

Jakub Háva
jakub@h2o.ai

Sparkling Water 2.0

Machine Learning with H2O, Paris
September 21, 2016

Spark[★] + **H₂O**

**SPARKLING
WATER**

Who am I

- Finishing high-performance cluster monitoring tool for JVM based languages
- Finishing Master's at Charles Uni in Prague
- Software engineer at H2O.ai
- Tea lover (doesn't mean I don't like beer!)

Distributed Sparkling Team

- Michal - Mt. View, CA
- Kuba - Prague, CZ
- Mateusz - Tokyo, JP
- Vlad - Mt. View, CA

**H₂O+Spark =
Sparkling
Water**

Sparkling Water

- Transparent integration of H2O with Spark ecosystem - MLlib and H2O side-by-side
- Transparent use of H2O data structures and algorithms with Spark API
- Platform for building Smarter Applications
- Excels in existing Spark workflows requiring advanced Machine Learning algorithms

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

Benefits



- Additional algorithms
 - NLP
- Powerful data munging
- ML Pipelines



- Advanced algorithms
 - speed v. accuracy
- advanced parameters
- Fully distributed and parallelised
- Graphical environment
- R/Python interface

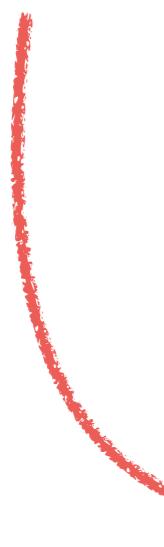
How to use Sparkling Water?

Start spark with Sparkling Water

`start.sh`

```
1 $SPARK_HOME/bin/spark-submit \
2   --class water.SparklingWaterDriver \
3   --packages ai.h2o:sparkling-water-examples_2.10:1.6.3 \
4   --executor-memory=6g \
5   --driver-class-path scalastyle.jar /dev/null
```

