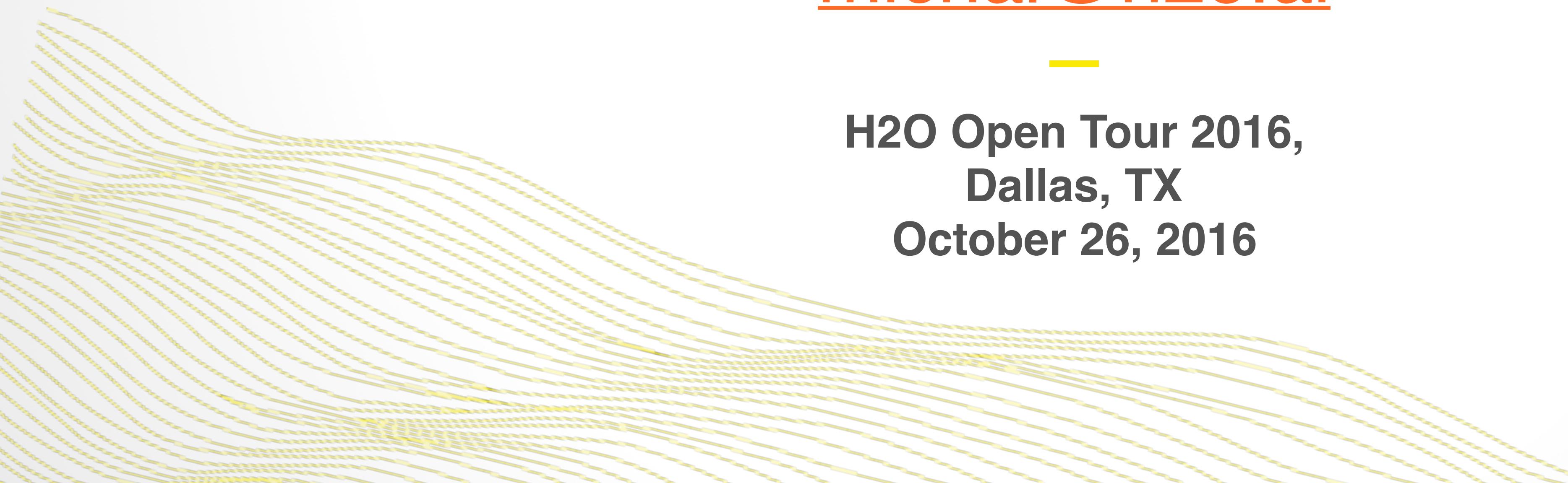


Sparkling Water 2.0

Michal Malohlava

michal@h2o.ai

—
H2O Open Tour 2016,
Dallas, TX
October 26, 2016



Spark + H₂O

**SPARKLING
WATER**

H₂O + Spark

=

**Sparkling
Water**

Sparkling Water provides

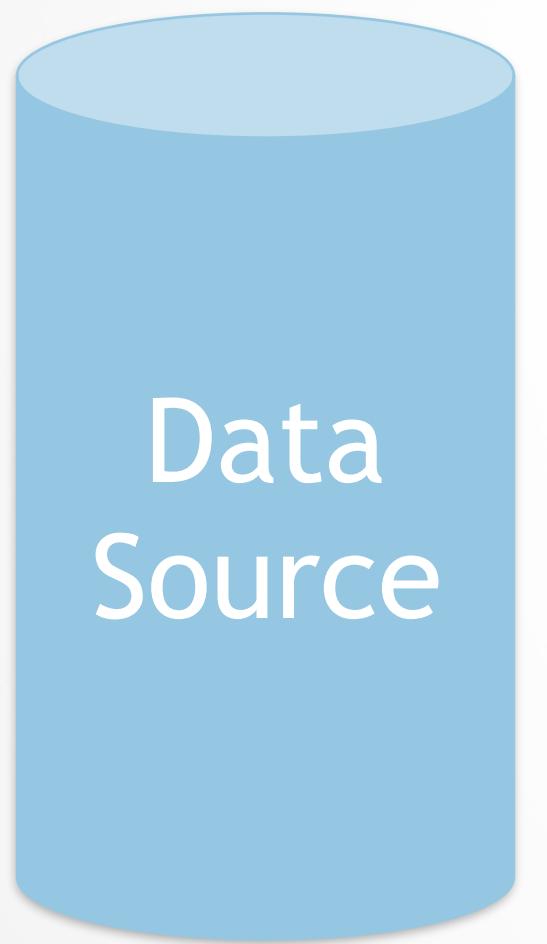
Transparent integration of H2O machine learning platform
with Spark ecosystem

Transparent use of **H2O data structures** (H2OFrame) and
algorithms with **Spark API**

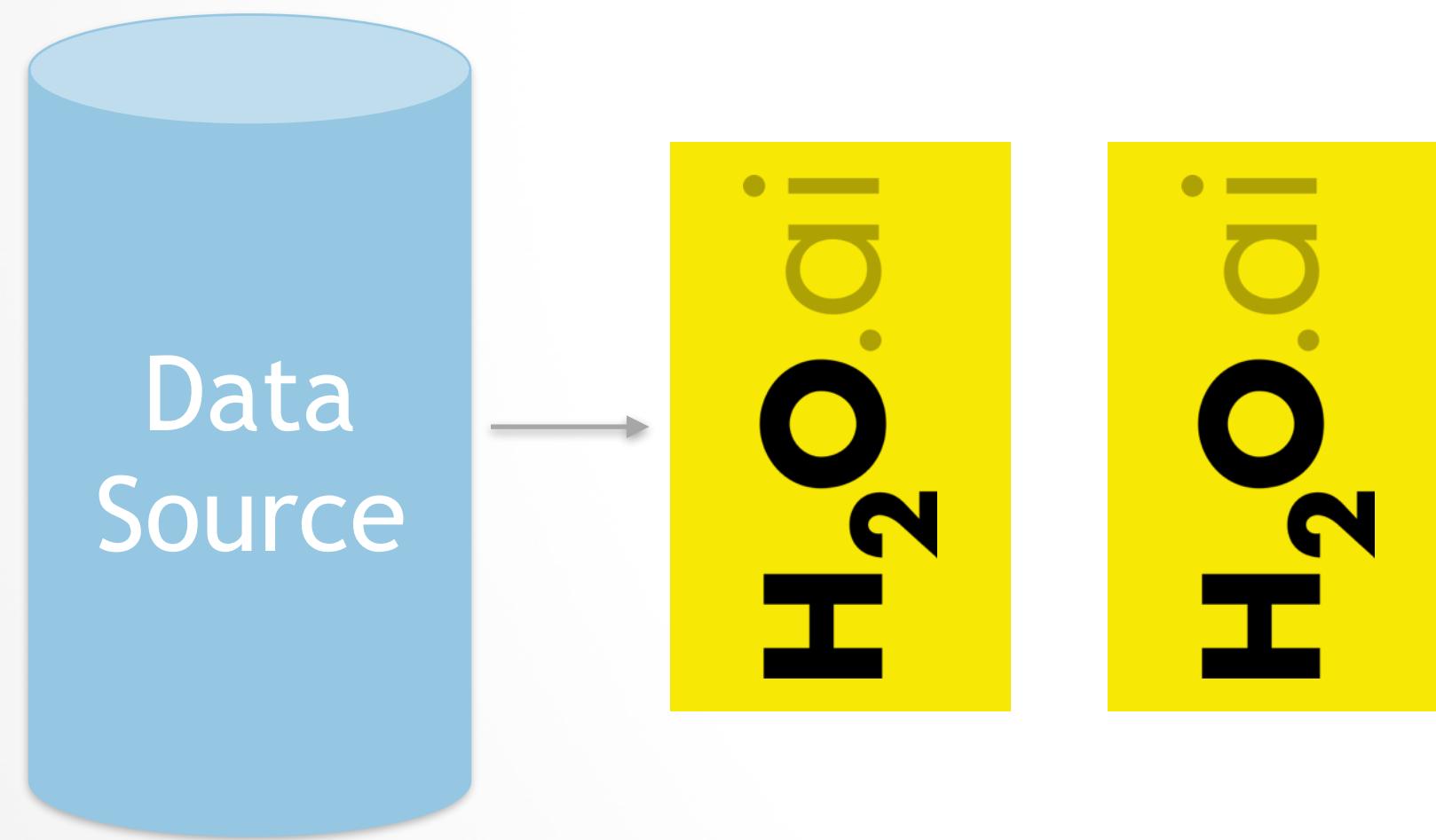
**Excels in existing Spark workflows requiring advanced
Machine Learning algorithms**

