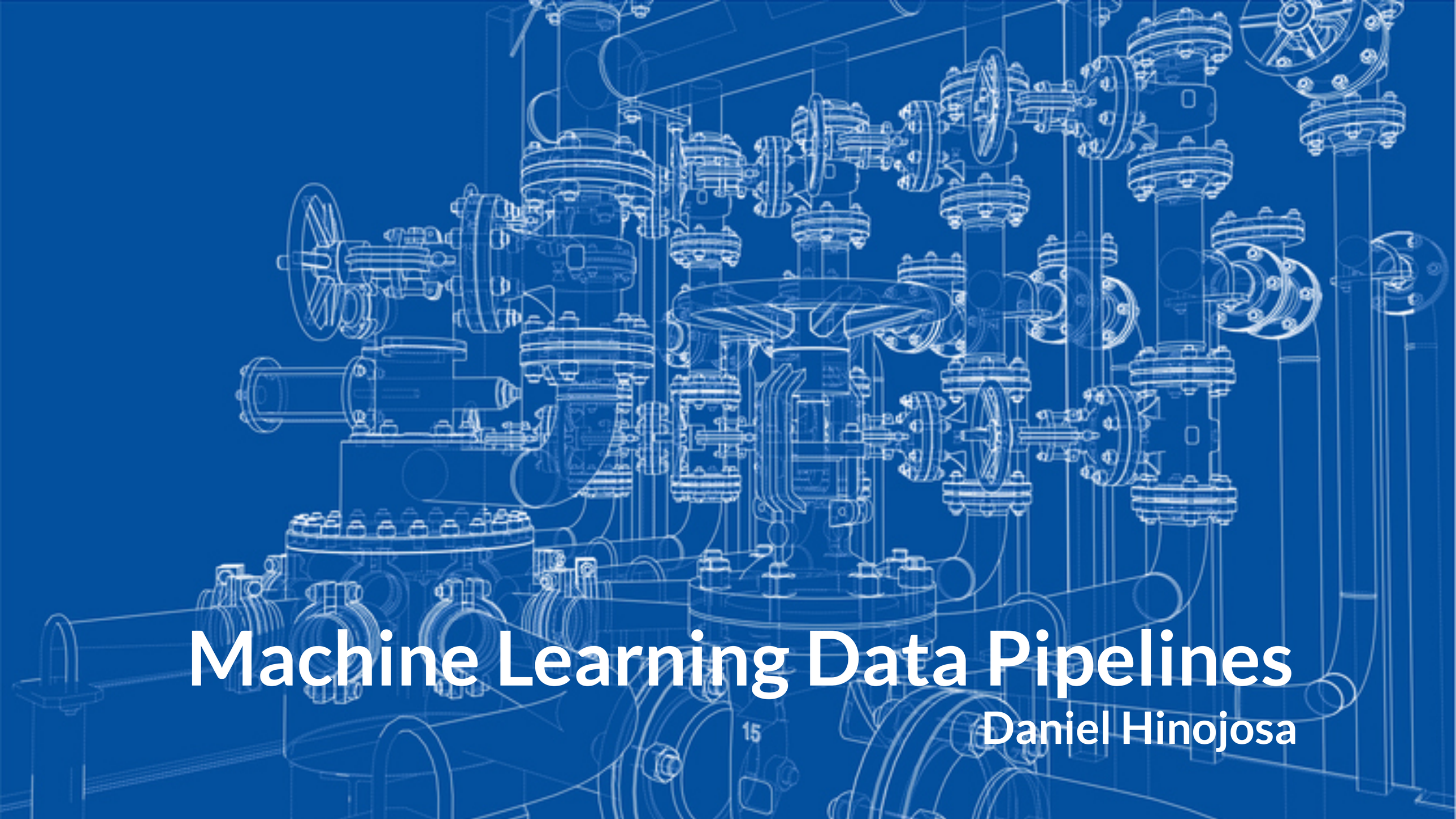
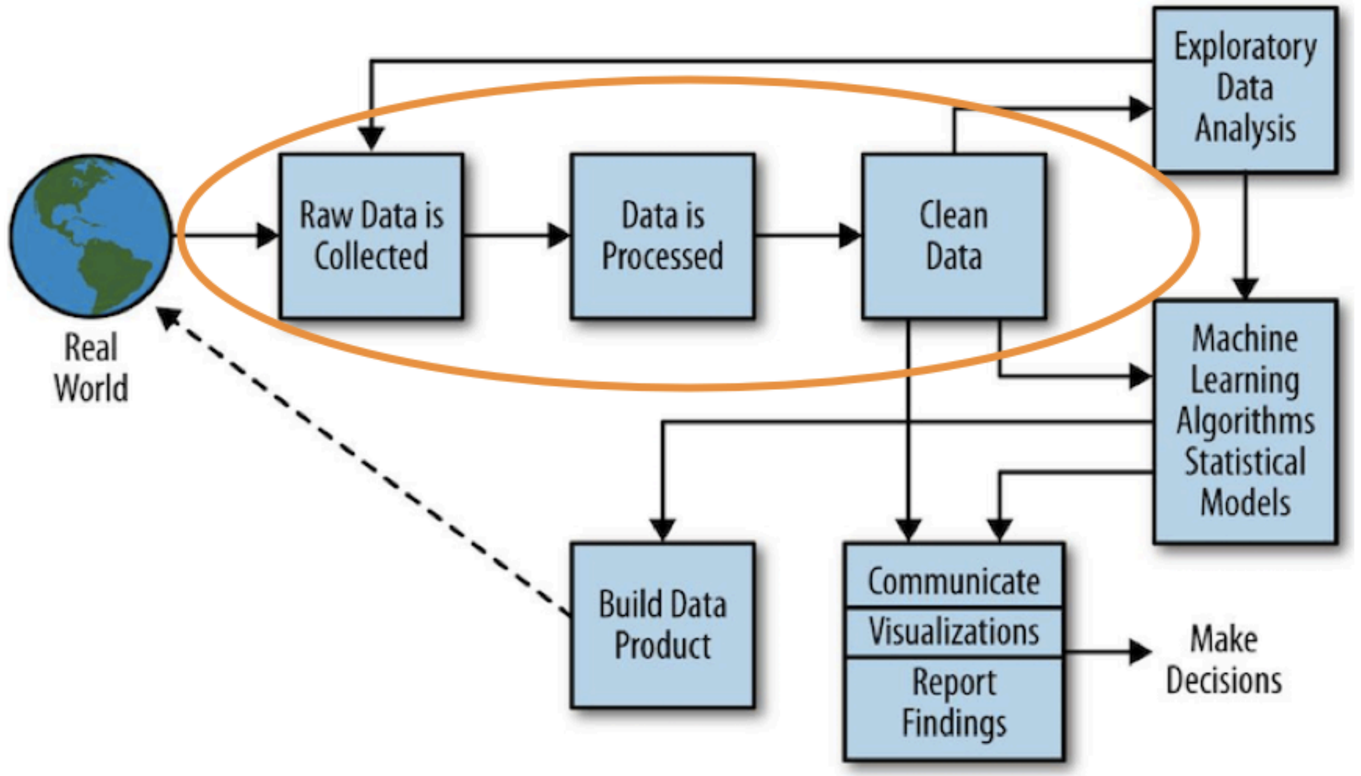


# Machine Learning Data Pipelines

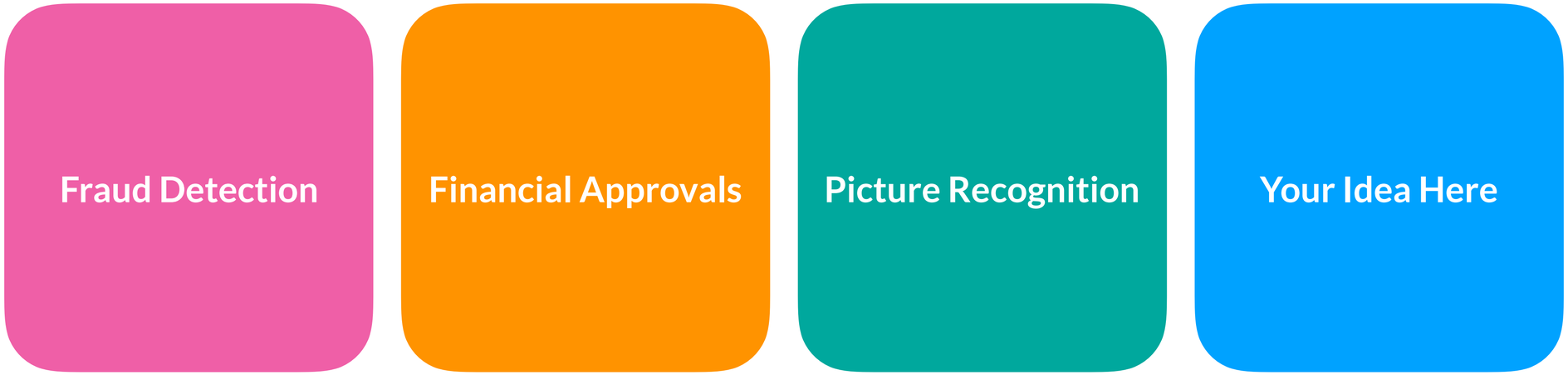
Daniel Hinojosa

# Machine Learning Models

| Machine Learning | Deep Learning |
| --- | --- |
| KNN | Convolutional Neural Networks |
| Naive Bayes | Recurrent Neural Networks |
| Linear Regression | Autoencoders |


Exploratory
Data
Analysis
Raw Data is
Collected
Data is
Processed
Clean
Data
Real
World
Machine
Learning
Algorithms
Statistical
Models
Build Data
Product
Communicate
Visualizations
Report
Findings
Make
Decisions

# Applications



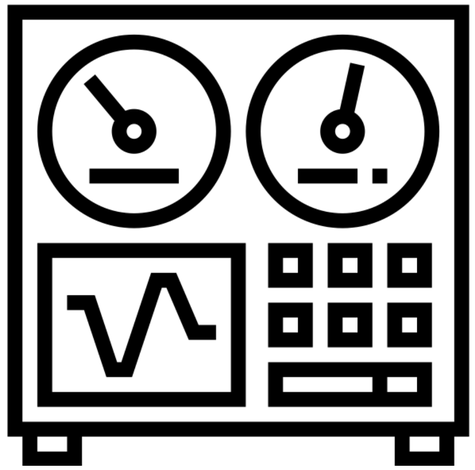
## training

| | | | | |
|---|---|---|---|---|
| 0 | 63 | 1 | 233 | 1 |
| 1 | 37 | 1 | 250 | 0 |
| 2 | 41 | 0 | 204 | 1 |
| 3 | 56 | 1 | 236 | 1 |
| 4 | 57 | 0 | 354 | 0 |
| 5 | 57 | 1 | 192 | 0 |
| 6 | 56 | 0 | 294 | 0 |
| 7 | 44 | 1 | 263 | 1 |
| 8 | 52 | 1 | 199 | 1 |
| 9 | 57 | 1 | 168 | 1 |

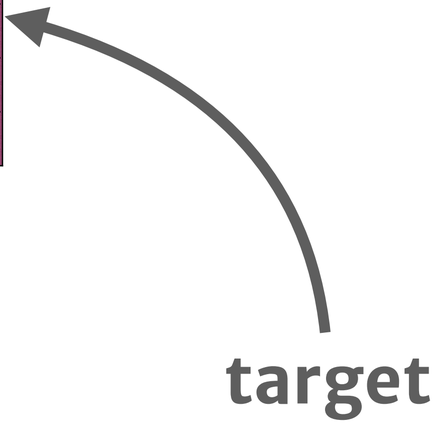
## testing

| | | | | |
|---|---|---|---|---|
| 10 | 54 | 1 | 239 | 1 |
| 11 | 48 | 0 | 275 | 0 |
| 12 | 49 | 1 | 266 | 1 |
| 13 | 64 | 1 | 211 | 0 |
| 14 | 58 | 0 | 283 | 1 |