Raw

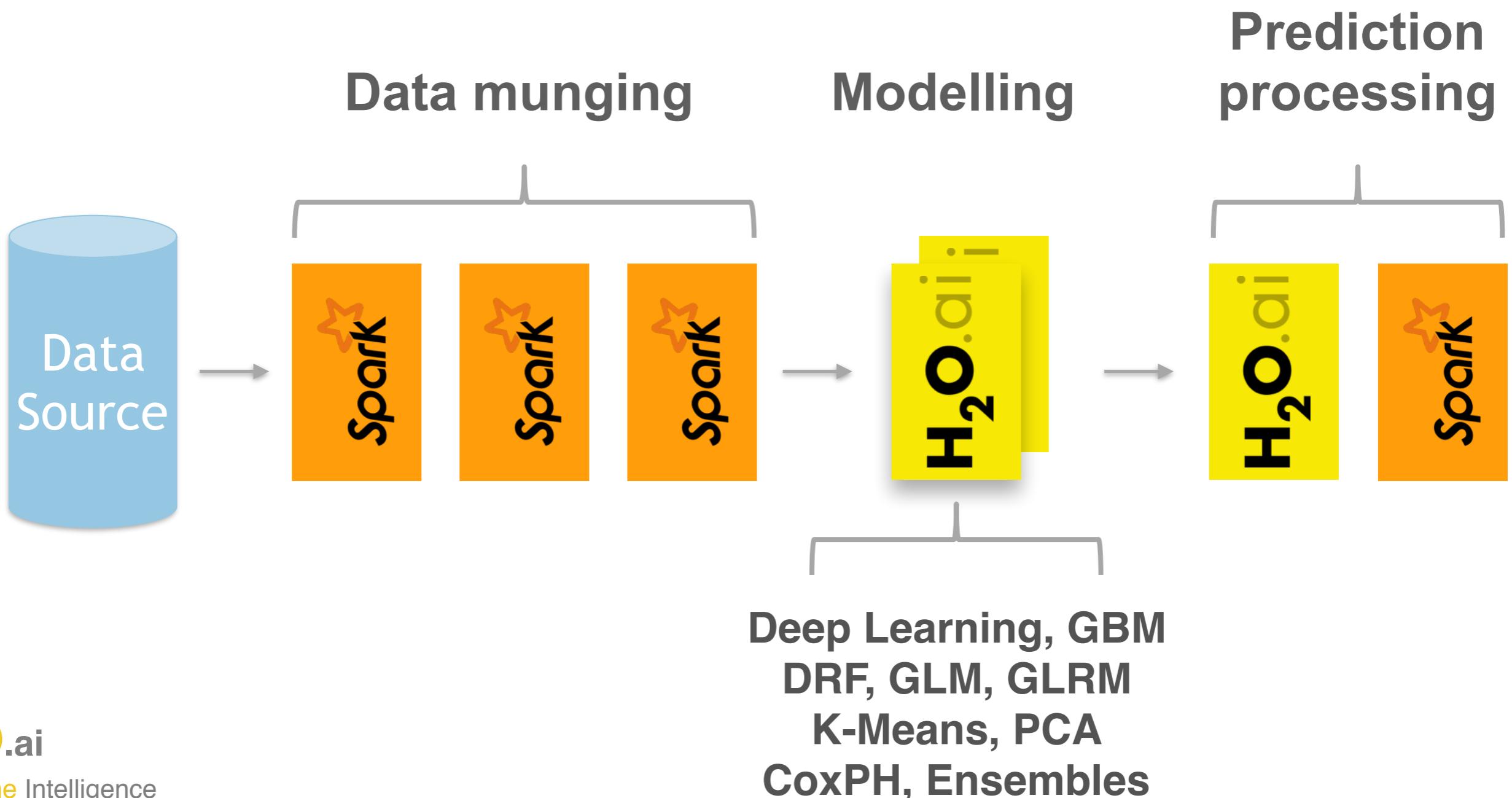


The screenshot shows the H2O Flow web application running at `127.0.0.1`. The title bar says "Start Spark with Sparkling Water". The main area is titled "Untitled Flow" and contains a toolbar with various icons. Below the toolbar is a search bar with the placeholder "assist". To the left of the search bar is a sidebar with the title "Assistance" containing a list of H2O routines:

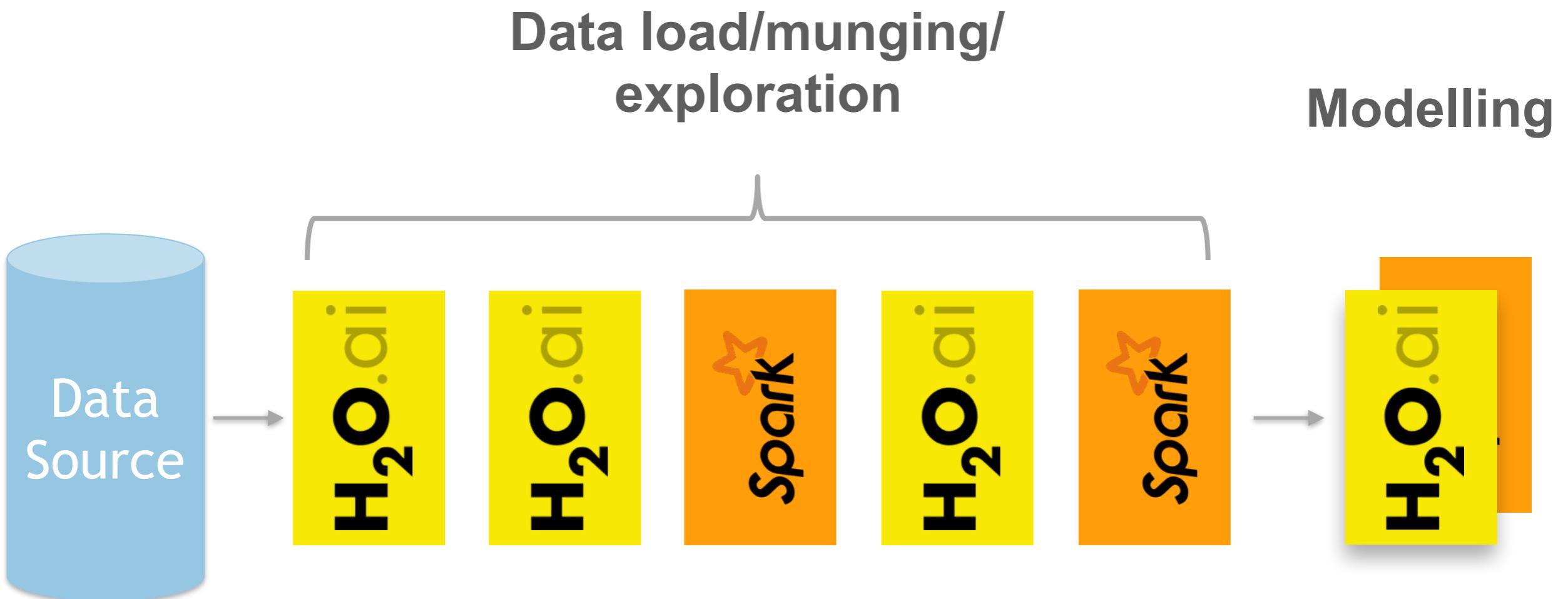
Routine	Description
<code>importFiles</code>	Import file(s) into H2O
<code>getFrames</code>	Get a list of frames in H2O
<code>splitFrame</code>	Split a frame into two or more frames
<code>getModels</code>	Get a list of models in H2O
<code>getGrids</code>	Get a list of grid search results in H2O
<code>getPredictions</code>	Get a list of predictions in H2O
<code>getJobs</code>	Get a list of jobs running in H2O
<code>buildModel</code>	Build a model
<code>importModel</code>	Import a saved model
<code>predict</code>	Make a prediction
<code>getRDDs</code>	Get a list of Spark's RDDs
<code>getDataFrames</code>	Get a list of Spark's data frames