Functionality missing in H2O can be
replaced by Spark and vice versa

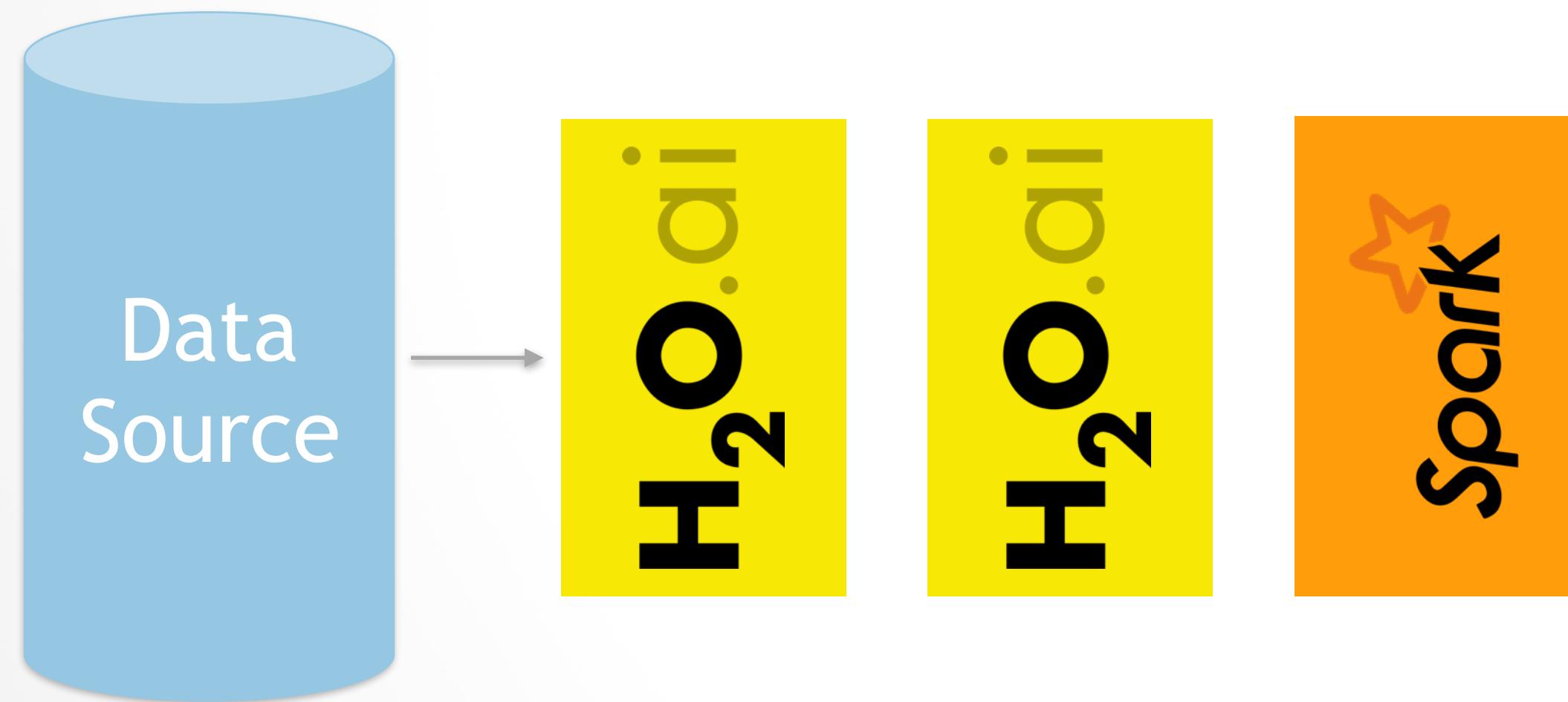
Data Munging & Modeling



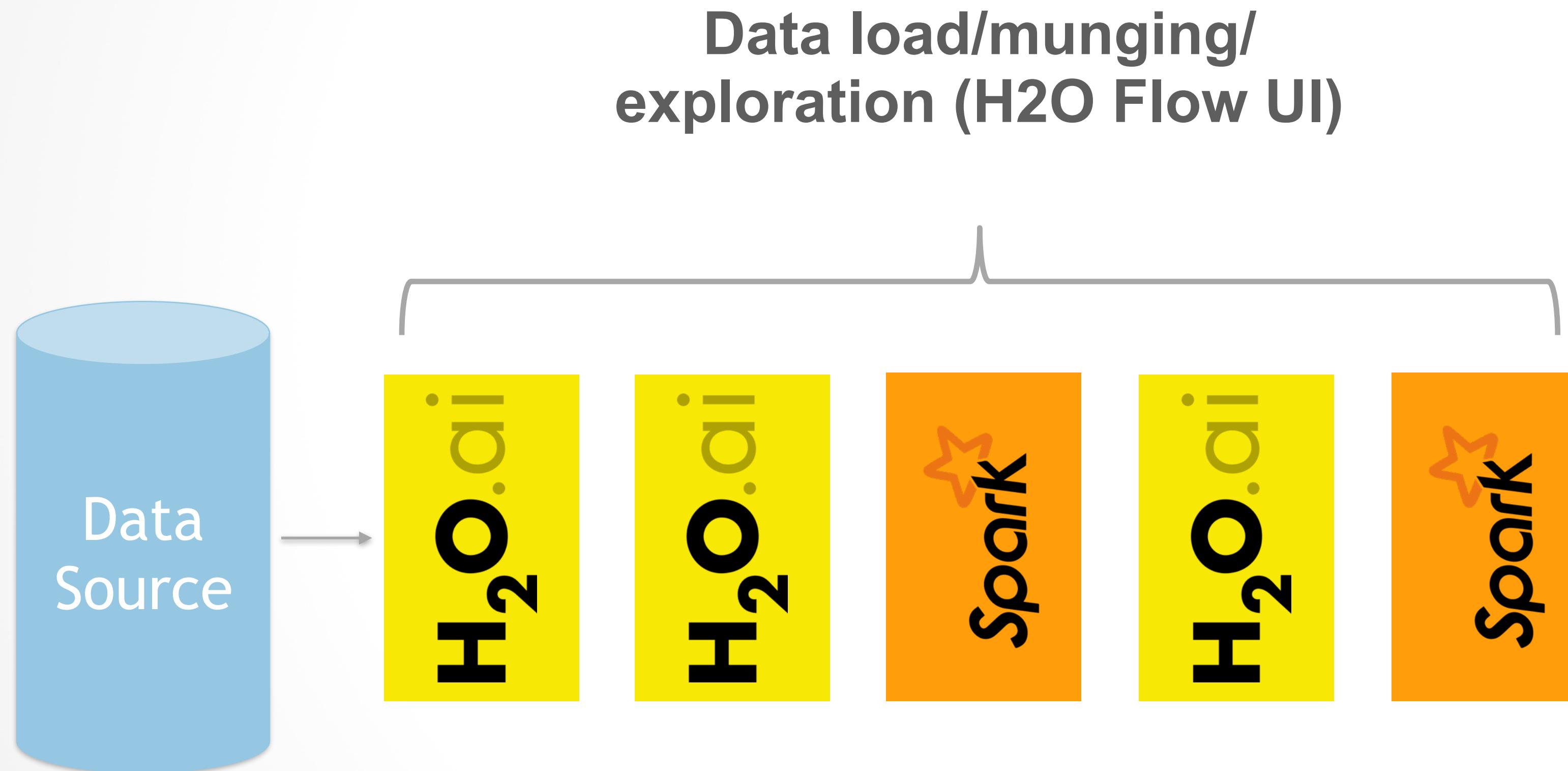
Data Munging & Modeling