## training phase



| id | age | sex | trestbps | risk |
|---|---|---|---|---|
| 0 | 20 | 1 | 145 | 1 |
| 1 | 45 | 1 | 130 | 1 |
| 2 | 43 | 0 | 130 | 0 |
| 3 | 33 | 1 | 150 | 0 |

training

testing

| id | age | sex | trestbps | target |
|---|---|---|---|---|
| 0 | 63 | 1 | 233 | 1 |
| 1 | 37 | 1 | 250 | 0 |
| 2 | 41 | 0 | 204 | 1 |
| 3 | 56 | 1 | 236 | 1 |
| 4 | 57 | 0 | 354 | 0 |
| 5 | 57 | 1 | 192 | 0 |
| 6 | 56 | 0 | 294 | 0 |
| 7 | 44 | 1 | 263 | 1 |
| 8 | 52 | 1 | 199 | 1 |
| 9 | 57 | 1 | 168 | 1 |
| 10 | 54 | 1 | 239 | 1 |
| 11 | 48 | 0 | 275 | 0 |
| 12 | 49 | 1 | 266 | 1 |
| 13 | 64 | 1 | 211 | 0 |
| 14 | 58 | 0 | 283 | 1 |

| id | age | sex | trestbps | target |
|---|---|---|---|---|
| 0 | 63 | 1 | 233 | 1 |
| 1 | 37 | 1 | 250 | 0 |
| 2 | 41 | 0 | 204 | 1 |
| 3 | 56 | 1 | 236 | 1 |
| 4 | 57 | 0 | 354 | 0 |
| 5 | 57 | 1 | 192 | 0 |
| 6 | 56 | 0 | 294 | 0 |
| 7 | 44 | 1 | 263 | 1 |
| 8 | 52 | 1 | 199 | 1 |
| 9 | 57 | 1 | 168 | 1 |
| 10 | 54 | 1 | 239 | 1 |
| 11 | 48 | 0 | 275 | 0 |
| 12 | 49 | 1 | 266 | 1 |
| 13 | 64 | 1 | 211 | 0 |
| 14 | 58 | 0 | 283 | 1 |

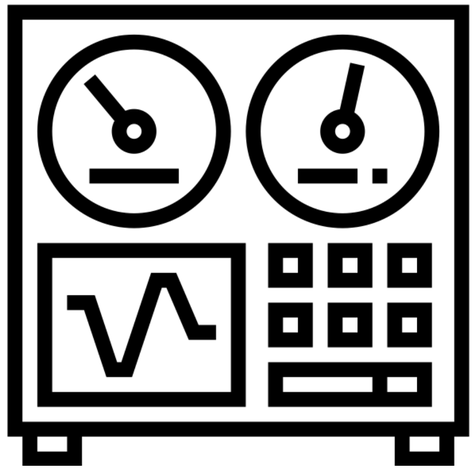
random distribution between training and testing

## training

| | | | | |
|---|---|---|---|---|
| 0 | 63 | 1 | 233 | 1 |
| 1 | 37 | 1 | 250 | 0 |
| 2 | 41 | 0 | 204 | 1 |
| 3 | 56 | 1 | 236 | 1 |
| 4 | 57 | 0 | 354 | 0 |
| 5 | 57 | 1 | 192 | 0 |
| 6 | 56 | 0 | 294 | 0 |
| 7 | 44 | 1 | 263 | 1 |
| 8 | 52 | 1 | 199 | 1 |
| 9 | 57 | 1 | 168 | 1 |

## testing

| | | | | |
|---|---|---|---|---|
| 10 | 54 | 1 | 239 | 1 |
| 11 | 48 | 0 | 275 | 0 |
| 12 | 49 | 1 | 266 | 1 |
| 13 | 64 | 1 | 211 | 0 |
| 14 | 58 | 0 | 283 | 1 |



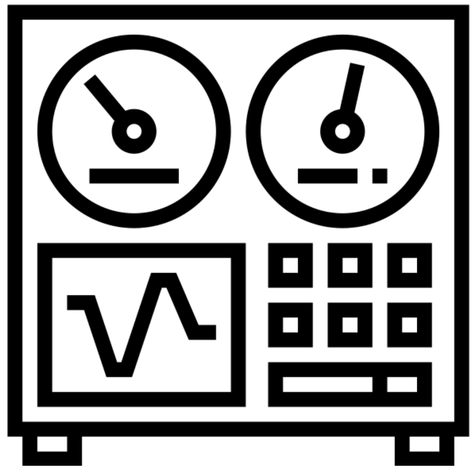
## training

| | | | | |
|---|---|---|---|---|
| 0 | 63 | 1 | 233 | 1 |
| 1 | 37 | 1 | 250 | 0 |
| 2 | 41 | 0 | 204 | 1 |
| 3 | 56 | 1 | 236 | 1 |
| 4 | 57 | 0 | 354 | 0 |
| 5 | 57 | 1 | 192 | 0 |
| 6 | 56 | 0 | 294 | 0 |
| 7 | 44 | 1 | 263 | 1 |
| 8 | 52 | 1 | 199 | 1 |
| 9 | 57 | 1 | 168 | 1 |

## testing

| | | | | |
|---|---|---|---|---|
| 10 | 54 | 1 | 239 | 1 |
| 11 | 48 | 0 | 275 | 0 |
| 12 | 49 | 1 | 266 | 1 |
| 13 | 64 | 1 | 211 | 0 |
| 14 | 58 | 0 | 283 | 1 |

model →



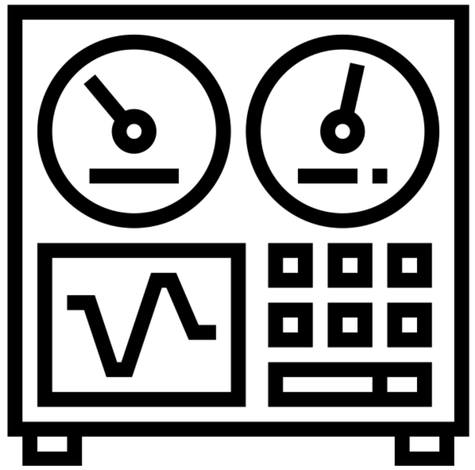
## training

| | | | | |
|---|---|---|---|---|
| 0 | 63 | 1 | 233 | 1 |
| 1 | 37 | 1 | 250 | 0 |
| 2 | 41 | 0 | 204 | 1 |
| 3 | 56 | 1 | 236 | 1 |
| 4 | 57 | 0 | 354 | 0 |
| 5 | 57 | 1 | 192 | 0 |
| 6 | 56 | 0 | 294 | 0 |
| 7 | 44 | 1 | 263 | 1 |
| 8 | 52 | 1 | 199 | 1 |
| 9 | 57 | 1 | 168 | 1 |

## testing

| | | | | |
|---|---|---|---|---|
| 10 | 54 | 1 | 239 | 1 |
| 11 | 48 | 0 | 275 | 0 |
| 12 | 49 | 1 | 266 | 1 |
| 13 | 64 | 1 | 211 | 0 |
| 14 | 58 | 0 | 283 | 1 |

## training phase

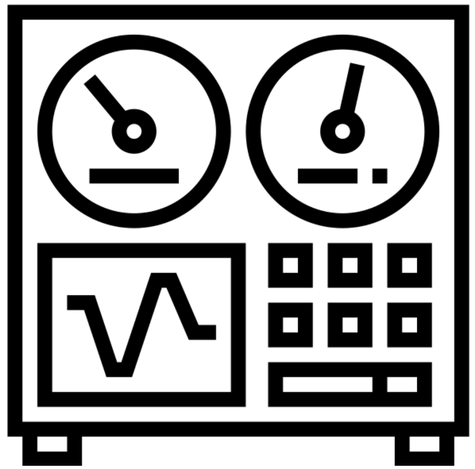


## training

| | | | | |
|---|---|---|---|---|
| 0 | 63 | 1 | 233 | 1 |
| 1 | 37 | 1 | 250 | 0 |
| 2 | 41 | 0 | 204 | 1 |
| 3 | 56 | 1 | 236 | 1 |
| 4 | 57 | 0 | 354 | 0 |
| 5 | 57 | 1 | 192 | 0 |
| 6 | 56 | 0 | 294 | 0 |
| 7 | 44 | 1 | 263 | 1 |
| 8 | 52 | 1 | 199 | 1 |
| 9 | 57 | 1 | 168 | 1 |

## testing

| | | | | |
|---|---|---|---|---|
| 10 | 54 | 1 | 239 | 1 |
| 11 | 48 | 0 | 275 | 0 |
| 12 | 49 | 1 | 266 | 1 |
| 13 | 64 | 1 | 211 | 0 |
| 14 | 58 | 0 | 283 | 1 |



## training

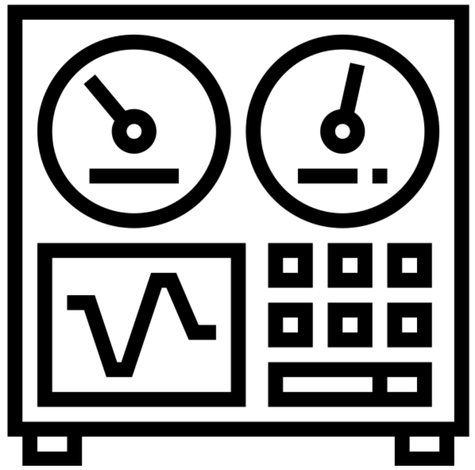
| | | | | |
|---|---|---|---|---|
| 0 | 63 | 1 | 233 | 1 |
| 1 | 37 | 1 | 250 | 0 |
| 2 | 41 | 0 | 204 | 1 |
| 3 | 56 | 1 | 236 | 1 |
| 4 | 57 | 0 | 354 | 0 |
| 5 | 57 | 1 | 192 | 0 |
| 6 | 56 | 0 | 294 | 0 |
| 7 | 44 | 1 | 263 | 1 |
| 8 | 52 | 1 | 199 | 1 |
| 9 | 57 | 1 | 168 | 1 |

## testing

| | | | |
|---|---|---|---|
| 10 | 54 | 1 | 239 |
| 11 | 48 | 0 | 275 |
| 12 | 49 | 1 | 266 |
| 13 | 64 | 1 | 211 |
| 14 | 58 | 0 | 283 |

## actual

| |
|---|
| 1 |
| 0 |
| 1 |
| 0 |
| 1 |



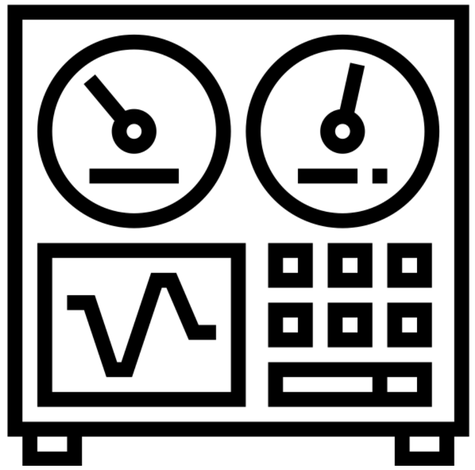
## reserve the actual later for verification

## training

| | | | | |
|---|---|---|---|---|
| 0 | 63 | 1 | 233 | 1 |
| 1 | 37 | 1 | 250 | 0 |
| 2 | 41 | 0 | 204 | 1 |
| 3 | 56 | 1 | 236 | 1 |
| 4 | 57 | 0 | 354 | 0 |
| 5 | 57 | 1 | 192 | 0 |
| 6 | 56 | 0 | 294 | 0 |
| 7 | 44 | 1 | 263 | 1 |
| 8 | 52 | 1 | 199 | 1 |
| 9 | 57 | 1 | 168 | 1 |

## testing

| | | | |
|---|---|---|---|
| 10 | 54 | 1 | 239 |
| 11 | 48 | 0 | 275 |
| 12 | 49 | 1 | 266 |
| 13 | 64 | 1 | 211 |
| 14 | 58 | 0 | 283 |



## testing phase

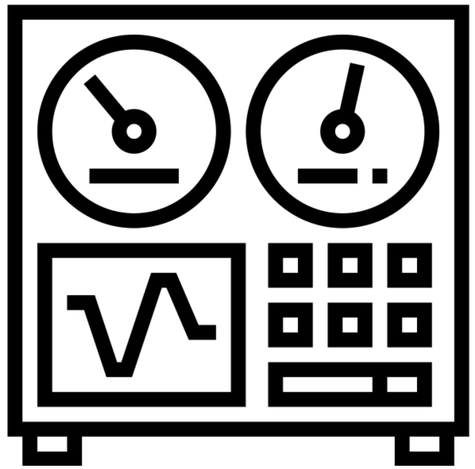
## training

| | | | | |
|---|---|---|---|---|
| 0 | 63 | 1 | 233 | 1 |
| 1 | 37 | 1 | 250 | 0 |
| 2 | 41 | 0 | 204 | 1 |
| 3 | 56 | 1 | 236 | 1 |
| 4 | 57 | 0 | 354 | 0 |
| 5 | 57 | 1 | 192 | 0 |
| 6 | 56 | 0 | 294 | 0 |
| 7 | 44 | 1 | 263 | 1 |
| 8 | 52 | 1 | 199 | 1 |
| 9 | 57 | 1 | 168 | 1 |

## testing

| | | | |
|---|---|---|---|
| 10 | 54 | 1 | 239 |
| 11 | 48 | 0 | 275 |
| 12 | 49 | 1 | 266 |
| 13 | 64 | 1 | 211 |
| 14 | 58 | 0 | 283 |

!!!!!!!



