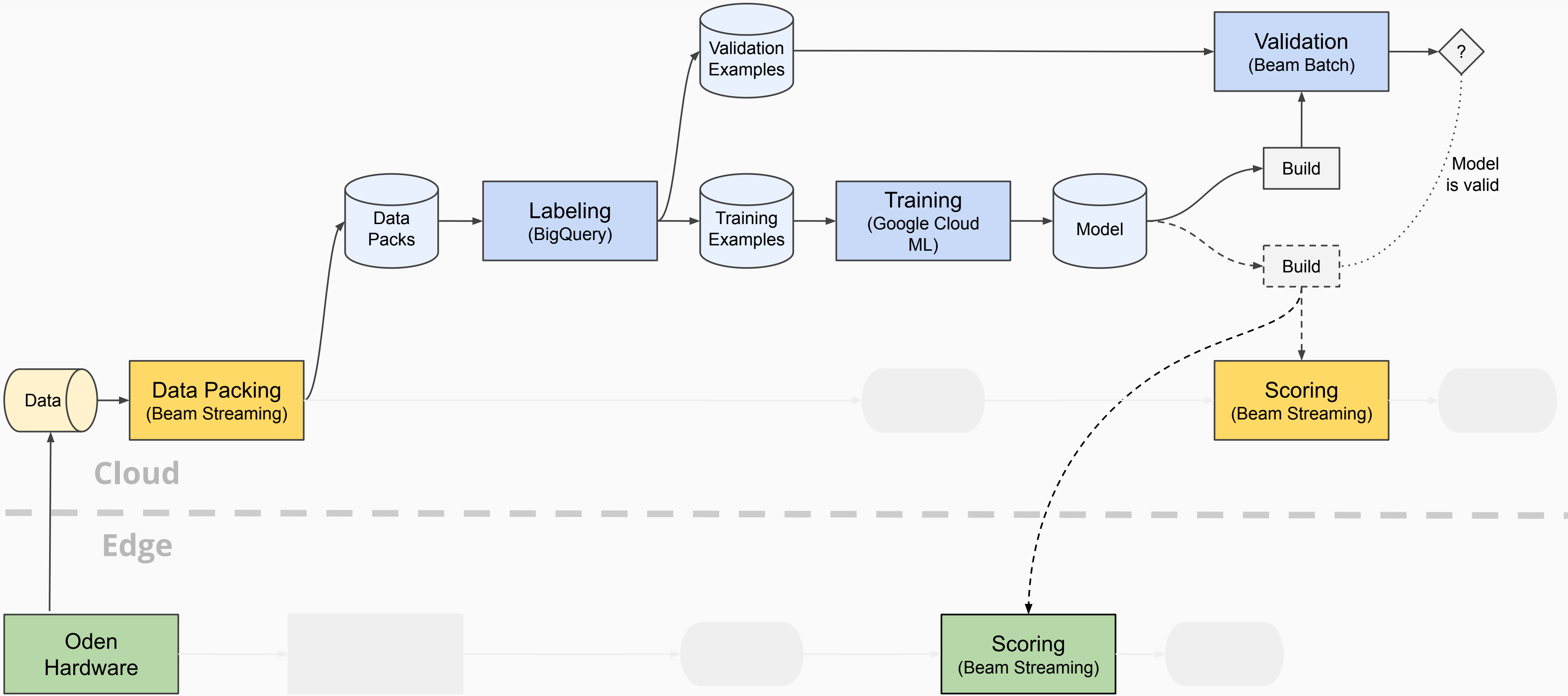
Application: Predictive Quality - Training



Application: Predictive Quality - Validation



Application: Predictive Quality - Deployment (Cloud)