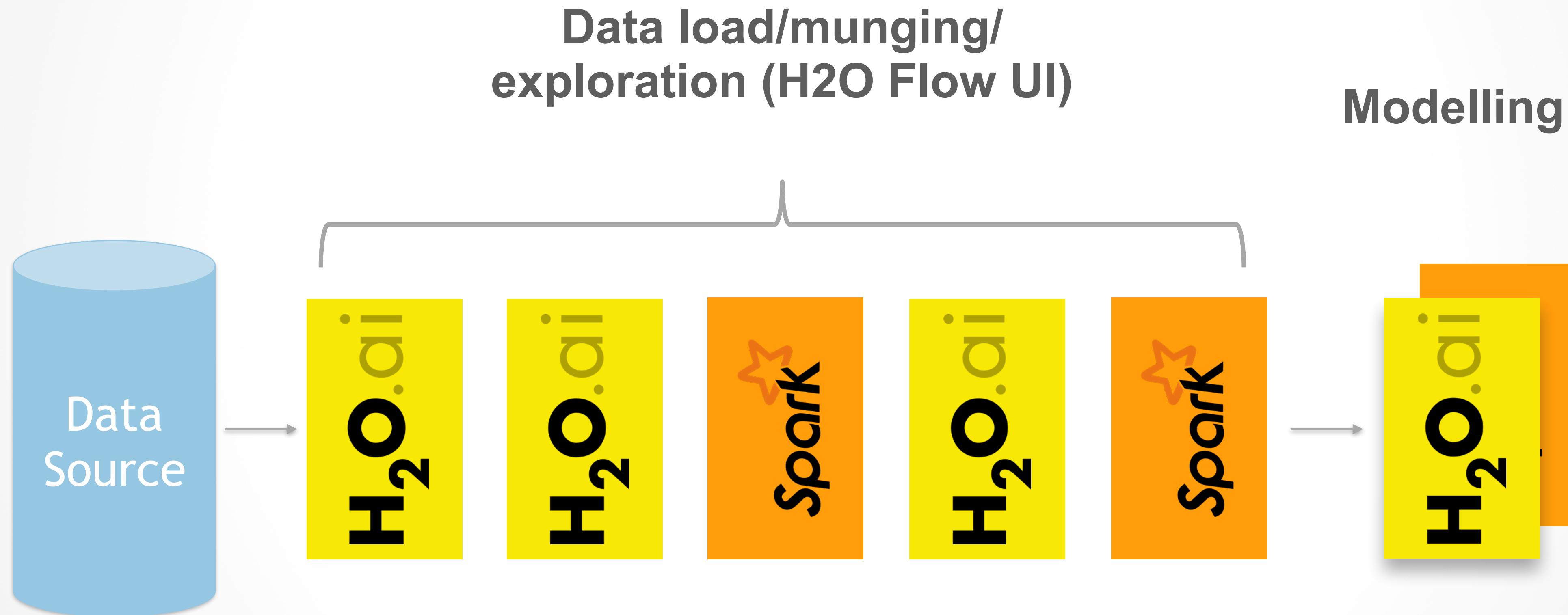
Data Munging & Modeling



Data Munging & Modeling



Data Munging & Modeling



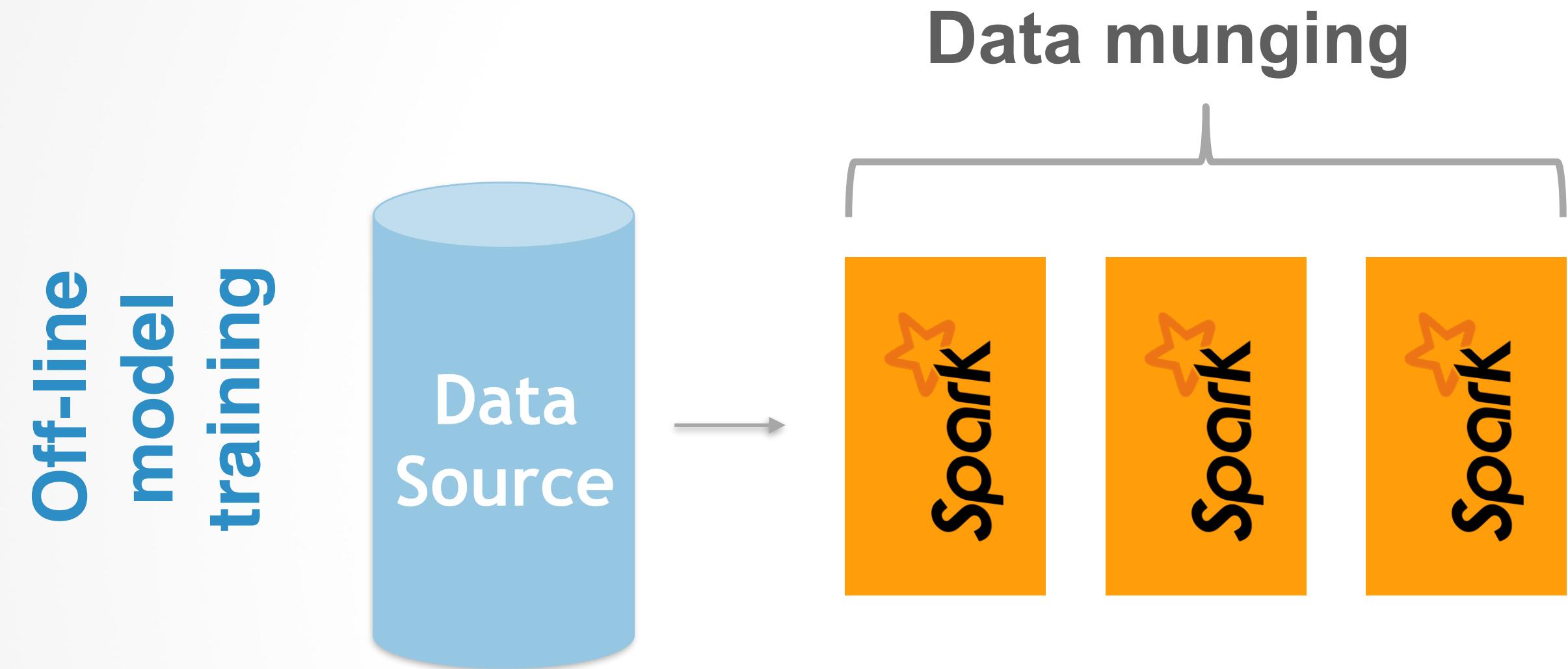
Stream processing

Stream processing