## generated result

| |
|---|
| 1 |
| 0 |
| 1 |
| 1 |
| 1 |

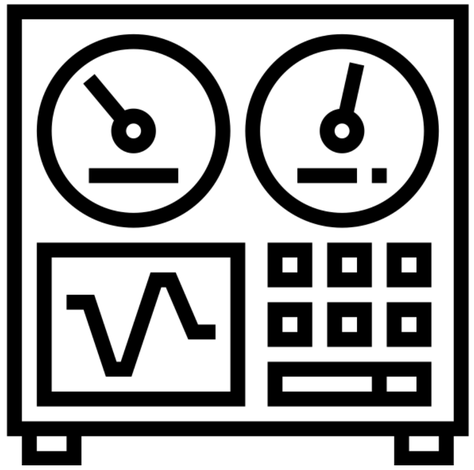
## training

| | | | | |
|---|---|---|---|---|
| 0 | 63 | 1 | 233 | 1 |
| 1 | 37 | 1 | 250 | 0 |
| 2 | 41 | 0 | 204 | 1 |
| 3 | 56 | 1 | 236 | 1 |
| 4 | 57 | 0 | 354 | 0 |
| 5 | 57 | 1 | 192 | 0 |
| 6 | 56 | 0 | 294 | 0 |
| 7 | 44 | 1 | 263 | 1 |
| 8 | 52 | 1 | 199 | 1 |
| 9 | 57 | 1 | 168 | 1 |

## testing

| | | | |
|---|---|---|---|
| 10 | 54 | 1 | 239 |
| 11 | 48 | 0 | 275 |
| 12 | 49 | 1 | 266 |
| 13 | 64 | 1 | 211 |
| 14 | 58 | 0 | 283 |

!!!!!!!



## generated result

| |
|---|
| 1 |
| 0 |
| 1 |
| 1 |
| 1 |

## actual

| |
|---|
| 1 |
| 0 |
| 1 |
| 0 |
| 1 |

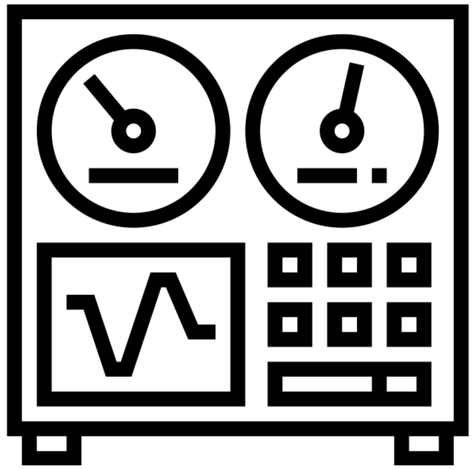
## training

| | | | | |
|---|---|---|---|---|
| 0 | 63 | 1 | 233 | 1 |
| 1 | 37 | 1 | 250 | 0 |
| 2 | 41 | 0 | 204 | 1 |
| 3 | 56 | 1 | 236 | 1 |
| 4 | 57 | 0 | 354 | 0 |
| 5 | 57 | 1 | 192 | 0 |
| 6 | 56 | 0 | 294 | 0 |
| 7 | 44 | 1 | 263 | 1 |
| 8 | 52 | 1 | 199 | 1 |
| 9 | 57 | 1 | 168 | 1 |

## testing

| | | | |
|---|---|---|---|
| 10 | 54 | 1 | 239 |
| 11 | 48 | 0 | 275 |
| 12 | 49 | 1 | 266 |
| 13 | 64 | 1 | 211 |
| 14 | 58 | 0 | 283 |

!!!!!!!



## generated result

| |
|---|
| 1 |
| 0 |
| 1 |
| 1 |
| 1 |

## actual

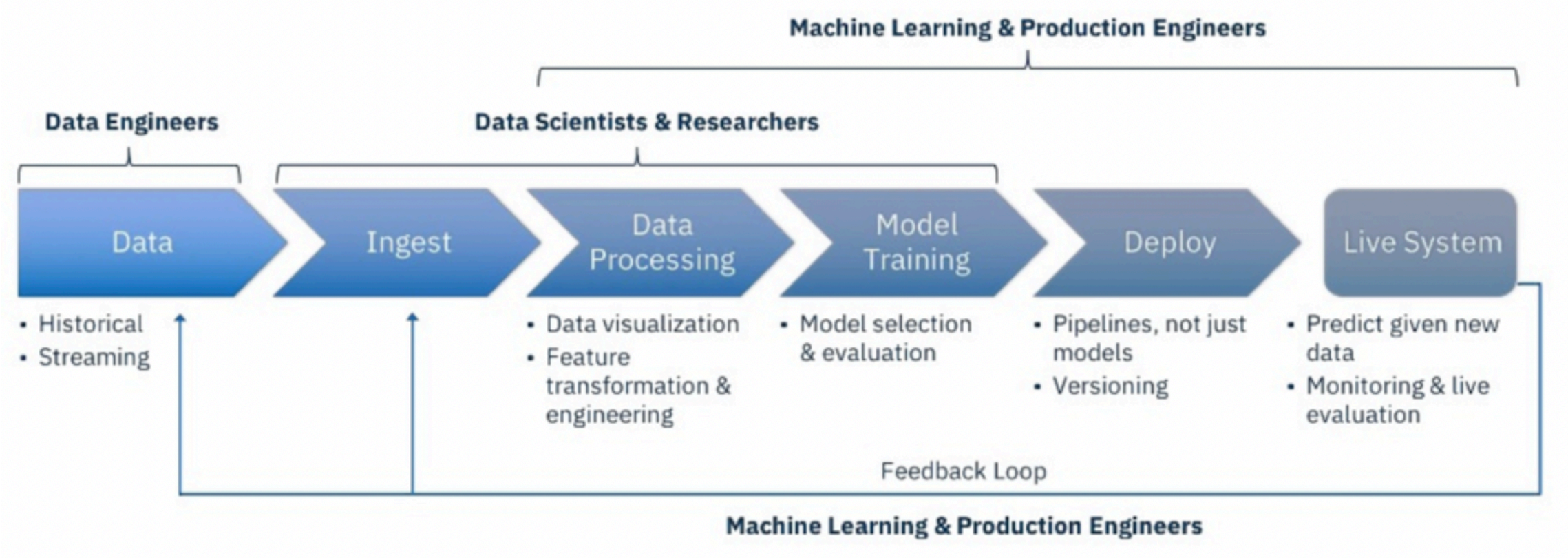
| |
|---|
| 1 |
| 0 |
| 1 |
| 0 |
| 1 |

# How did we do?

| generated result | actual |
| --- | --- |
| 1 | 1 |
| 0 | 0 |
| 1 | 1 |
| 1 | 0 |
| 1 | 1 |

# How did we do?

# Roles and Responsibilities

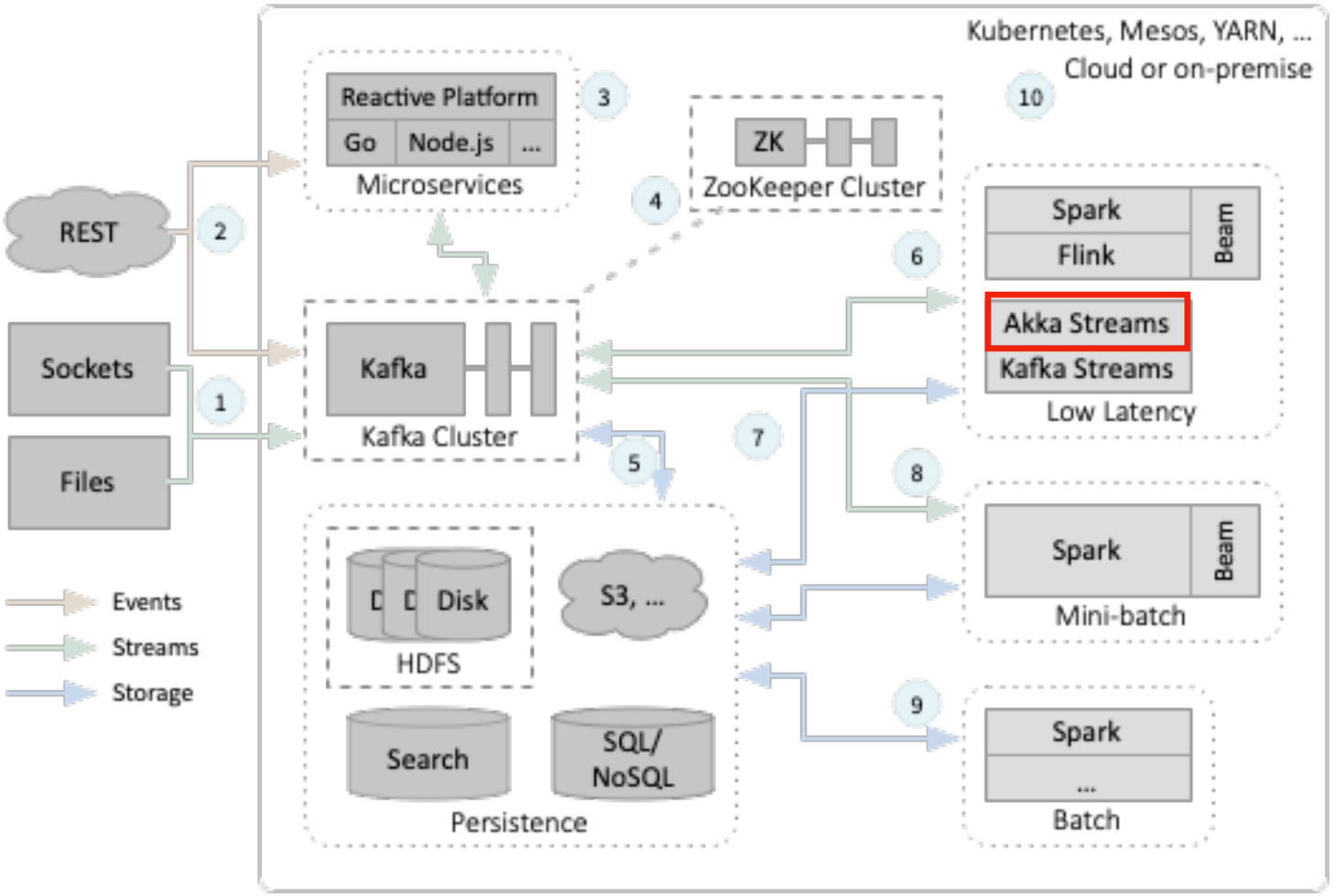


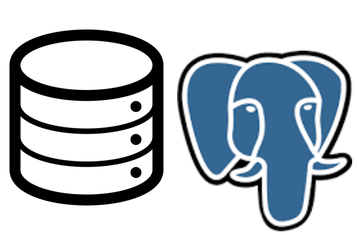
# What is MLOps?