To the right of the assistance sidebar is a "HELP" tab which is currently selected. It contains sections for "Using Flow for the first time?", "Or, view example Flows to explore and learn H2O.", and "GENERAL" and "EXAMPLES" sections. A footer at the bottom of the page says "Flow packs are a great way to explore and learn H2O. Try out these flows and run them".

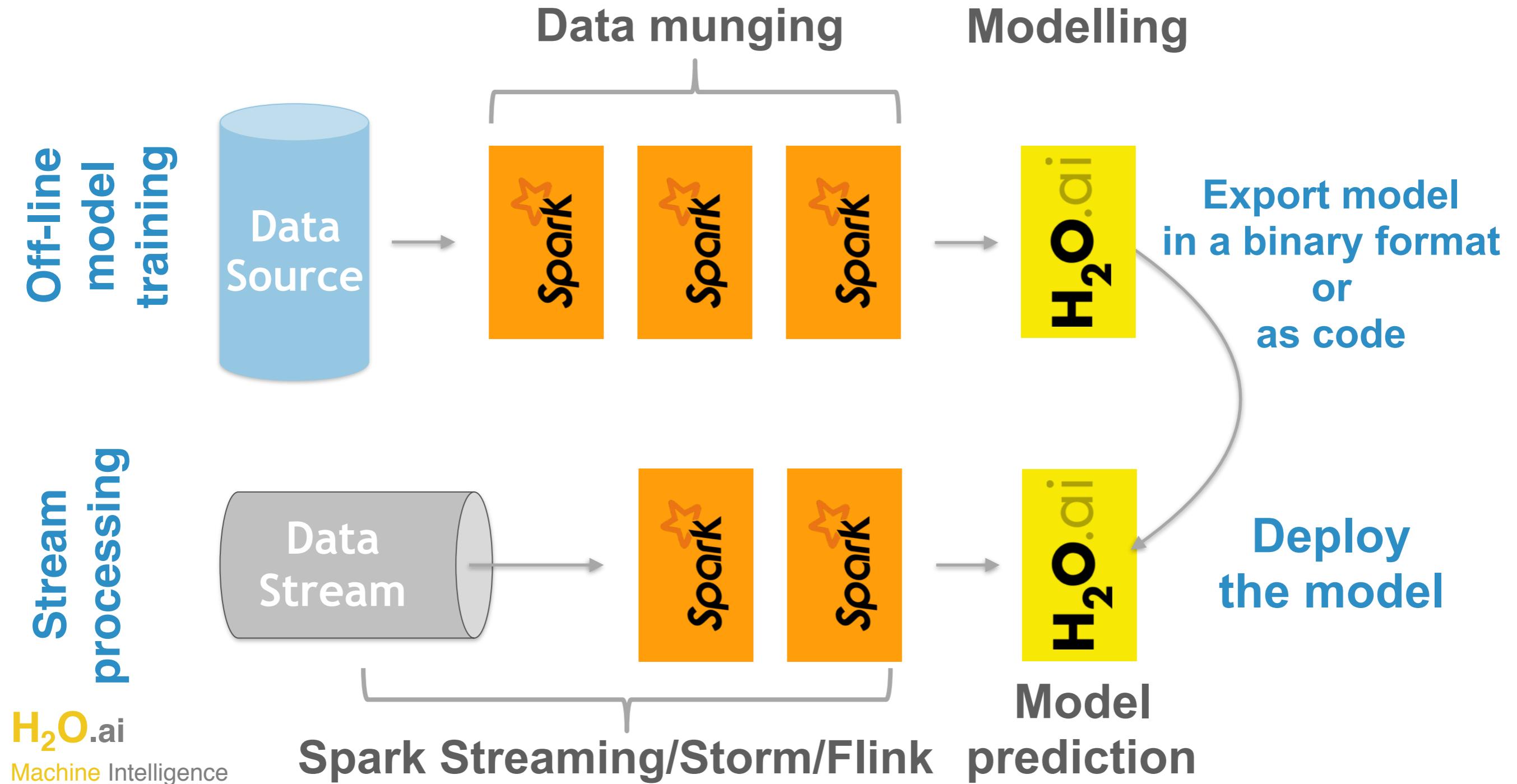
Model Building



Data Munging