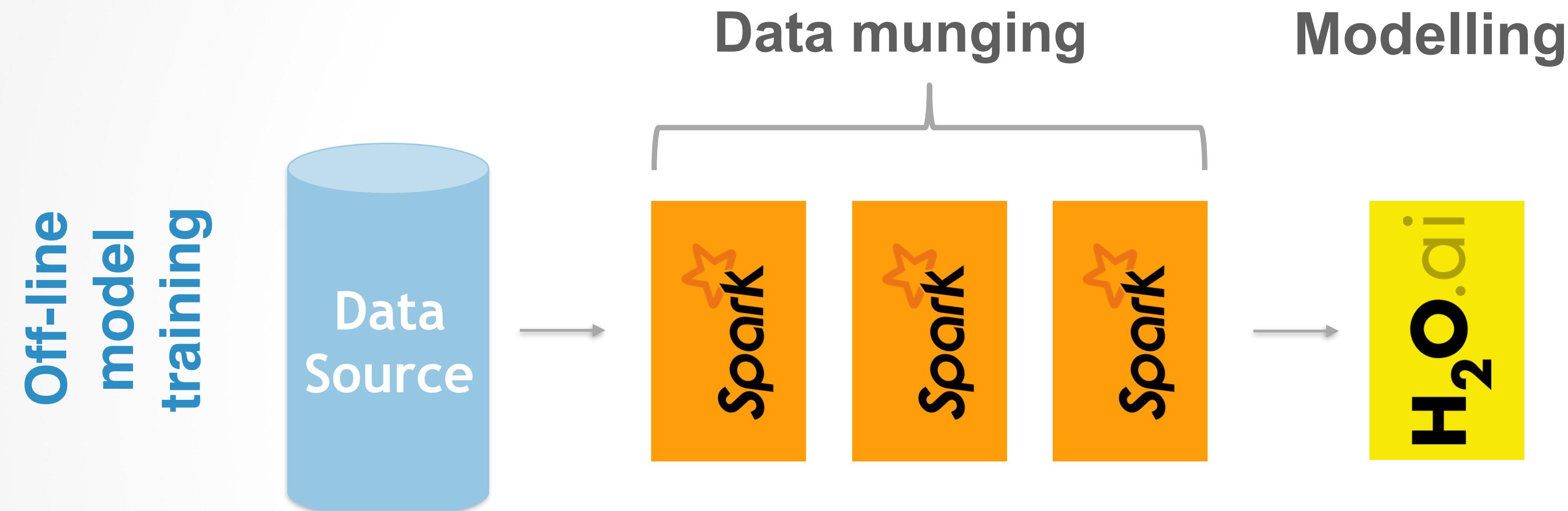
Off-line
model
training



Stream processing

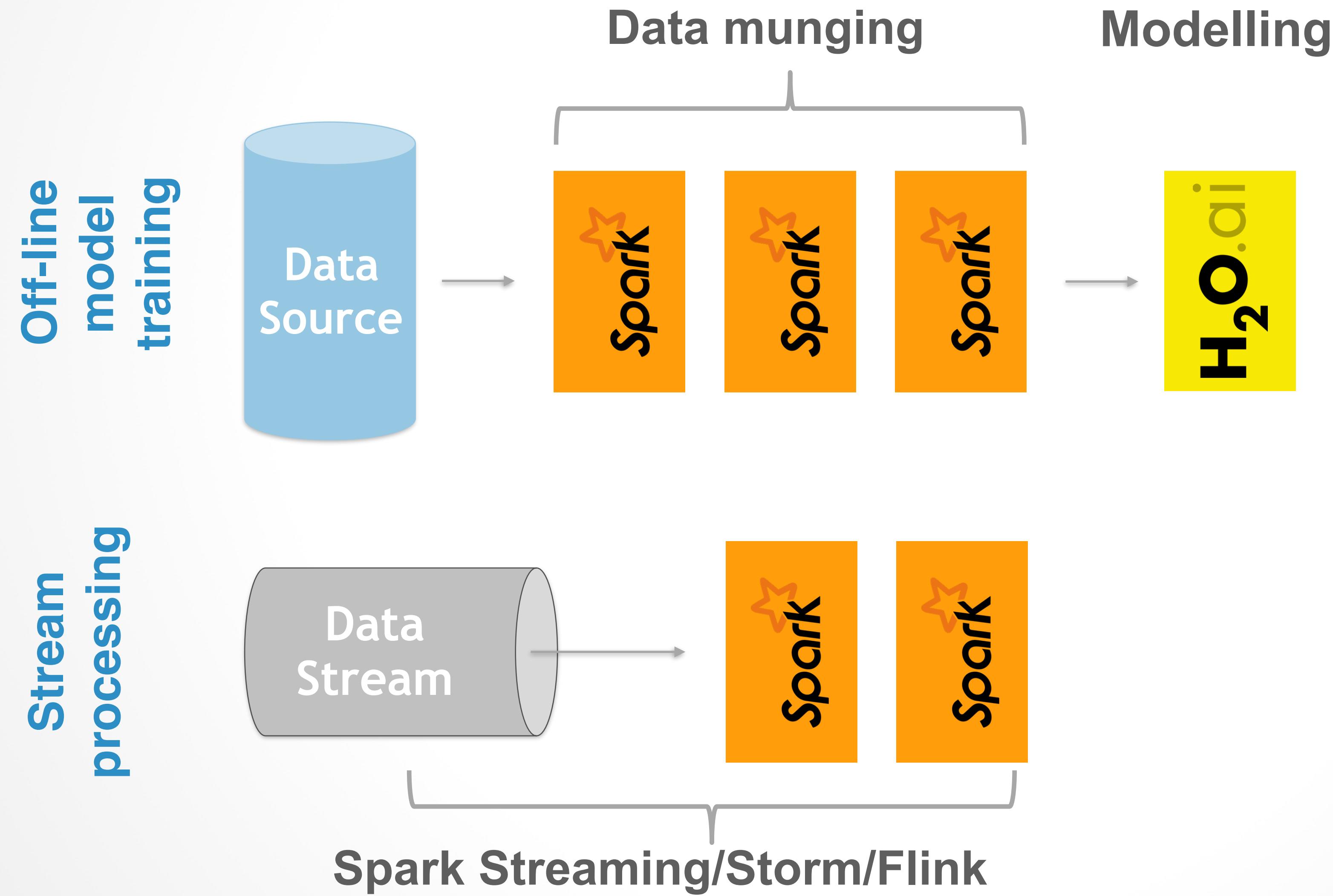


Stream processing

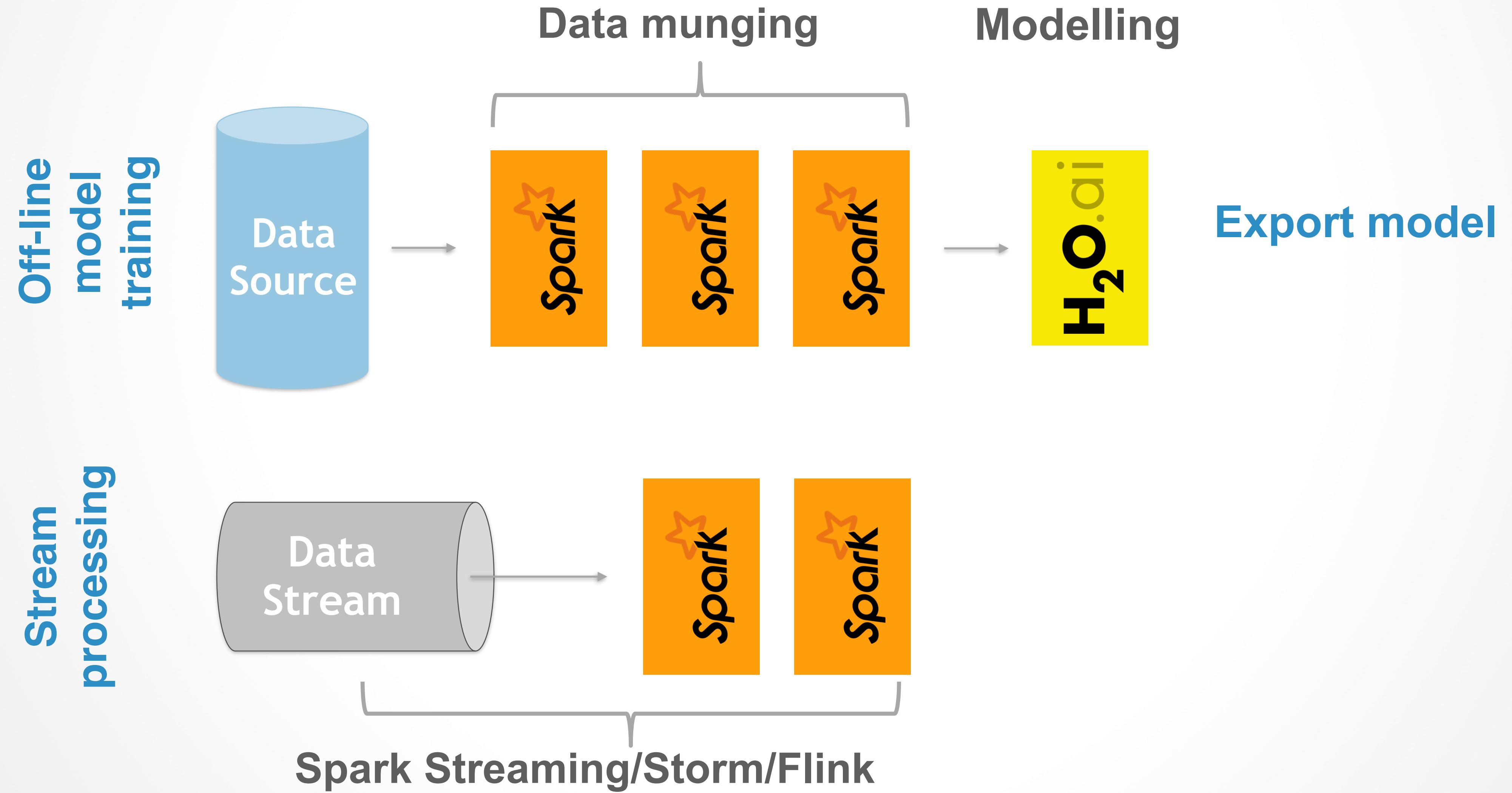


Off-line
model
training