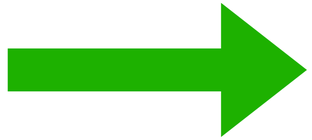
**MLOps (a compound of “machine learning” and “operations”) is a practice for collaboration and communication between data scientists and operations professionals to help manage production ML lifecycles.**

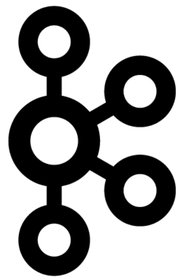

Java™

# Dean Wampler Architecture

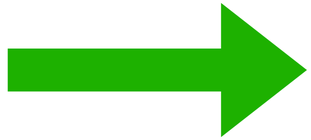


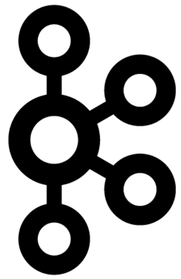




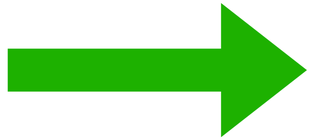


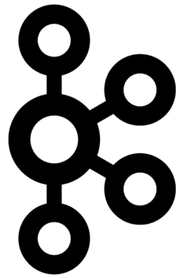
Kafka Client Producer

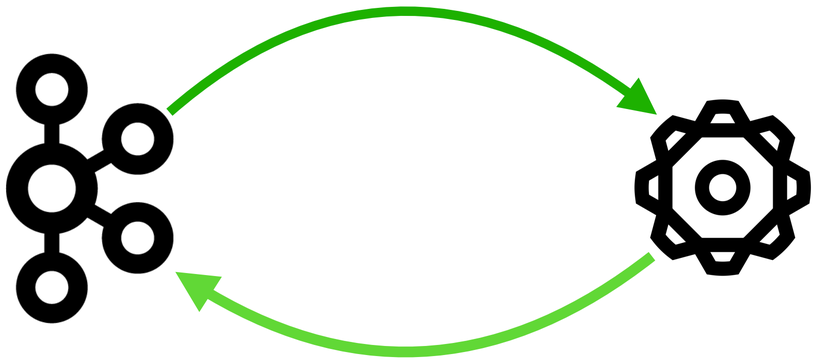




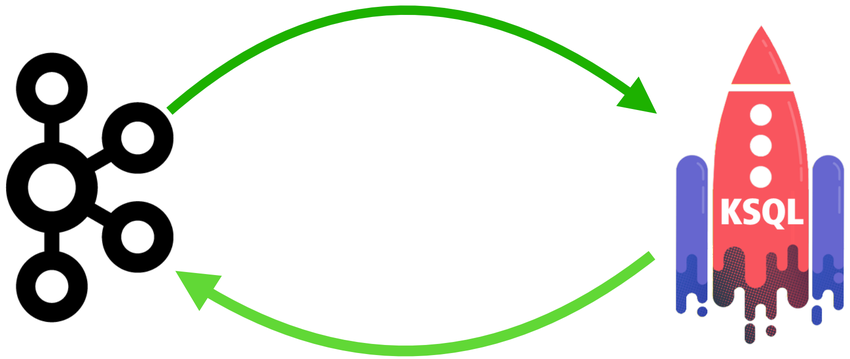
Kafka Rest Proxy





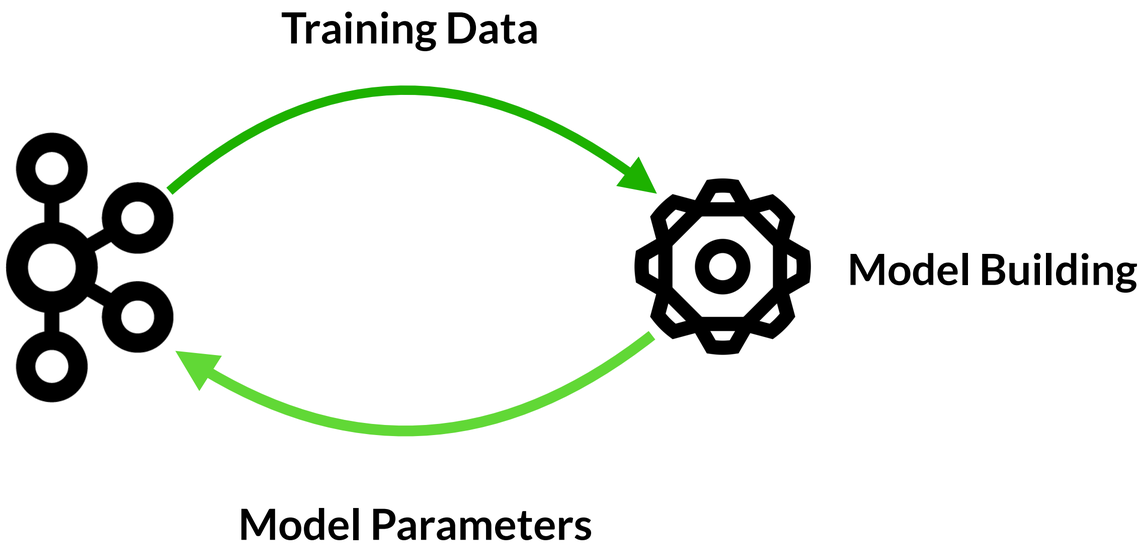


**Kafka Streams**

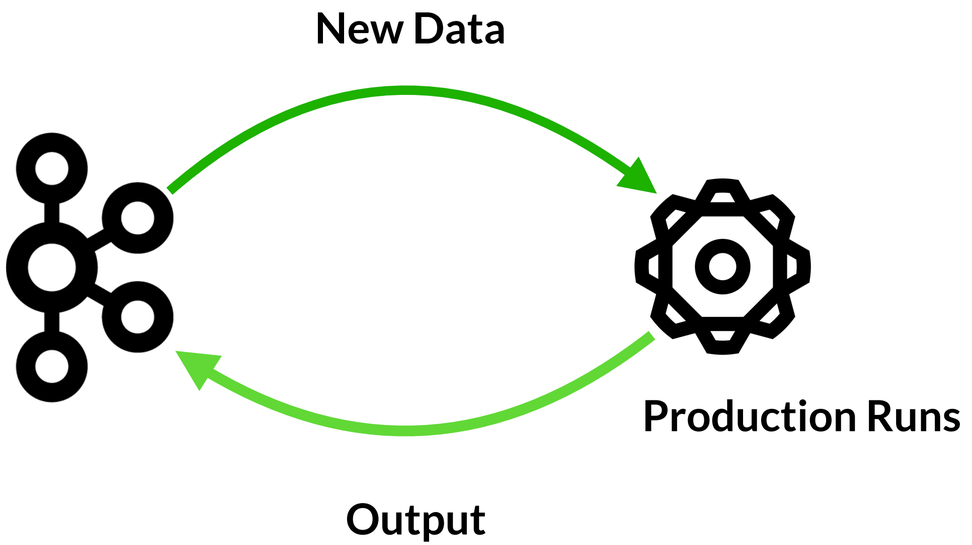



KSQL

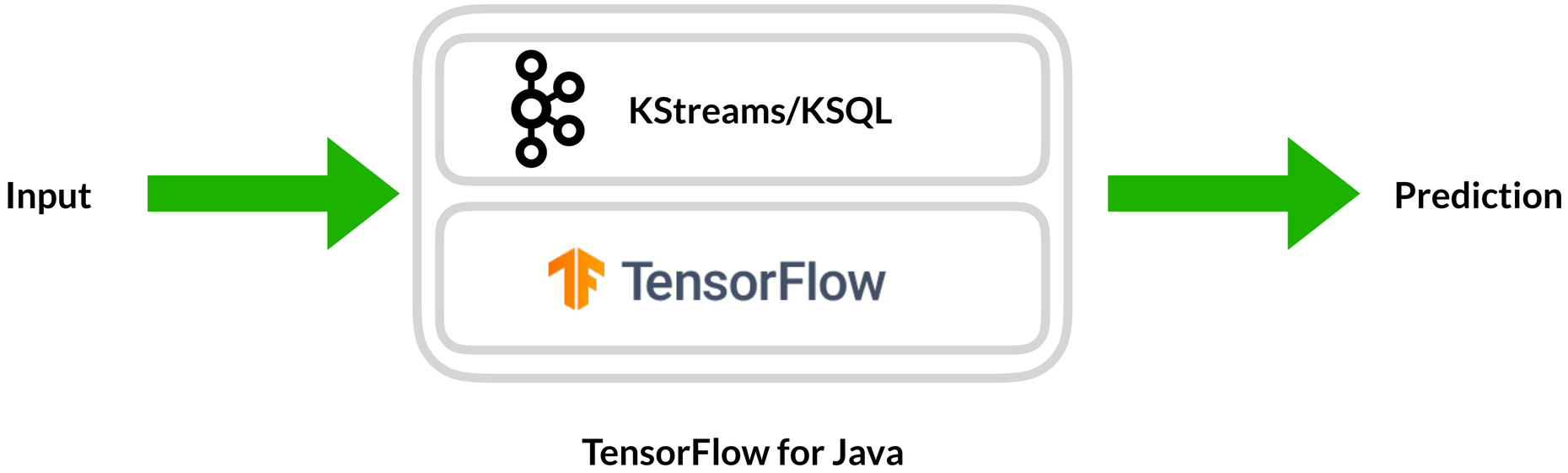
# Phase One: Model Building



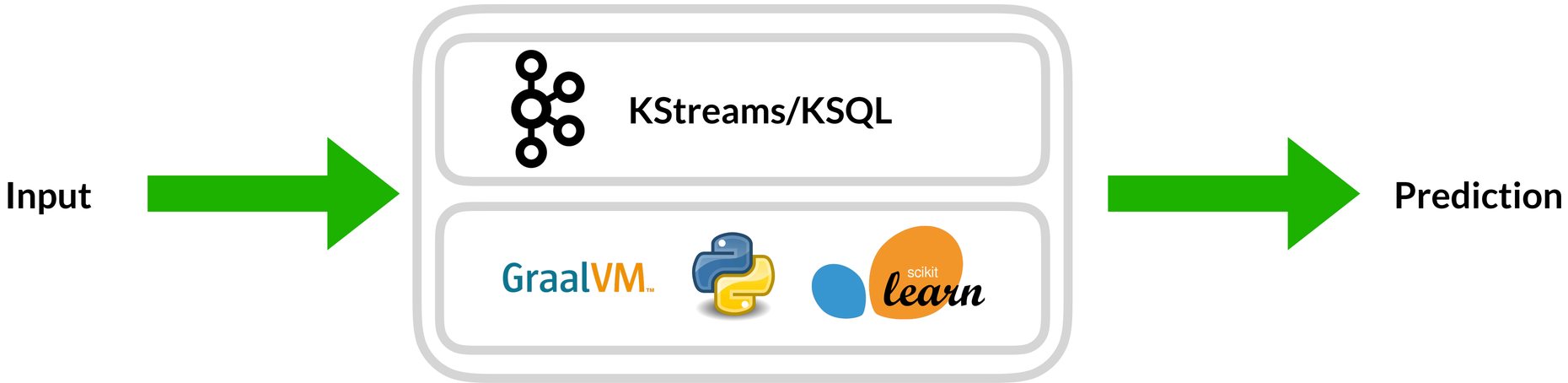
# Phase Two: Production Run



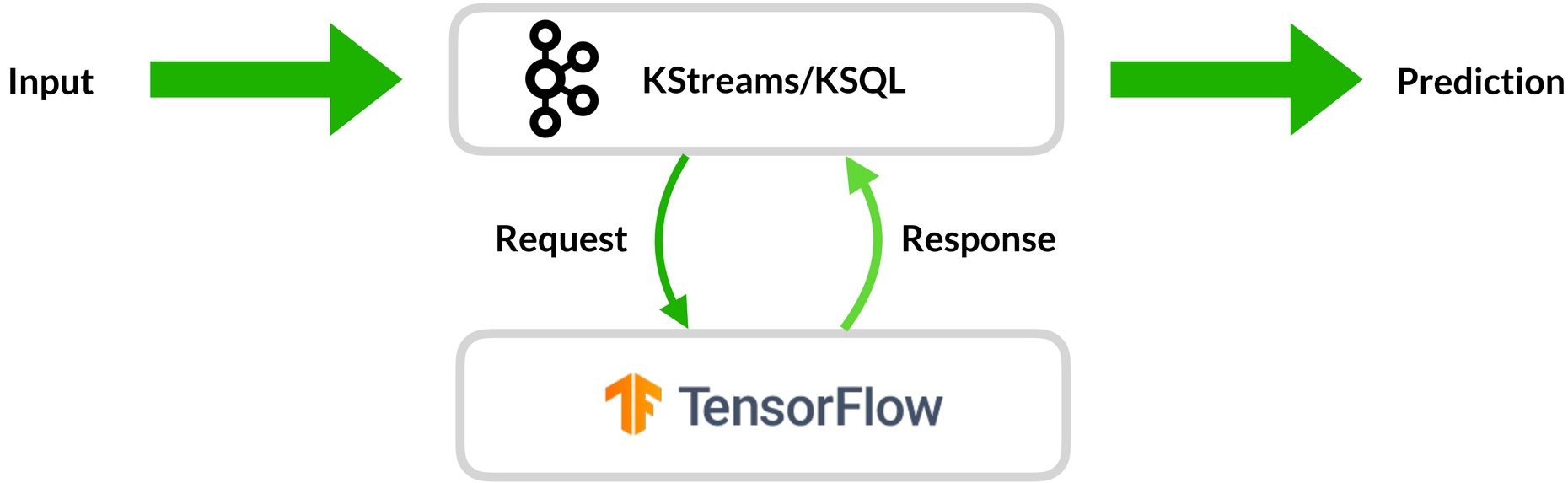
# Model Deployments : Embedded



# Model Deployments : Embedded with GraalVM



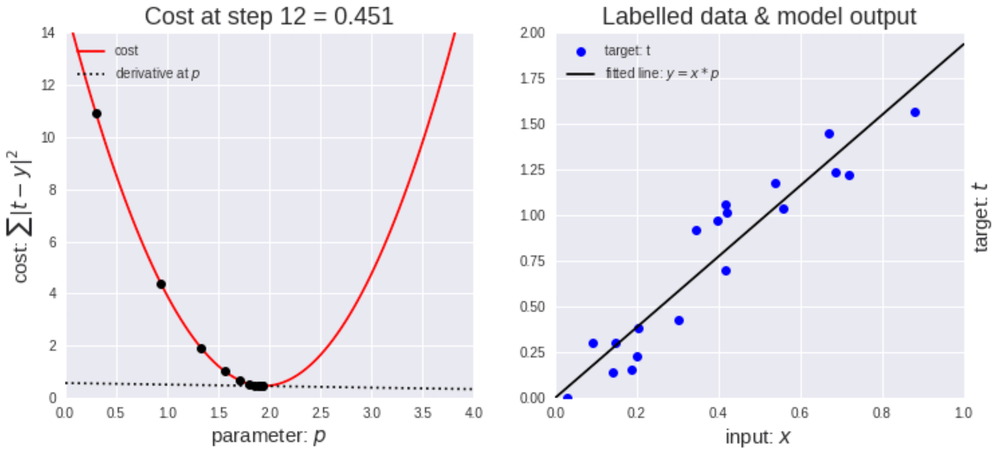
# Model Deployments : Tensor Flow Serving



# Model Deployments : Tensor Flow Serving

```
python tensor_flow_saved_model.py \
  --training_iteration=100 \
  --model_version=1 /tmp/my_model
```

**Iterations - Number of iterations until you find the optical minima**




Source: https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9

# Model Deployments : Tensor Flow Serving

```
$ ls /tmp/my_model
1  2
```

**Each subsequent run will have better and better model, as you refine**

# Model Deployments : Preparing Tensor Flow Serving

```
mkdir /tmp/monitored
cp -r /tmp/mnist/1 /tmp/monitored
```

**Move your model into a monitored directory**

# Model Deployments : Starting Tensor Flow Serving

```
docker run -p 8500:8500 \
--mount type=bind,source=/tmp/monitored,target=/models/my_model \
-t --entrypoint=tensorflow_model_server \
tensorflow/serving --enable_batching \
--port=8500 --model_name=my_model --model_base_path=/models/
my_model &
```

**Move your model into a monitored directory**

# Model Deployments : Starting Tensor Flow Serving

```
curl -d '{"instances": [1.0, 2.0, 5.0]}' \
    -X POST http://localhost:8500/v1/models/my_model:predict
```