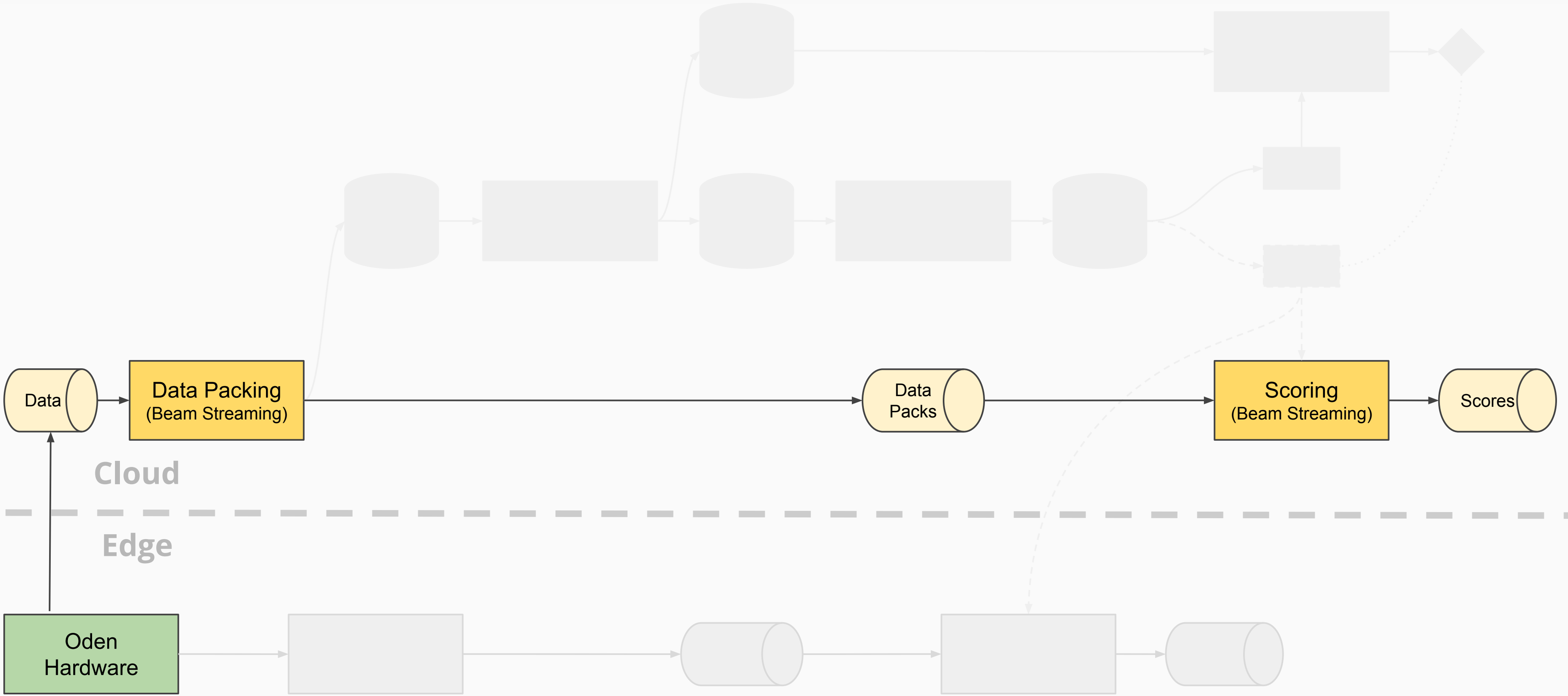


Predictive Model Scoring

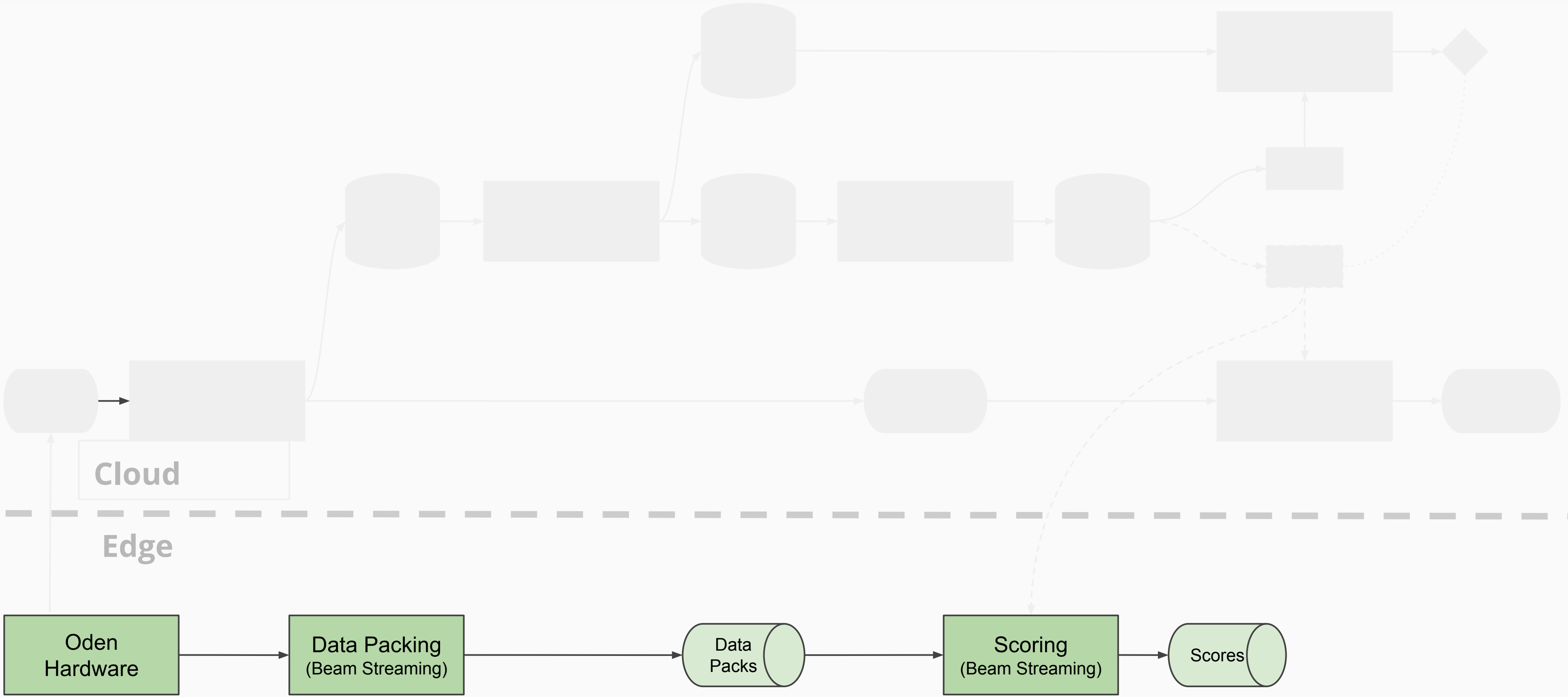
1. Load skpipeline from GCS
2. Read from pubsub/mqtt
3. Create prediction (score)
4. Write predictions to pubsub/mqtt

```
class ModelScoreDoFn(apache_beam.DoFn):
    def process(self, element, skpipeline):
        x = np.array([])
        (id, fv) = element
        if fv is not None:
            x = np.append(x, fv).reshape(1, -1)
            pred = skpipeline.predict(x)
            id_date_list = id.split(".")
            if len(id_date_list) >= 2:
                ts = (pd.to_datetime(id_date_list[1])).timestamp()
                return [json.dumps(...)]

def run(argv=None):
    pipeline_options = PipelineOptions()
    pipeline_options.view_as(SetupOptions).save_main_session = True
    ops = pipeline_options.view_as(UserOptions)

    # Copy the model from Cloud storage
    subprocess.check_call(["gsutil", "cp", ops.modeluri, MODEL_LOCAL_FILE_PATH])  (1)
    skpipeline = joblib.load(MODEL_LOCAL_FILE_PATH)

    # Create the Apache Beam Pipeline
    p = apache_beam.Pipeline(options=pipeline_options)

    preds = (p
        # Read from queue into a PCollection                                     (2)
        | "read" >> ReadFromPubSubOrMqtt(subscription=ops.inputtopic)

        # Parse the JSON messages and then score with the model
        | "parse" >> apache_beam.ParDo(FeaturePacktoFeatureVectorDoFn())
        | "score" >> apache_beam.ParDo(ModelScoreDoFn(), skpipeline)             (3)

        # Write the result to queue
        | "write" >> WriteStringsToPubSubOrMqtt(user_options.outputtopic)        (4)
    )

    result = p.run().wait_until_finish()
```