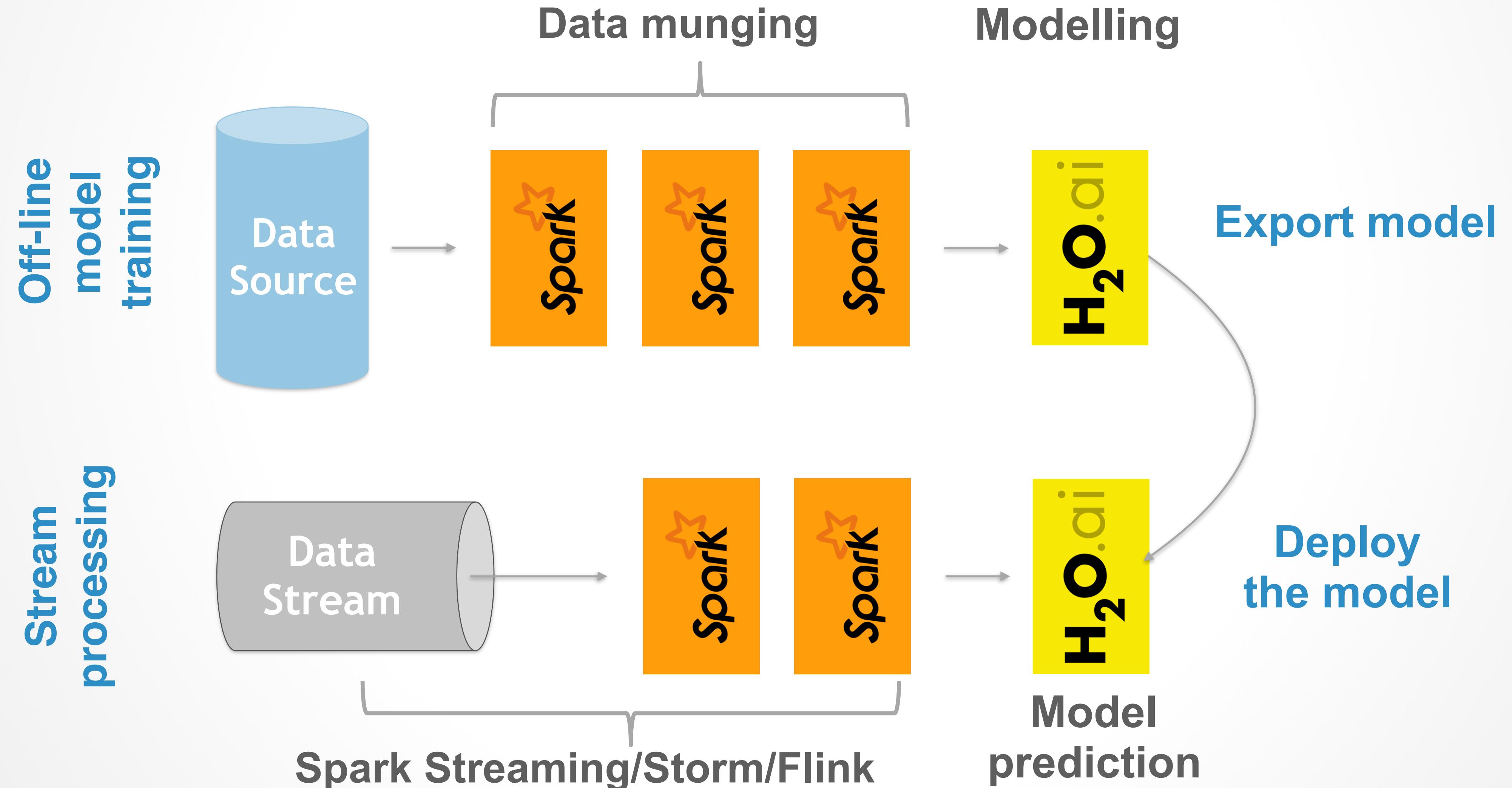
Stream processing



Stream processing



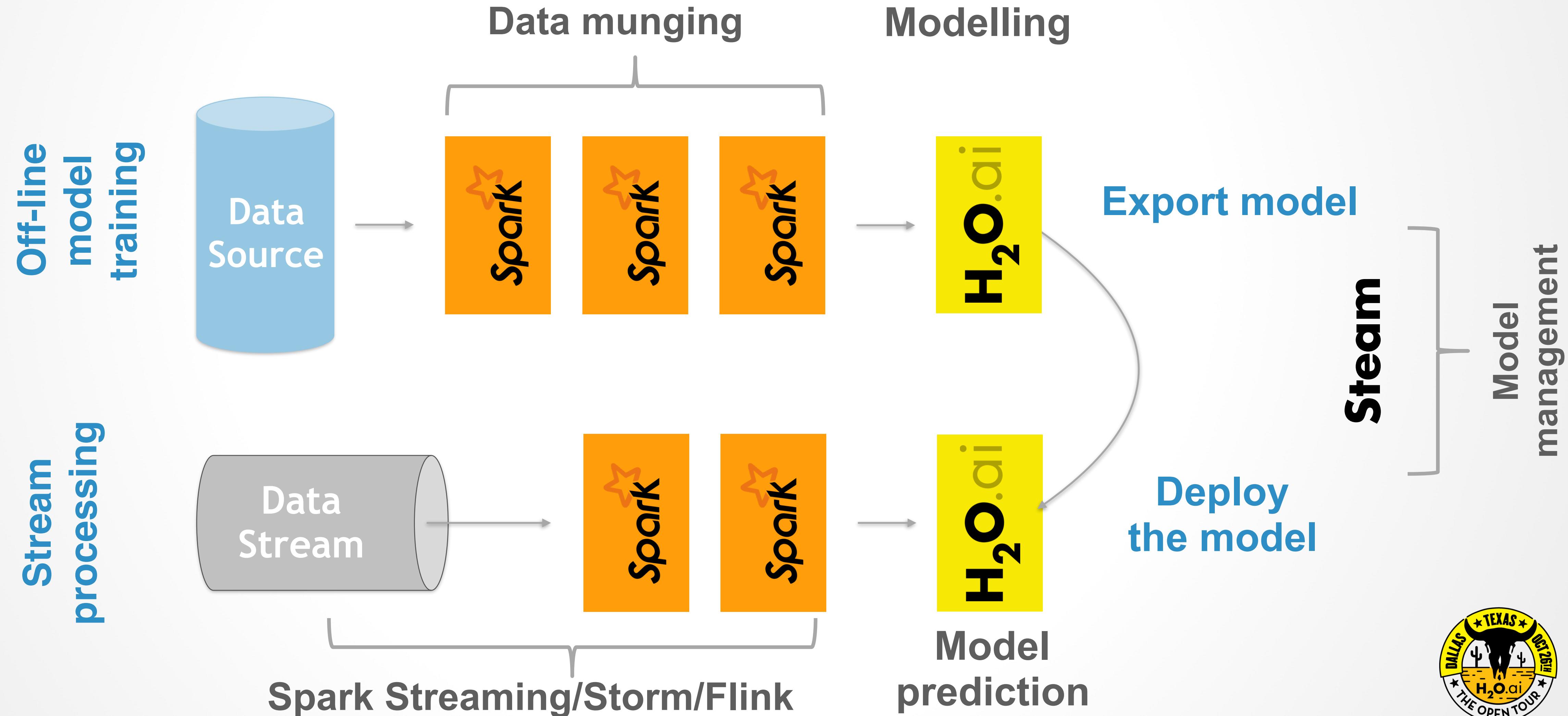
Stream processing



H₂O.ai



Stream processing



Interested in more details?