**Call the web service**

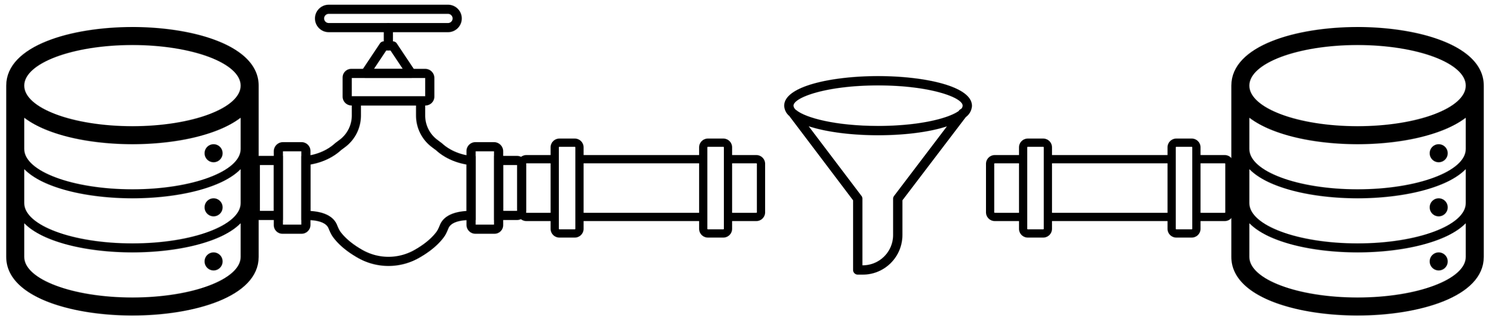

Java™

# Kafka Streams


NY
100.00

NY

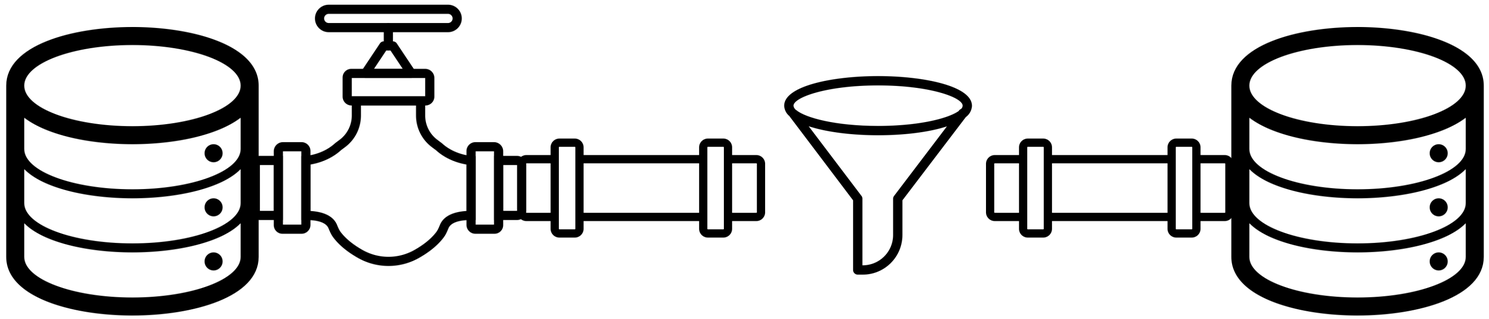
100.00



`value > 10000`

OH

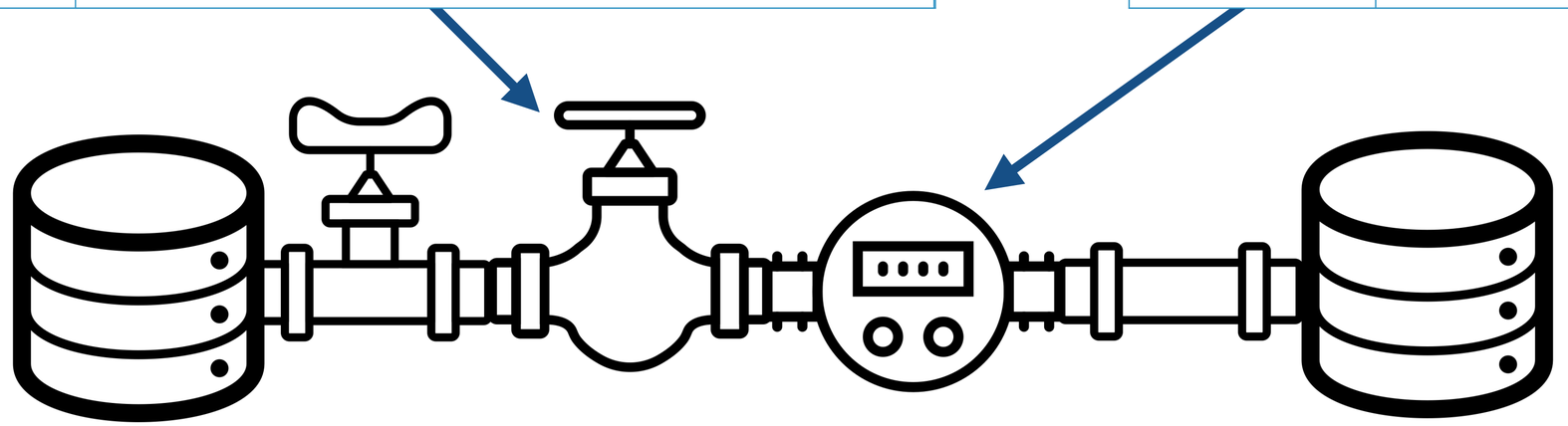
20000



```
value > 10000
```

| Key | Value |
|---|---|
| **OH** | 100.0 |
| | |
| | |
| | |

| Key | Value |
|---|---|
| **OH** | 100.0 |
| | |
| | |
| | |



OH

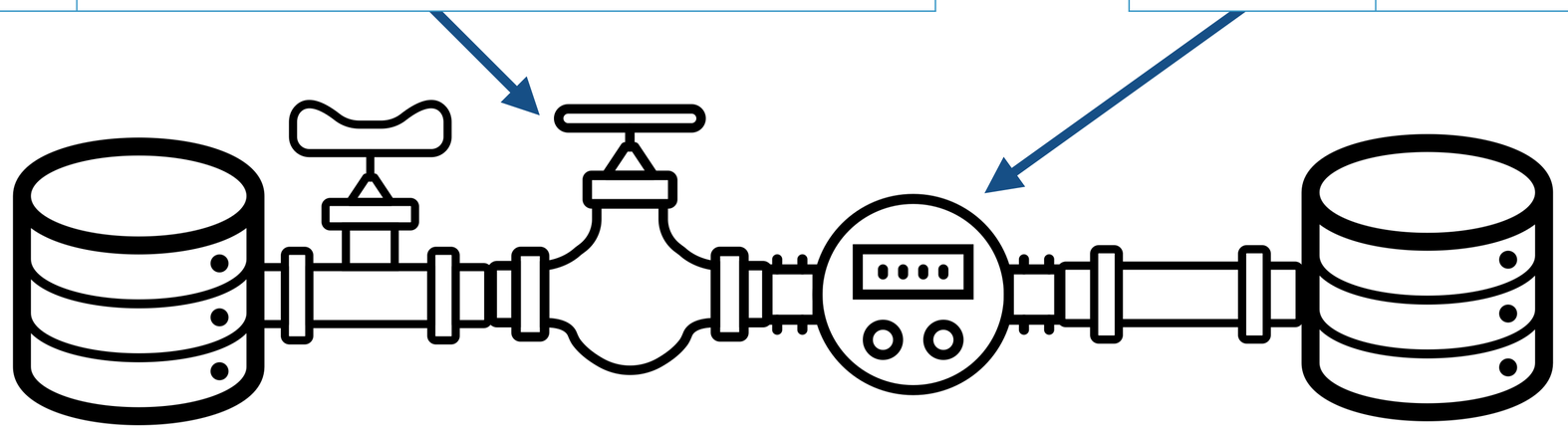
100.00

OH

100.00

| Key | Value |
|---|---|
| **OH** | 100.0 |
| **NY** | 20.0 |
| | |
| | |

| Key | Value |
|---|---|
| **OH** | 100.0 |
| **NY** | 20.0 |
| | |
| | |



NY

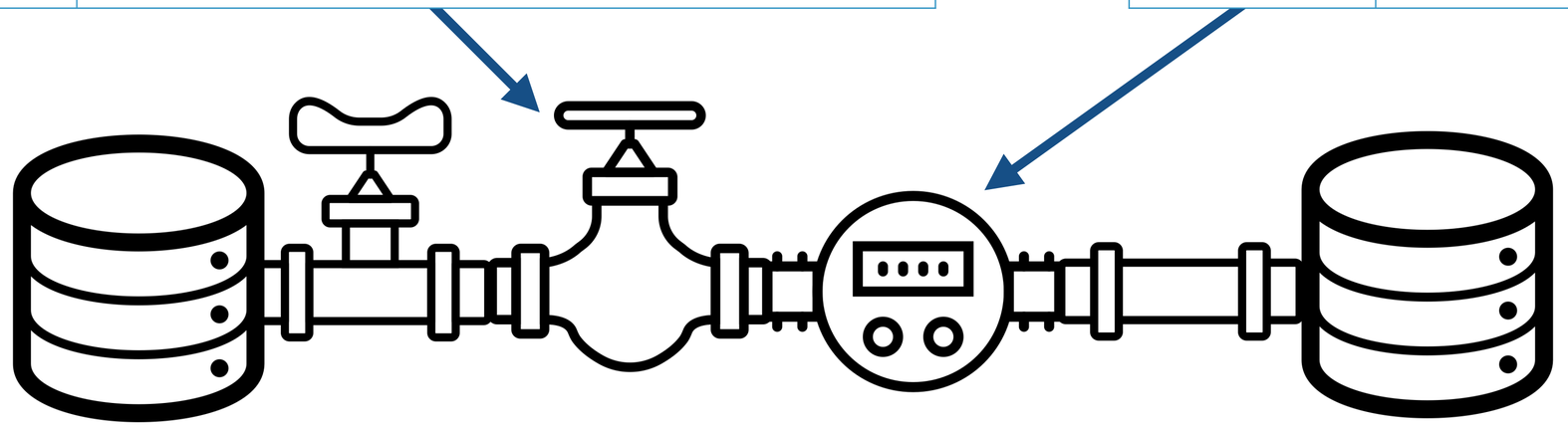
20

NY

20

| Key | Value |
|---|---|
| **OH** | 100.0 |
| **NY** | 20.0, 60.0 |
| | |
| | |

| Key | Value |
|---|---|
| **OH** | 100.0 |
| **NY** | 80.0 |
| | |
| | |

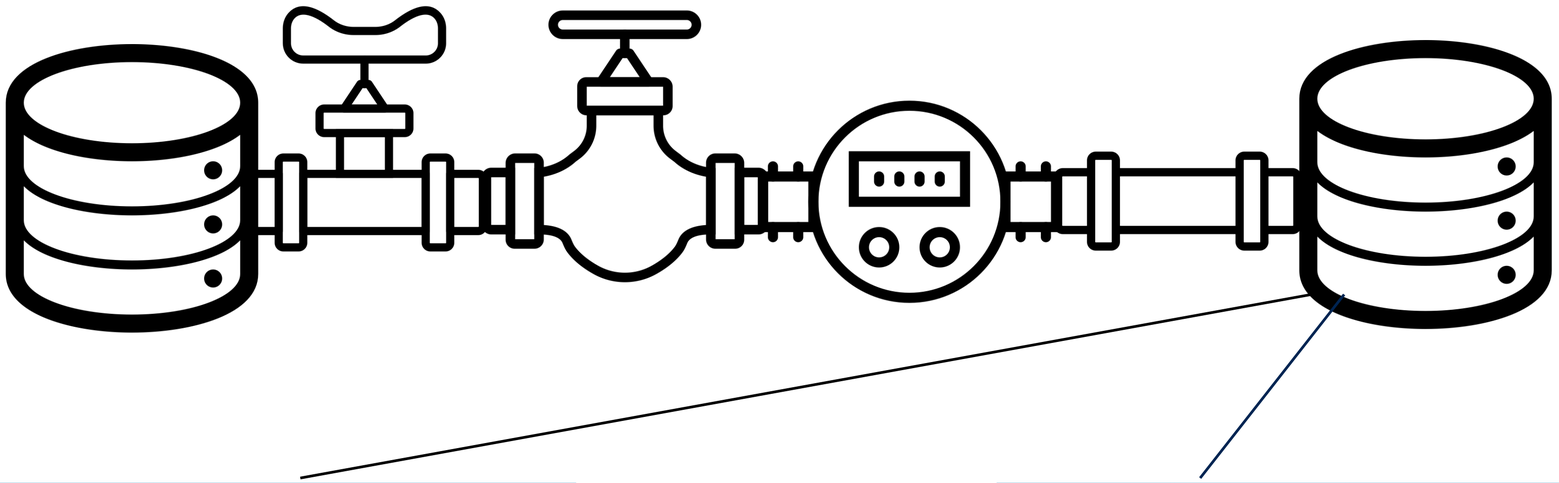


NY

60

NY

80.0

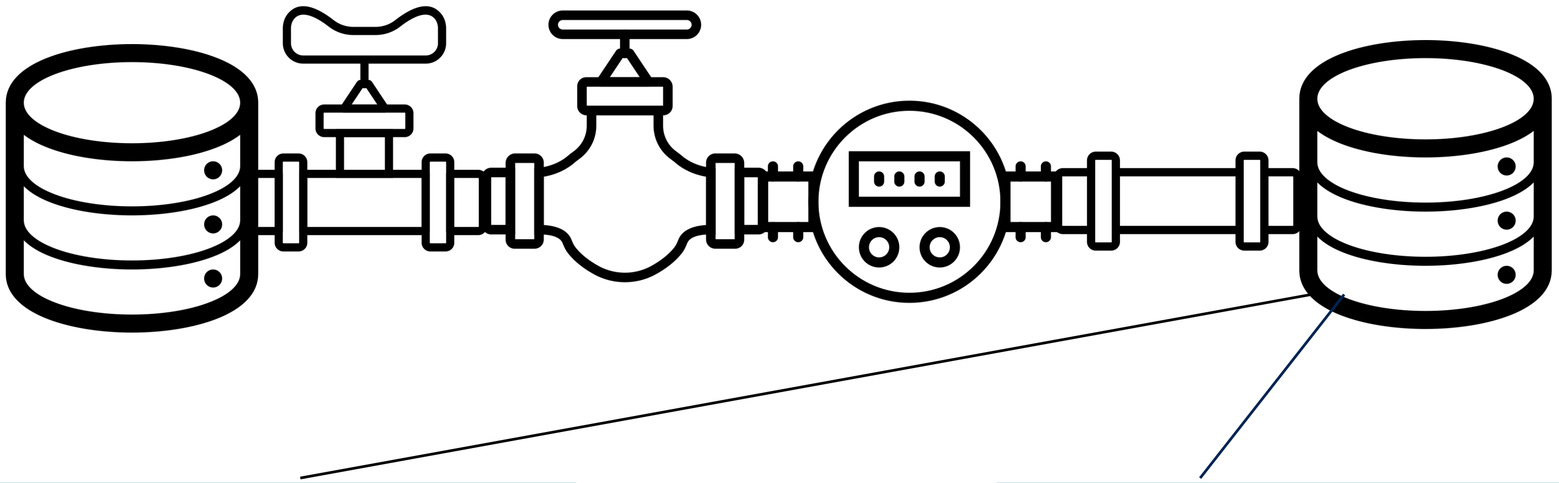


| Offset | Key | Value |