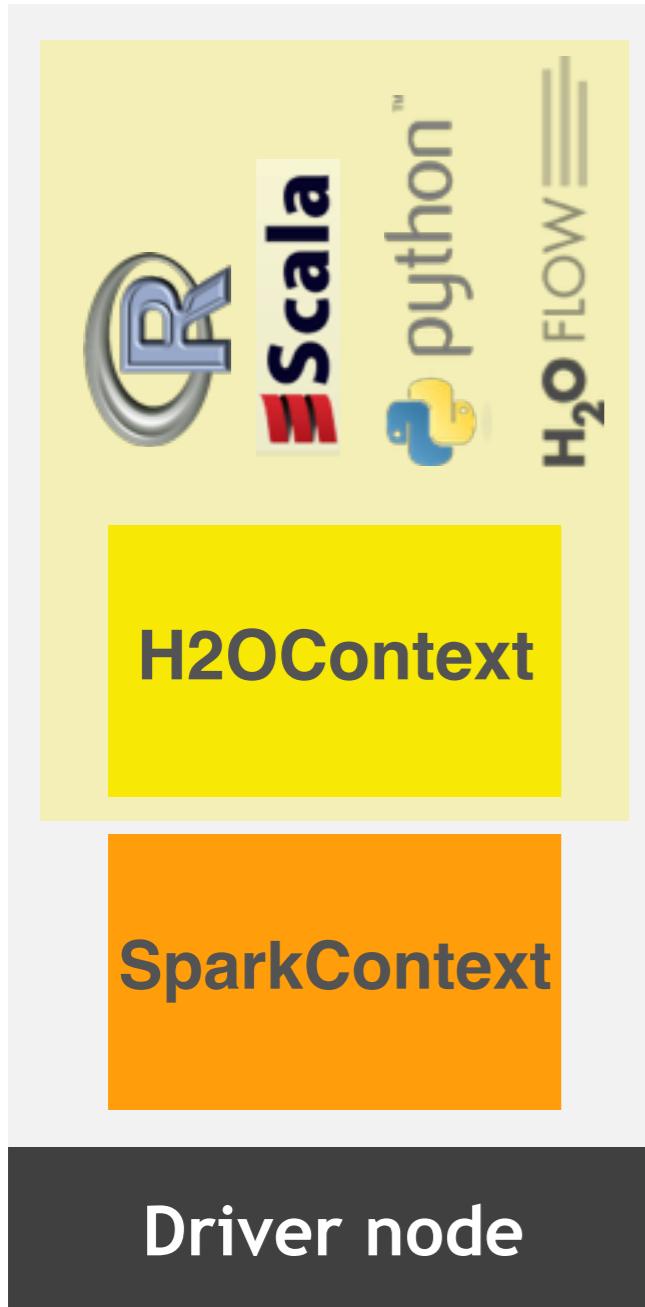


Stream processing



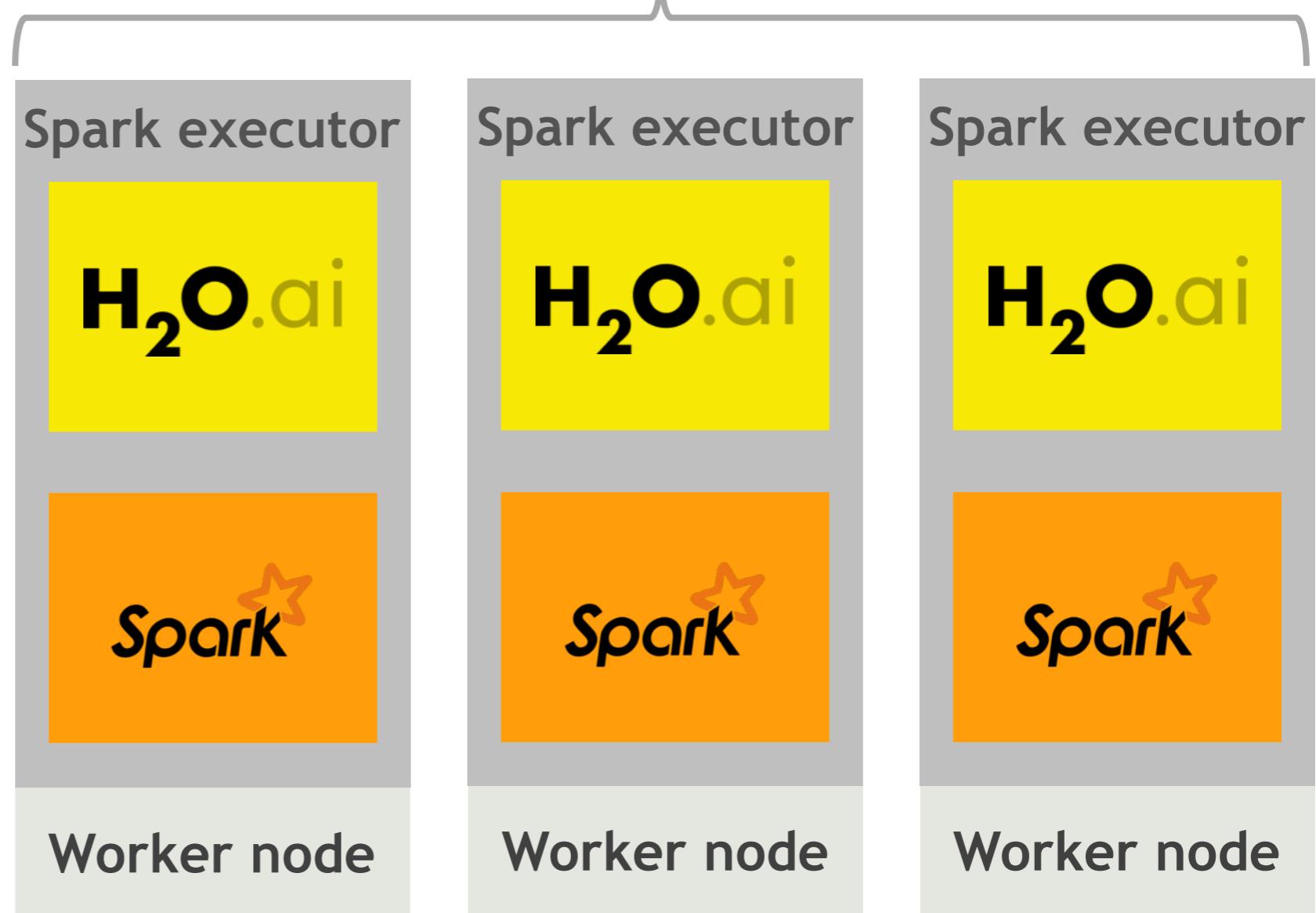
**What is
inside?**

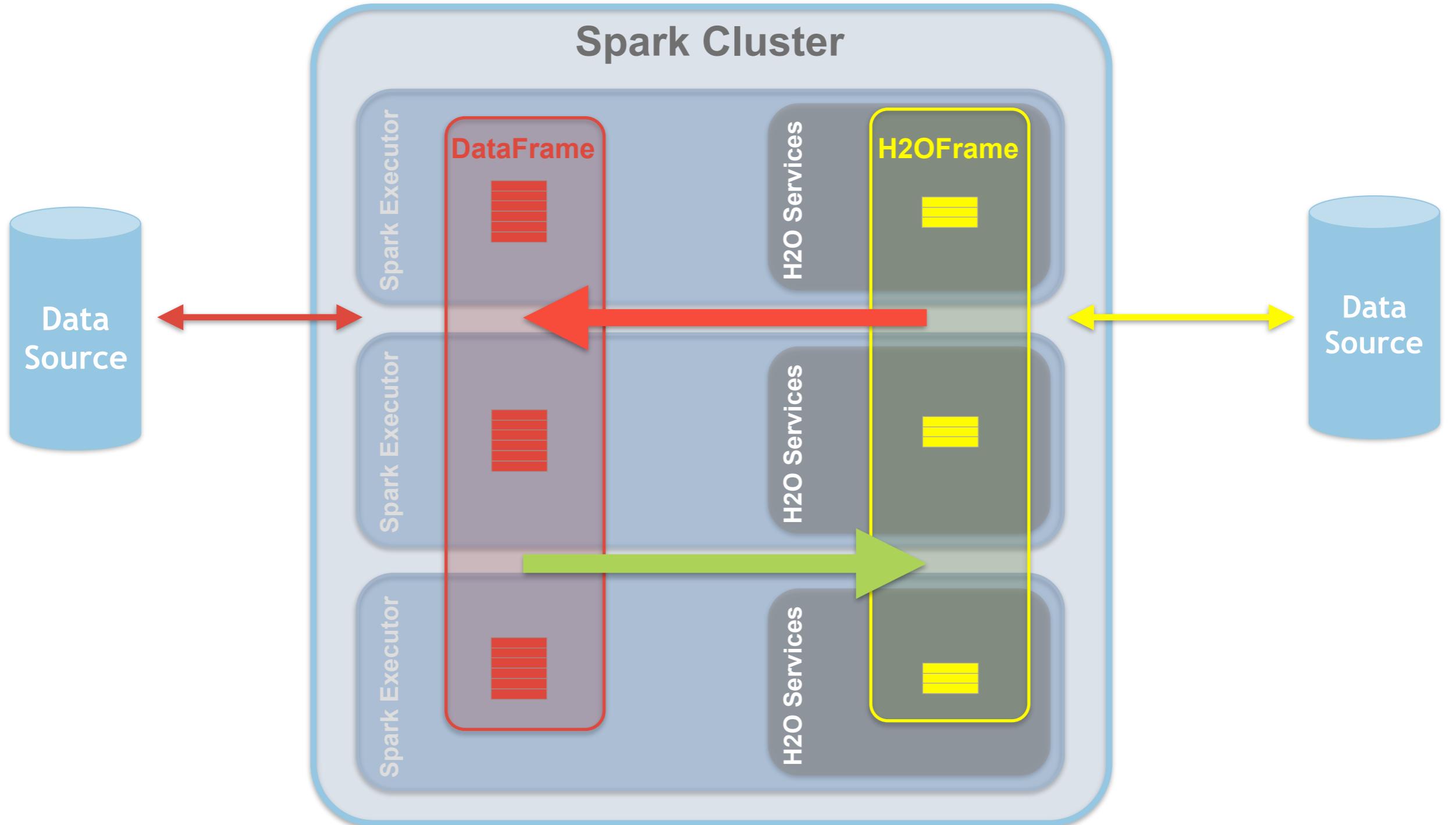
Scala/Py main program



Spark + H₂O

SPARKLING WATER