Track Two	Sparkling Water Workshop Michal Malohlava, Design & Visualization, H2O.ai	3:40 PM
This workshop will get you up and running with visualization and teach you how to use visual intelligence in your data products.		

**What's new
in 2.0 ?**

Releasing Schema



Spark 2.0 API Support

- ✓ Support of Spark **Session** based API
- ✓ More aggressive optimization during Spark DataFrame to H2OFrame transformation
- ✓ **Scala 2.11** support
 - useful if you are building Spark based applications

Datasets Support

✓ Support of new Spark data structure **DataSet**

- **DataSet: fusion of strongly typed RDD[X] and DataFrame query API**

```
case class SamplePerson(name: String, age: Int, email: String)

val dataSource = ...

val samplePeople: List[SamplePerson] = dataSource map SamplePerson.tupled
// Create Dataset
val df: Dataset[SamplePerson] = sqlContext.createDataset(samplePeople)

// Publish Dataset as H2OFrame
val hf: H2OFrame = h2oContext.asH2OFrame(df)
```



Scala Code from Browser

Available in Sparkling Water 1.6!

✓ We can now execute Scala code directly from H2O Flow UI!

The screenshot shows the H2O Flow UI interface. On the left, there's a vertical toolbar with icons for file operations like Open, Save, and Print. The main area is divided into several sections:

- Extract columns:** Contains the following Scala code:

```
1 val cols = Array("loan_status","loan_amnt","term","int_rate", "emp_length","home_ownership","annual_inc",
  "annual_inc_joint", "purpose","addr_state","dti", "dti_joint", "delinq_2yrs","earliest_cr_line",
  "revol_util","total_acc","verification_status", "verification_status_joint", "desc")
2
3 // Select columns
4 val loanDataHF = new H2OFrame("LoanStats3a.hex")(cols)
```
- Publish H2O Frame as Spark DataFrame:** Contains the following Scala code:

```
1 val h2oContext = H2OContext.getOrCreate(sc)
2
3 val loanDataDF = h2oContext.asDataFrame(loanDataHF)
4 loanDataDF.printSchema()
```
- Transformations:** Contains the following Scala code:

```
1 import org.apache.spark.sql.functions._
2   val toLoanStatus = udf[String, String](status => status.trim.toLowerCase() match {
3     case "fully paid" => "good"
4     case _ => "bad"
5   })
```
- From string '1.5%' to a float number:** Contains the following Scala code:

```
1 val toNumericRate = udf[Float, String](rate => f
```

On the right side, there's a sidebar titled "Flows" which lists various flow definitions with their creation times:

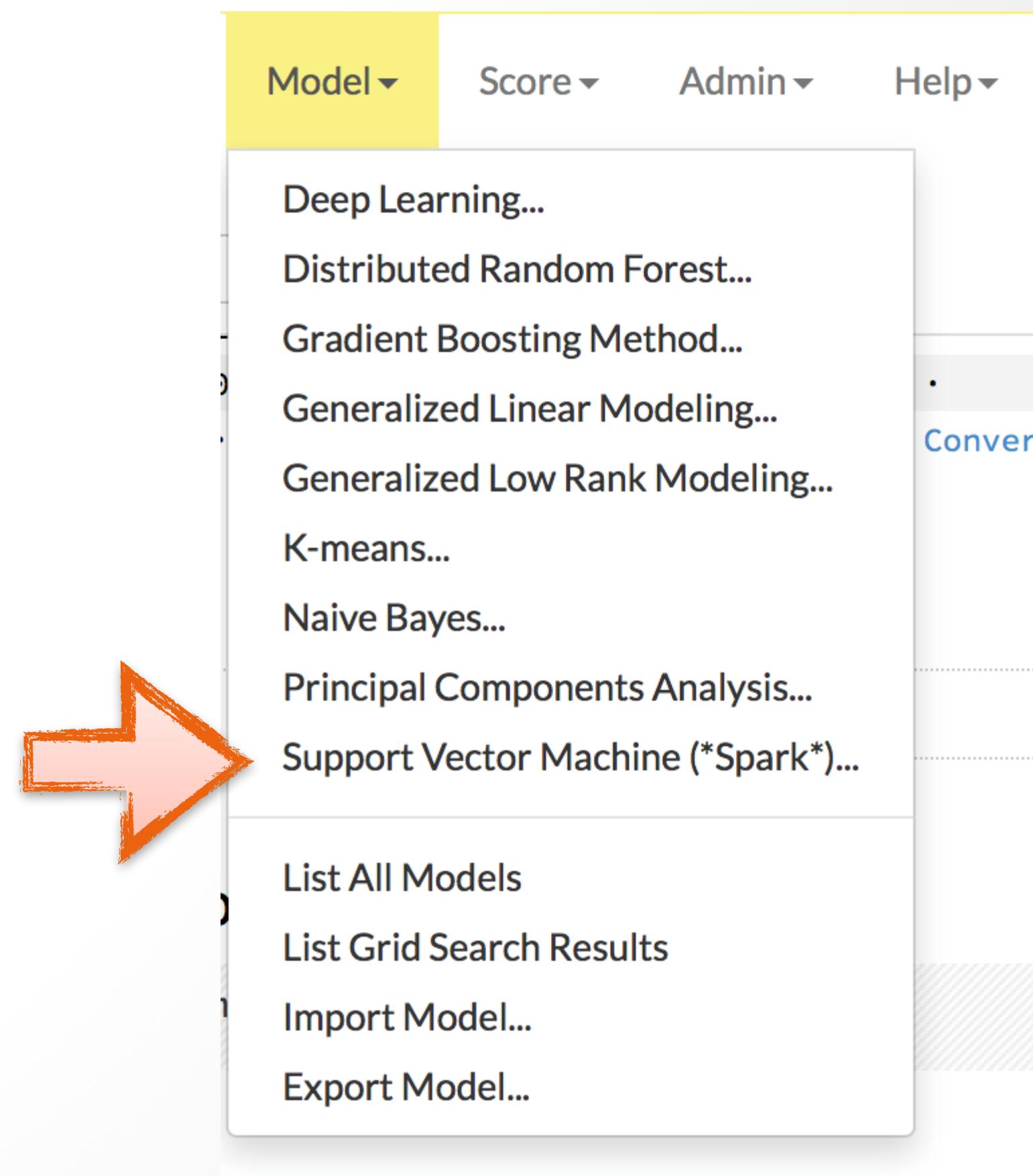
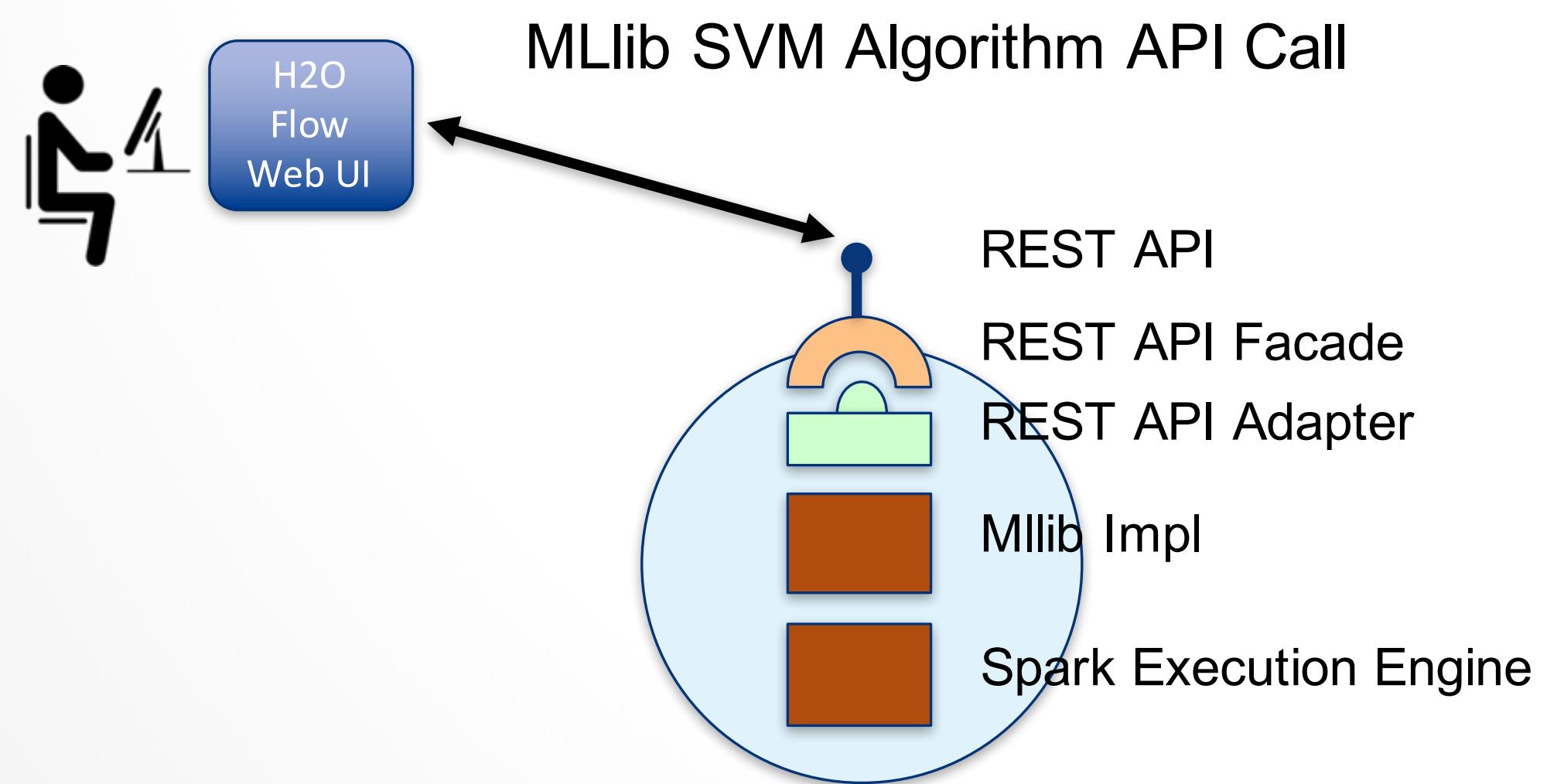
- H2O Tour Dallas - Flow n1 (a few seconds ago)
- Amazon Fine Food Sentiment Analysis (19 days ago)
- Strata NYC 2016 (a month ago)
- SVM with airlines (a month ago)
- Services (3 months ago)
- Fat frame (3 months ago)
- Huge GBM model (3 months ago)
- SVM with cars (3 months ago)
- LendingClub (5 months ago)
- Graph Tests (5 months ago)
- H2O Tour Chicago (6 months ago)
- HamOrSpam (6 months ago)
- Streaming (6 months ago)
- Parse AVRO (7 months ago)
- Deep learning MNIST