|---|---|---|
| **0** | **OH** | 100.0 |
| | | |
| | | |
| | | |

Partition 0

| Offset | Key | Value |
|---|---|---|
| **0** | **NY** | 20.0 |
| **1** | **NY** | 80.0 |
| | | |
| | | |

Partition 1



| Offset | Key | Value |
|---|---|---|
| **0** | **OH** | 100.0 |
| | | |
| | | |
| | | |

Partition 0

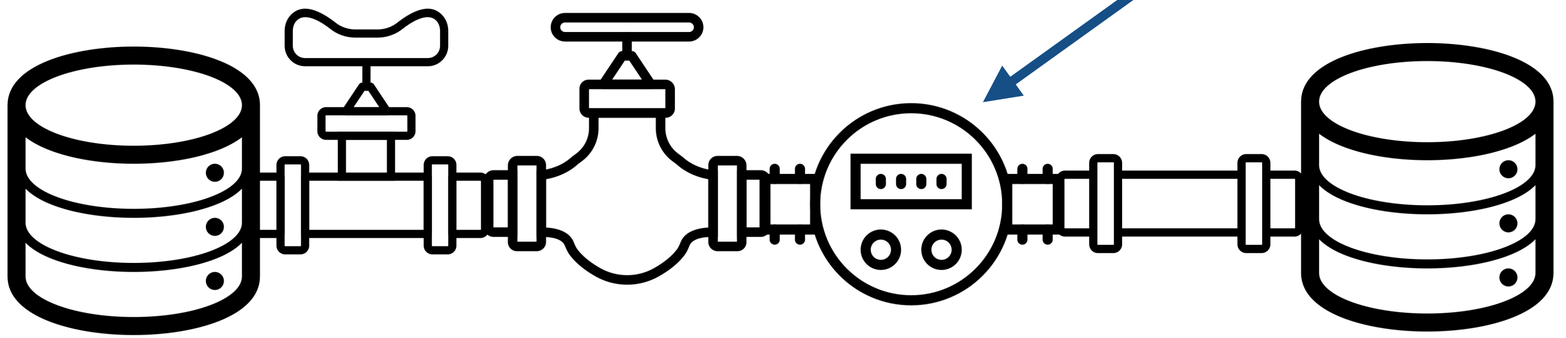
| Offset | Key | Value |
|---|---|---|
| **1** | **NY** | 80.0 |
| | | |
| | | |
| | | |

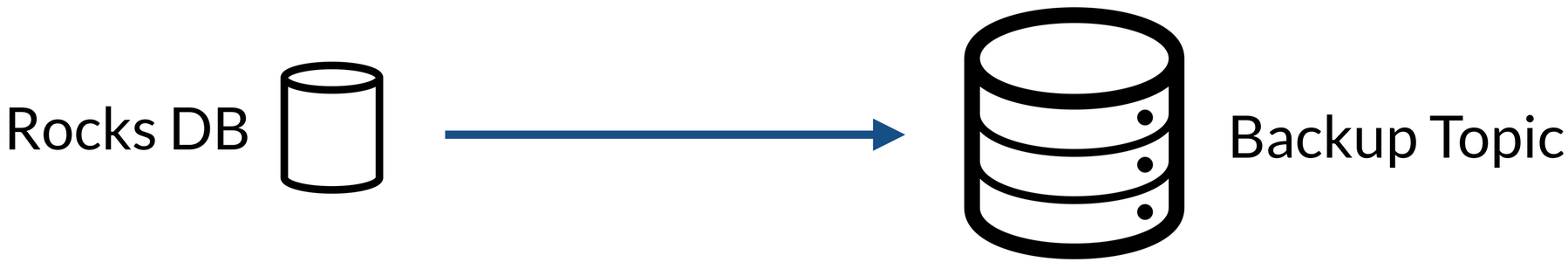
Partition 1

# After Compaction

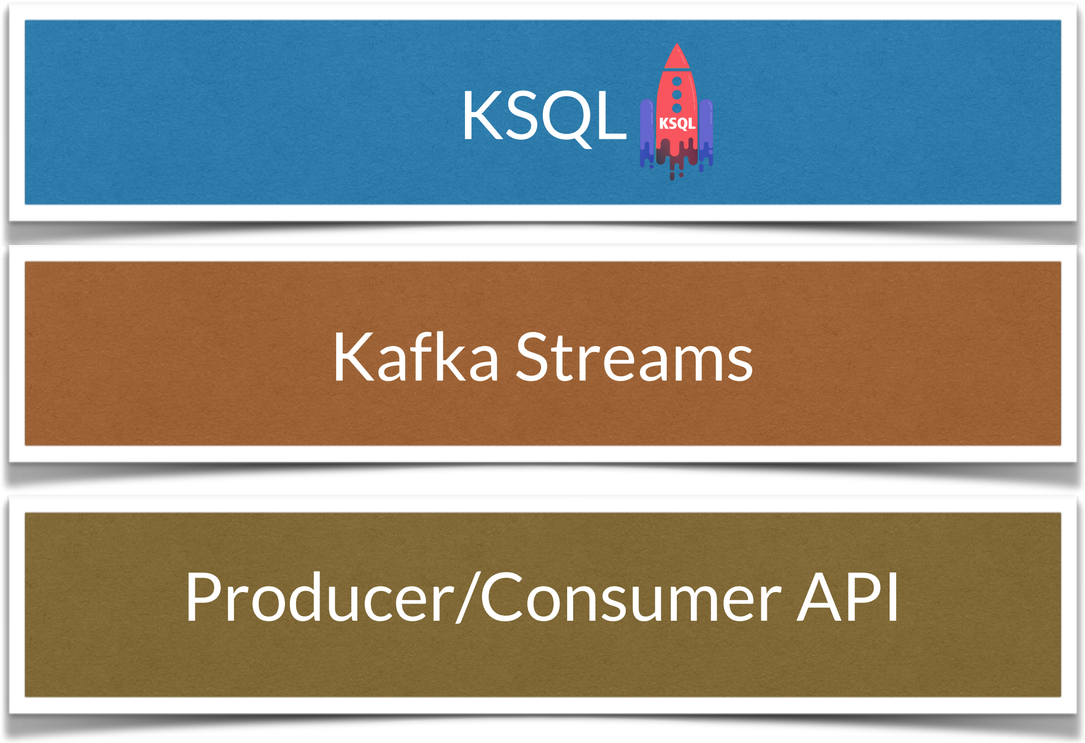
Rocks DB

| Key | Value |
|---|---|
| **OH** | 100.0 |
| **NY** | 80.0 |
| | |
| | |




Rocks DB
Backup Topic


KSQL


KSQL
KSQL
Kafka Streams
Producer/Consumer API

# Starting in CLI Mode

```
/bin/ksql http://localhost:8088
```

# Starting in Headless Mode

```
/bin/ksql-start-server \
server_path/etc/ksql/ksql-server.properties \
--queries-file /path/to/queries.sql
```