Application: Predictive Quality - Scoring (Cloud)

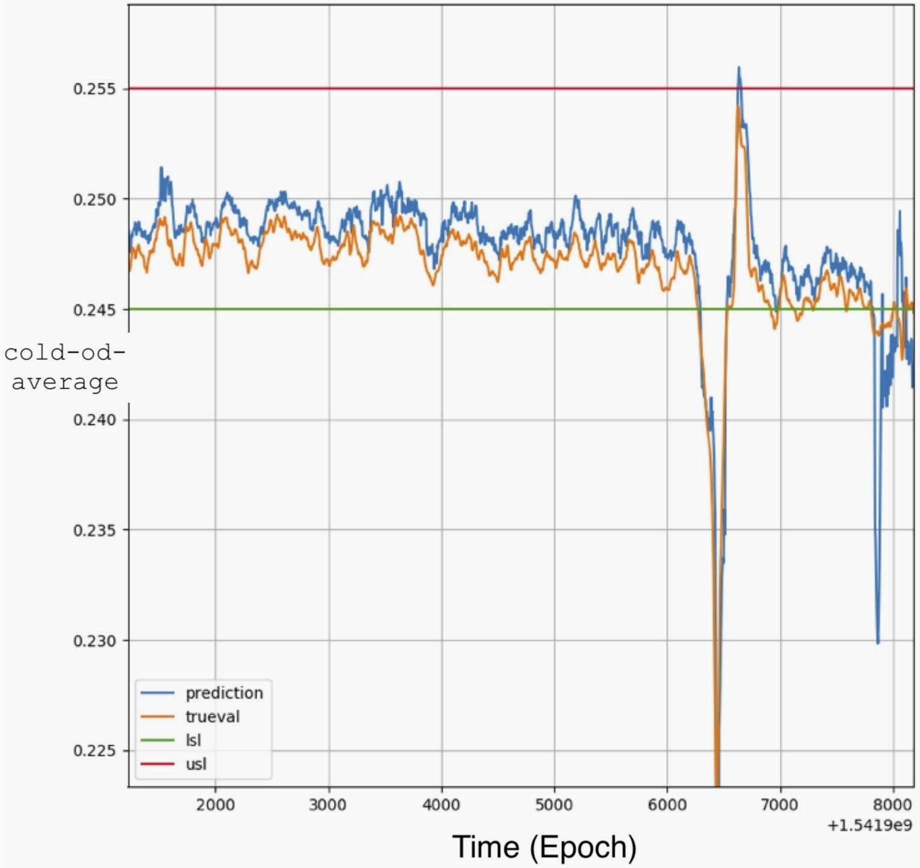


Application: Predictive Quality - Scoring (Edge)



Predictive Quality - Results

- Example with Lasso Model
- MSE of $17e^{-6}$
- 85% precision
- 90% recall
- Acheives max of 430 tuples/sec per worker in cloud
- 325 tuples/sec per worker in cloud
- **Meets customers desired trade-offs for false alarms and missed detection**



Future

- Exposing this framework to customers
- Controlling machines as a result of models
- Internal applications for framework in self-monitoring

In summary

- Infrastructure for optimized data capture opens the possibility for a variety of powerful real-time models.
- Multi-resolution storage of manufacturing data makes exploring and training models easy.
- Workflow for ML model training and validation allows for continuous iterative improvement of models.
- Network-partition resilient deployment of ML to cloud and factory means models can service mission-critical needs in the factory.

Thank You