At the bottom right, there's a circular icon with a question mark and the text "Connections: 0".

Spark Models in Flow UI

Spark algorithms can be called from H2O Flow UI

- ✓ Quick visual feedback
- ✓ Model performance reports
- ✓ Model export (as code)



RSparkling

✓ A New Member of Sparkling Water Ecosystem

sparklyr

H₂O.ai



RSparkling

✓ A New Member of Sparkling Water Ecosystem

Scala: Sparkling Water

```
Spark  
val sc =  
  SparkContext.getOrCreate(...)  
  
val df = sc.parallelize(1 to 10).toDF
```

```
sparklyr  
val h2oContext =  
  H2OContext.getOrCreate(sc)  
  
val hf = h2oContext.asH2OFrame(df)
```

RSparkling

✓ A New Member of Sparkling Water Ecosystem

Scala: Sparkling Water

```
val sc =  
  SparkContext.getOrCreate(...)  
  
val df = sc.parallelize(1 to 10).toDF
```

```
val h2oContext =  
  H2OContext.getOrCreate(sc)  
  
val hf = h2oContext.asH2OFrame(df)
```

H₂O.ai

Python: PySparkling Water

```
sc = SparkContext(...)  
  
df = sc.parallelize(range(1,11))  
     .toDF("int")
```

```
h2o_context =  
  H2OContext.getOrCreate(sc)  
  
hf = h2o_context.as_h2o_frame(df)
```

sparklyr



RSparkling

✓ A New Member of Sparkling Water Ecosystem

Scala: Sparkling Water

```
val sc =  
  SparkContext.getOrCreate(...)  
  
val df = sc.parallelize(1 to 10).toDF
```

```
val h2oContext =  
  H2OContext.getOrCreate(sc)  
  
val hf = h2oContext.asH2OFrame(df)
```

Python: PySparkling Water

```
sc = SparkContext(...)  
  
df = sc.parallelize(range(1,11))  
     .toDF("int")
```

```
h2o_context =  
  H2OContext.getOrCreate(sc)  
  
hf = h2o_context.as_h2o_frame(df)
```

R: RSparkling Water

```
sc <- spark_connect(...)  
  
tbl <- data_frame(c(1:10))  
df <- copy_to(sc, tbl)
```

```
hc <- h2o_context(sc)  
  
hf <- as_h2o_frame(sc, df)
```

Jeff will showcase the sparklyr the new R package to interface with Spark and talk about the different use extensions including the rsparkling ML package.

✓ A New Member of Sparkling Water Ecosystem

Scala: Sparkling Water

```
val sc =  
  SparkContext.getOrCreate(...)  
  
val df = sc.parallelize(1 to 10).toDF
```

```
val h2oContext =  
  H2OContext.getOrCreate(sc)  
  
val hf = h2oContext.asH2OFrame(df)
```

Python: PySparkling Water

```
sc = SparkContext(...)  
  
df = sc.parallelize(range(1,11))  
     .toDF("int")
```

```
h2o_context =  
  H2OContext.getOrCreate(sc)  
  
hf = h2o_context.as_h2o_frame(df)
```

R: RSparkling Water

```
sc <- spark_connect(...)  
  
tbl <- data_frame(c(1:10))  
df <- copy_to(sc, tbl)
```

```
hc <- h2o_context(sc)  
  
hf <- as_h2o_frame(sc, df)
```

Available in Sparkling Water 1.6!



Coming
soon...

Bringing both platforms together

API Convergence

- H2O model builders compatibility with Spark pipelines
 - Scala/Python level
 - Reuse existing Spark functionality
 - e.g., applying Spark UDFs directly on H2OFrame
 - Heterogenous (Spark/H2O) model ensembles/pipelines
 - More Spark Algorithms in UI
 - e.g. ALS

Coming
soon!