# Tapping a Stream Based Topic

```
CREATE STREAM pageviews \
  (viewtime BIGINT, \
   userid VARCHAR, \
   pageid VARCHAR) \
  WITH (KAFKA_TOPIC='pageviews', \
        VALUE_FORMAT='DELIMITED');
```

# Tapping a Table Based Topic

```
CREATE TABLE users \
  (registertime BIGINT, \
   userid VARCHAR, \
   gender VARCHAR, \
   regionid VARCHAR) \
  WITH (KAFKA_TOPIC = 'users', \
        VALUE_FORMAT='JSON', \
        KEY = 'userid');
```

# Non-Persistent Query

```
SELECT * FROM pageviews
  WHERE ROWTIME >= 1510923225000
    AND ROWTIME <= 1510923228000;
```

A `LIMIT` can be used to limit the number of rows returned.
Once the limit is reached the query will terminate.

# Persistent vs. Non-Persistent

| | Persistent | Non Persistent |
|---|---|---|
| Where does data go? | Will store into a topic | Will display on screen |
| How do I stop it? | Find query id using `SHOW QUERIES` and `TERMINATE <id>` | `CTRL + C` |
| How to create? | `CREATE STREAM AS SELECT ...` | `SELECT ....` |

# UDF (User-Defined Functions)

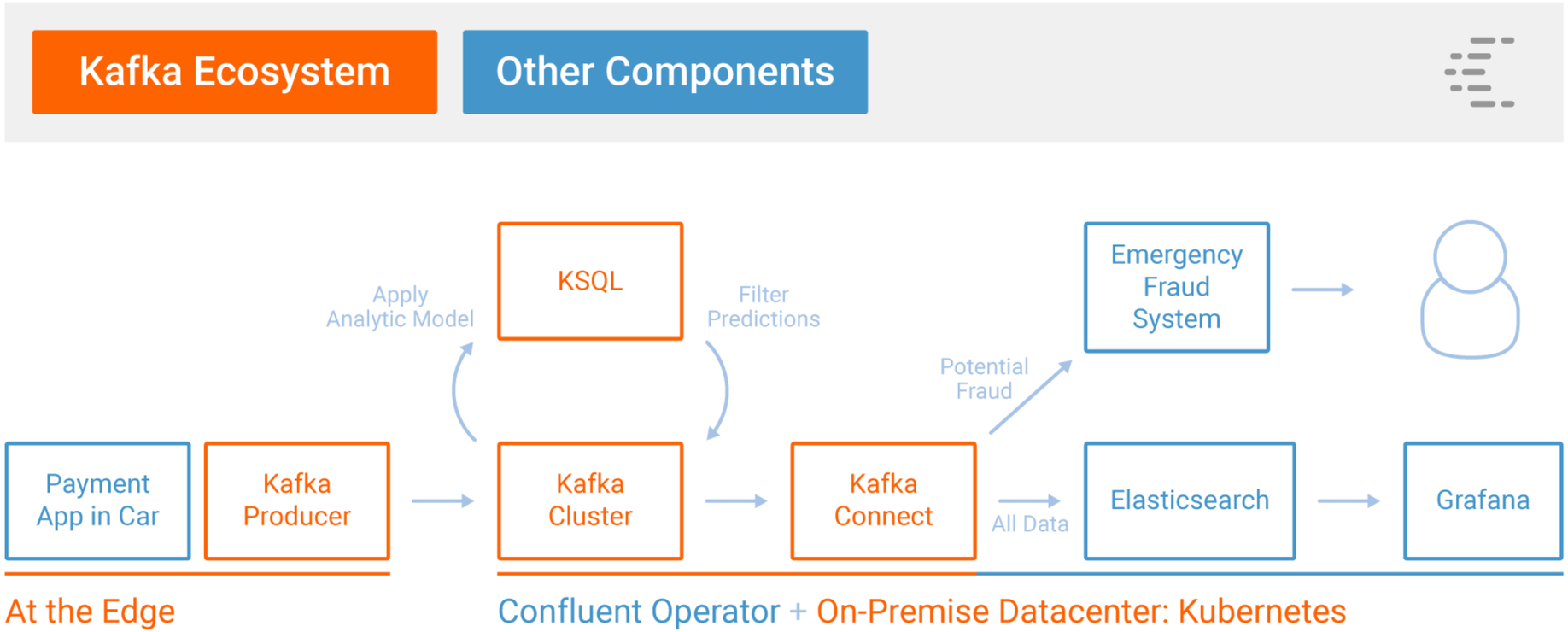
```
package com.example.ksql.functions;

import io.confluent.ksql.function.udf.Udf;
import io.confluent.ksql.function.udf.UdfDescription;
import io.confluent.ksql.function.udf.UdfParameter;

@UdfDescription(
    name = "reverse",
    description = "Example UDF that reverses an object",
    version = "0.1.0",
    author = ""
)
public class ReverseUdf {
  @Udf(description = "Reverse a string")
  public String reverseString(
      @UdfParameter(value = "source", description = "the value to reverse")
      final String source
  ) {
    return new StringBuilder(source).reverse().toString();
  }
}
```

# What if we use UDF for ML?

# The Tie-in




https://www.confluent.io/blog/build-udf-udaf-ksql-5-0/

# Creating a UDF Model with H2O

```
GenModel rawModel;

rawModel = (hex.genmodel.GenModel)
Class.forName(modelClassName).newInstance();

EasyPredictModelWrapper model = new
   EasyPredictModelWrapper(rawModel);
```

**H2O.ai**

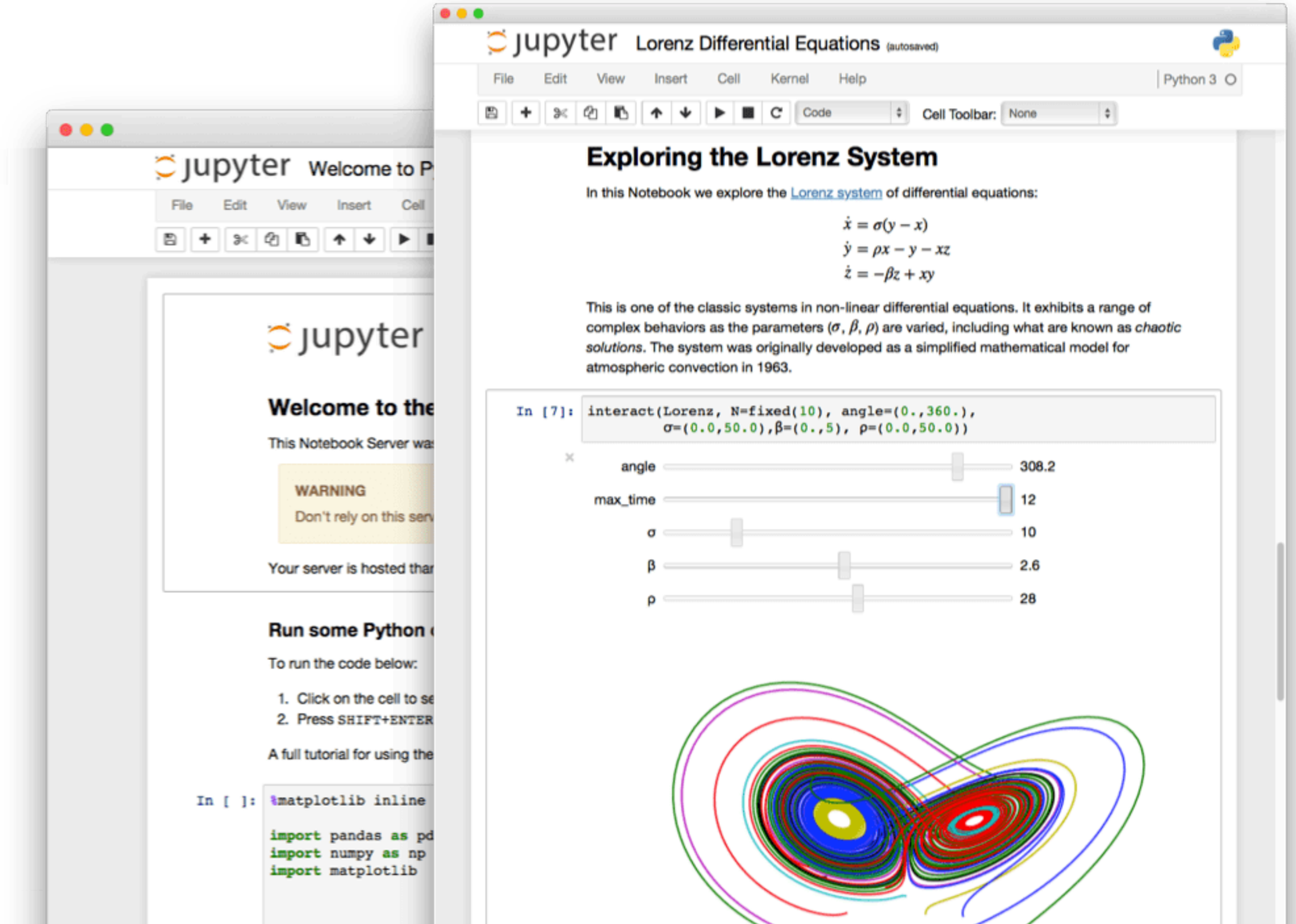
```
SELECT car_id, event_id, ANOMALY(sensor_input) FROM car_sensor;
```

**Source:** https://bit.ly/32zEgB0

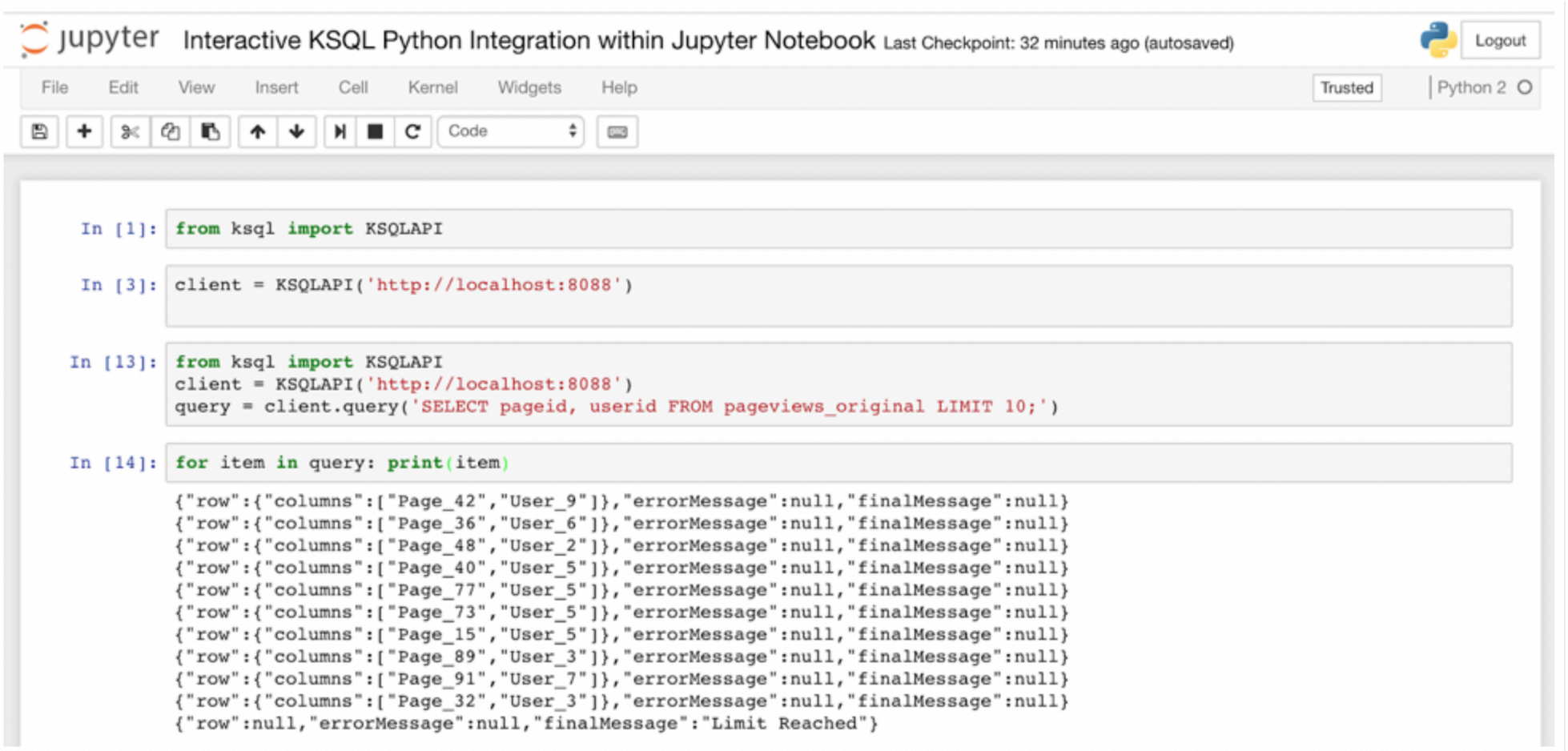

H2O.ai

# Jupyter Notebooks

# Popular use among Data-Scientists



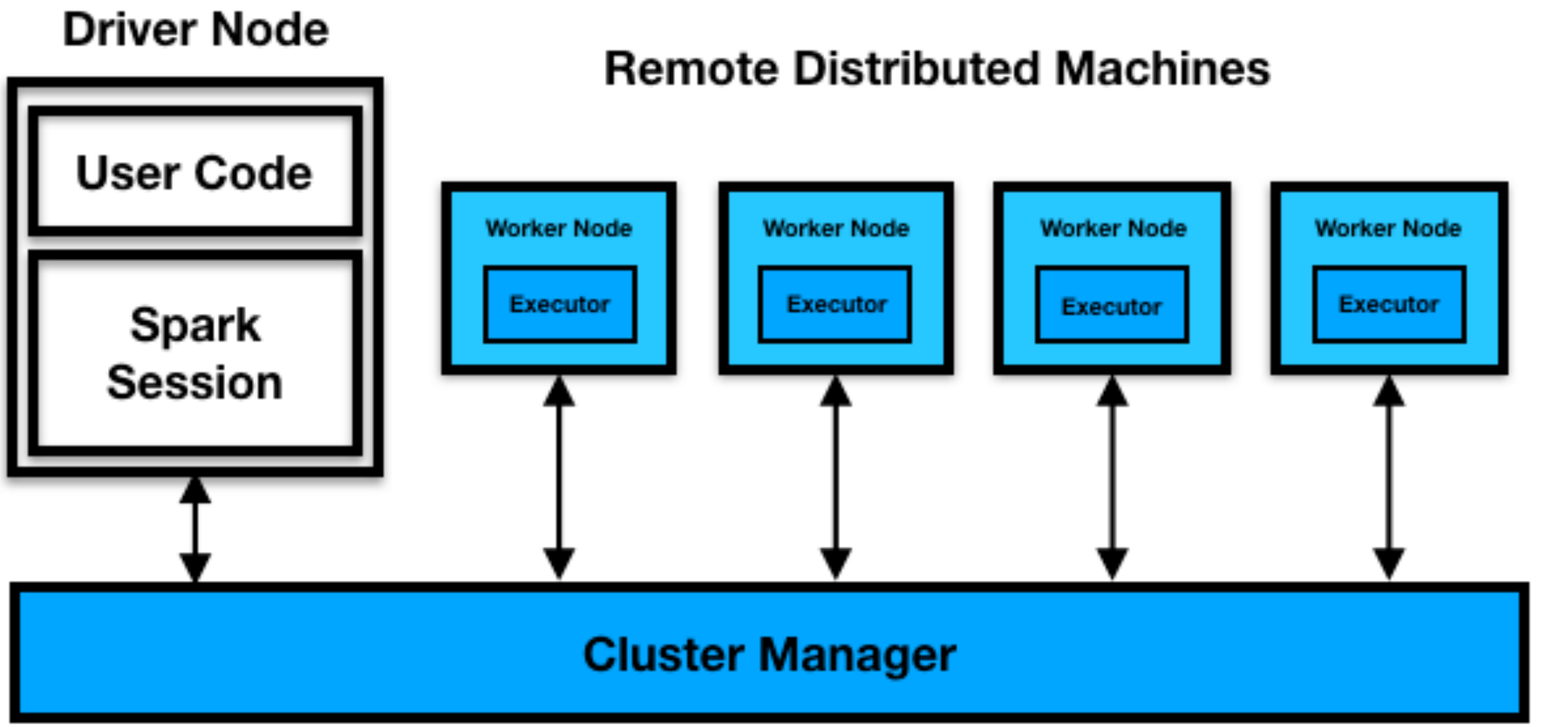
# Using KSQL-Python




**This enables live use of data through the notebook to experiment with possible modelling**

# Spark Streaming

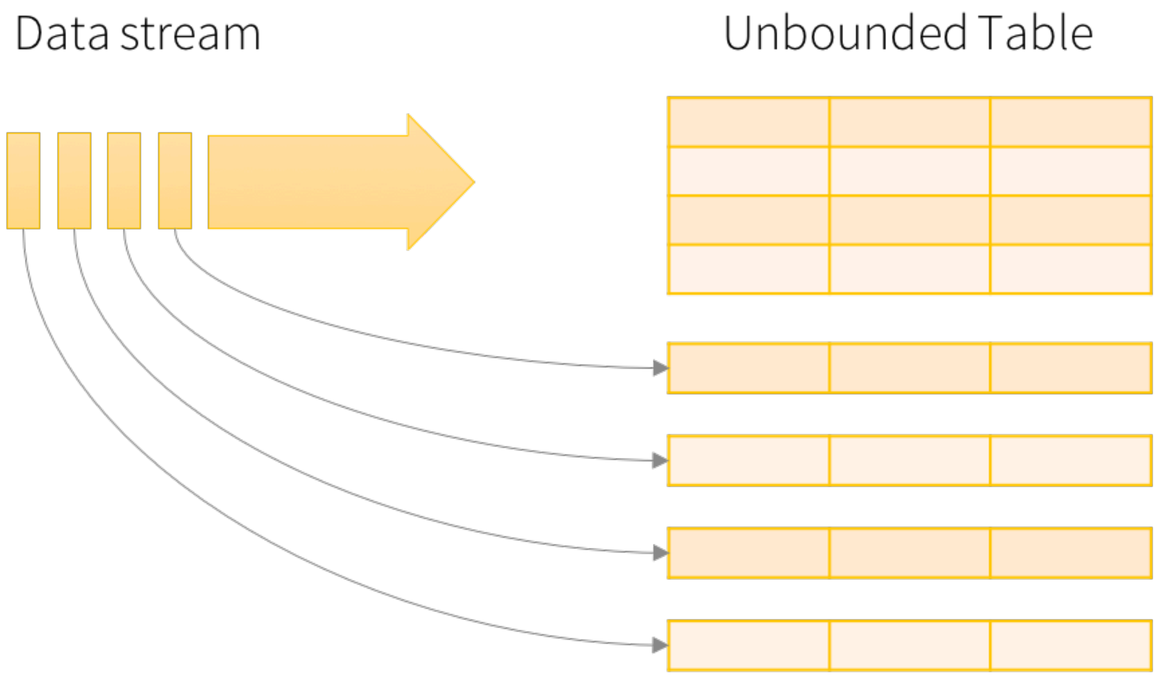

Java™


Driver Node
User Code
Spark Session
Remote Distributed Machines
Worker Node
Executor
Worker Node
Executor
Worker Node
Executor
Worker Node
Executor
Cluster Manager

# Two Types of Streaming APIs

- Unstructured Streaming API
  - Lower level API
  - Based on DStream
- Structured Streaming API
  - Higher level optimization
  - Micro-batch API
  - Streaming, but with DataFrame
  - Uses DataFrame, DataSet, and SparkSQL

# Structured Streaming




new data in the
data stream
=
new rows appended
to a unbounded table

Data stream as an unbounded table

# DataFrame

- Structured
- Are the most efficient due to catalyst optimizer
- Are available in all languages
- A table with data rows and columns
- Analogous to a spreadsheet or table
- **Distributed and spans over multiple machines!**
- Easiest to use, particularly for non-functional programmers

# DataSet

- Structured
- Rows optimized by the catalyst optimizer
- Fully functional programmable
  - map
  - filter
  - flatMap
- A `DataFrame` is actually a `DataSet[Row]`

# Setting up the Structured Stream

```
private lazy val sparkSession = SparkSession
  .builder()
  .config(sparkConf)
  .getOrCreate()

private lazy val stream: DataFrame = sparkSession
  .readStream
  .format("socket")
  .option("host", "localhost")
  .option("port", "10150")
  .load()
```

# Doing something with that Stream

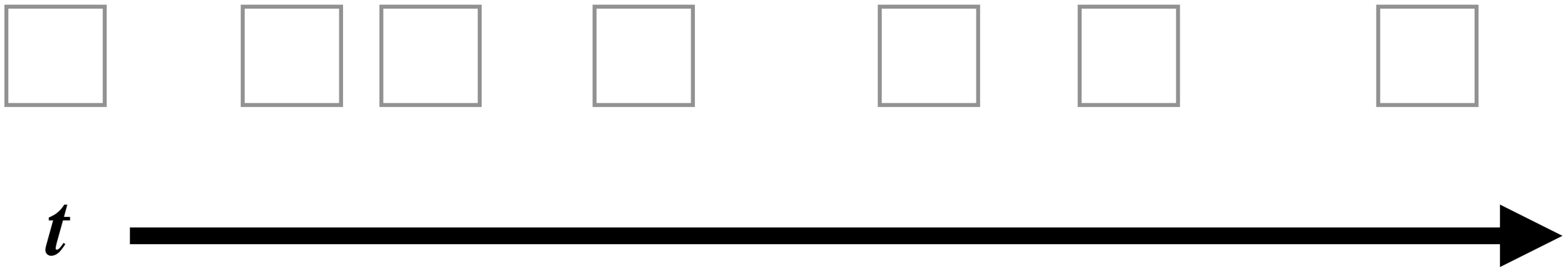
```
val words: Dataset[String] =
  stream.as[String].flatMap(x => x.split(" "))

val wordCounts = words.groupBy("value").count()
```

# Place the results somewhere

```
val query = wordCounts
  .writeStream
  .outputMode(OutputMode.Complete()) //Group By Requires Complete
  .format("console")
  .start()
```

# Unstructured Streaming



# Establishing a Unstructured Stream

```
val conf = new SparkConf().setMaster(master).setAppName("appname")
val ssc = new StreamingContext(conf, Seconds(1))
```

# Doing work with the DStream

```
val sourceDStream: ReceiverInputDStream[String] = streamingContext
  .socketTextStream("127.0.0.1", 10150)

val words: DStream[String] =
  sourceDStream.flatMap(s => s.split(" "))

val pairs: DStream[(String, Int)] =
  words.map(word => (word, 1))
```

# Closing an Unstructured Stream

```
ssc.start()                // Start the computation
ssc.awaitTermination()  // Wait for the computation to terminate
```

# Applying Spark Streaming ML

# Applying Spark ML

https://blog.clairvoyantsoft.com/machine-learning-with-spark-streaming-281b2d1e4fd5

# Conclusion

# Conclusion

What your DataScience team requires will determine where you will operationalize your ML

# Thank You

- Email: dhinojosa@evolutionnext.com
- Github: https://www.github.com/dhinojosa
- Twitter: http://twitter.com/dhinojosa
- Linked In: http://www.linkedin.com/in/dhevolutionnext