More Tooling From H2O Ecosystem

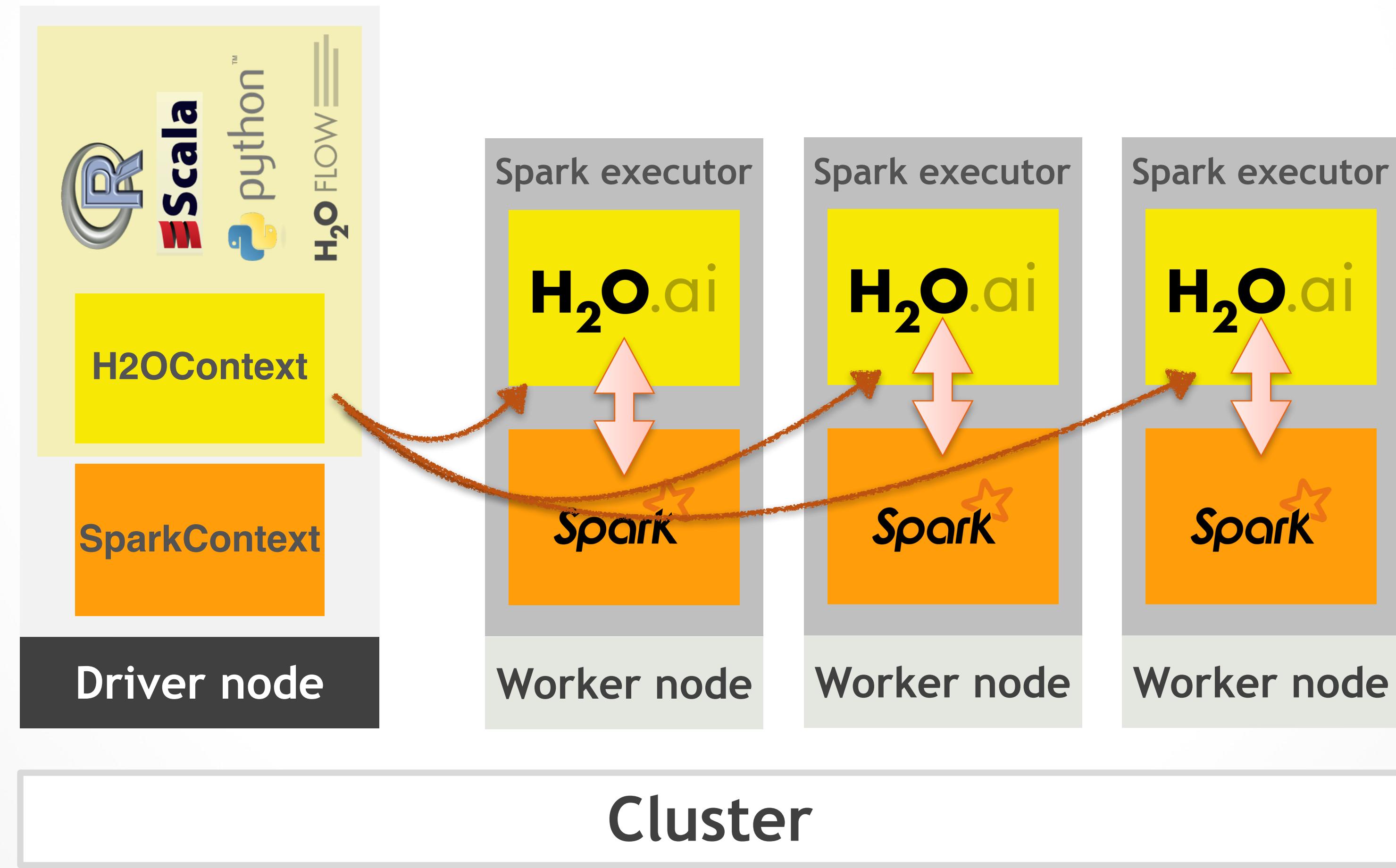
- Exposing DeepWater Algorithms
- Integration with Steam
 - Steam as Sparkling Water launcher
 - Steam as model manager

Coming
soon!

High Availability

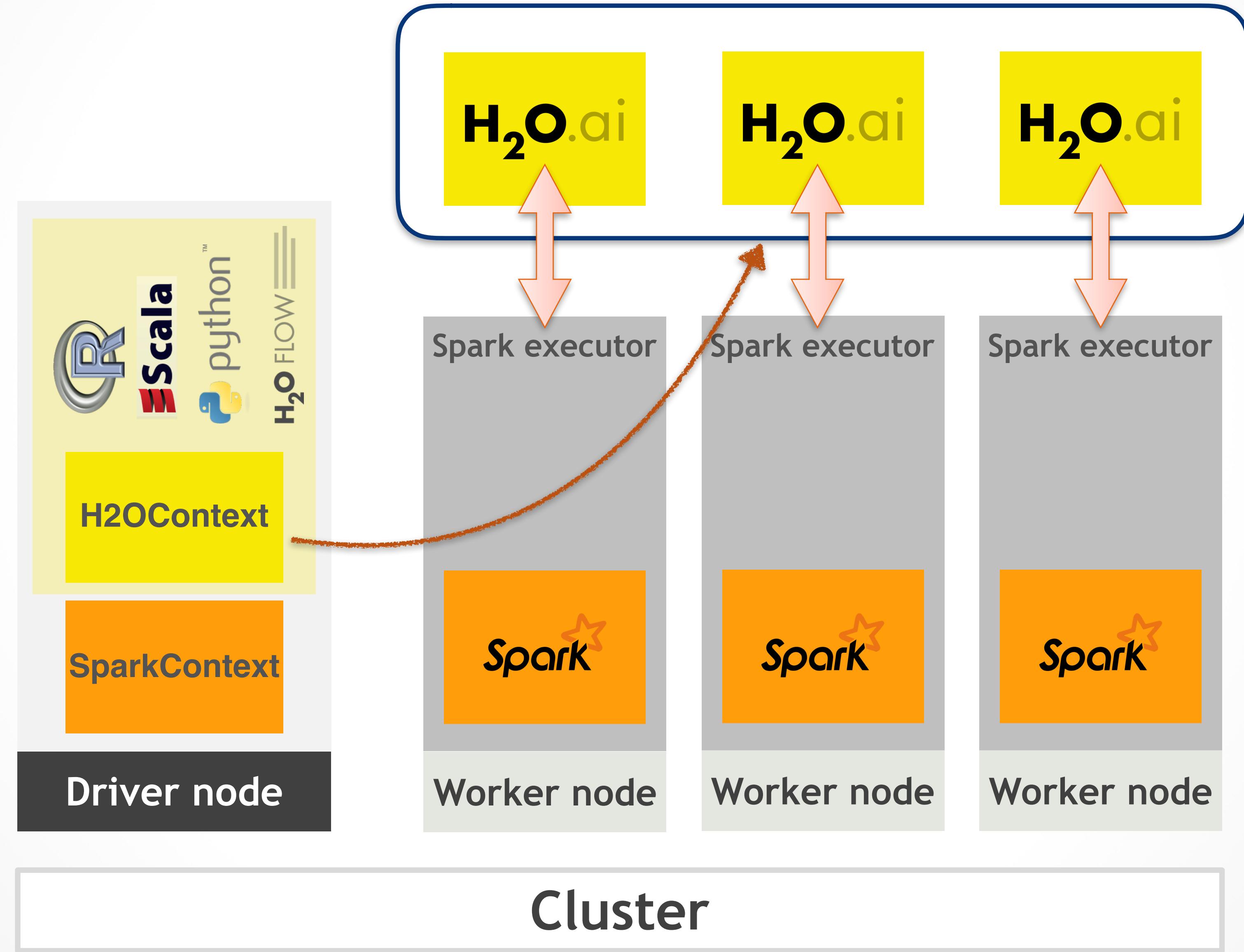
NOW

H₂O.ai



High Availability

Coming soon!



Thank you!

Sparkling Water is
open-source
ML application platform
combining
power of Spark and H2O



Learn more at h2o.ai
Follow us at [@h2oai](https://twitter.com/h2oai)