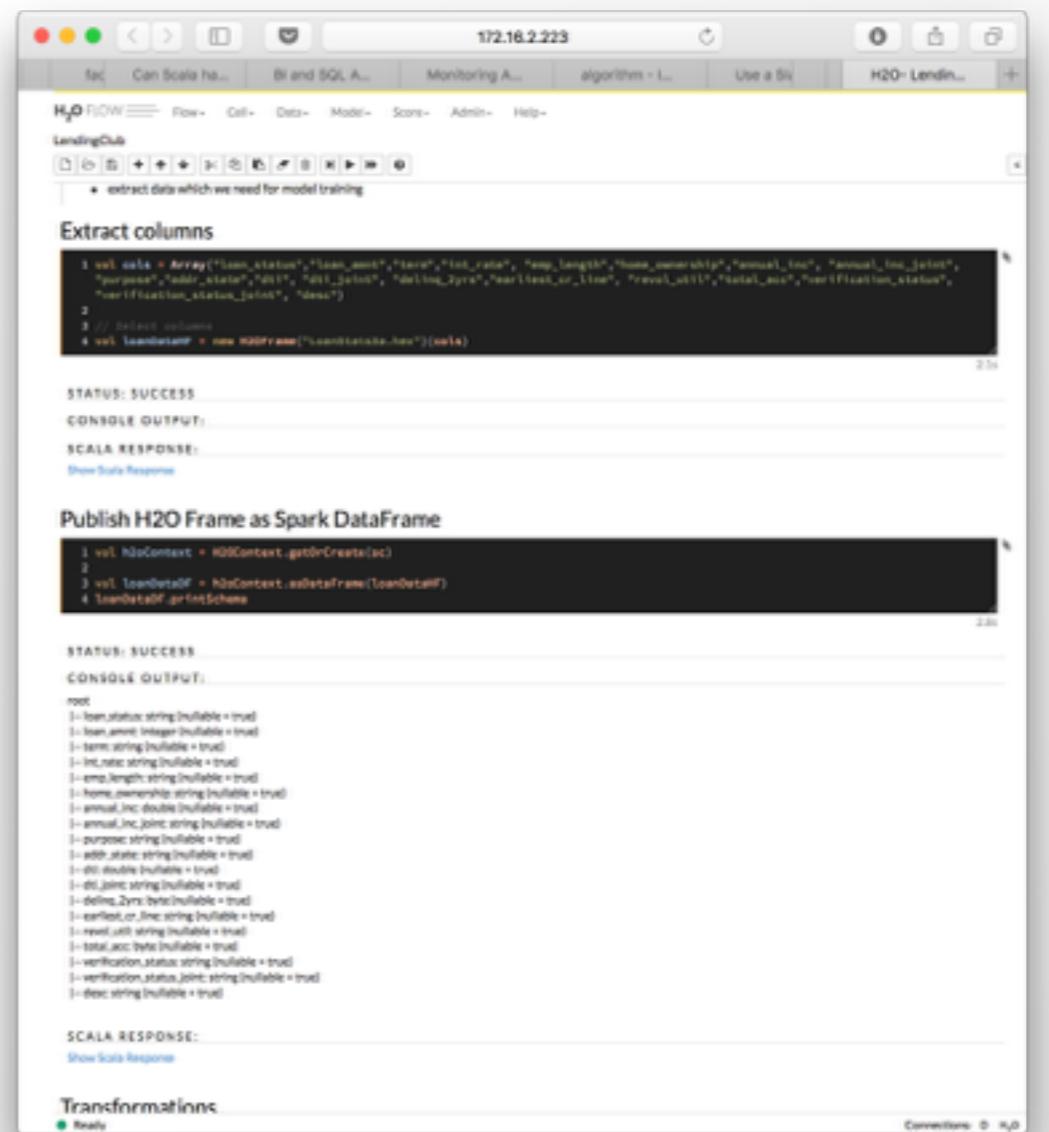
← **h2oContext.asDataFrame**

→ **h2oContext.asH2OFrame**

**New features,
finally!**

Scala code in H2O Flow

- New type of cell
- Access Spark from Flow UI
- Experimenting made easy



The screenshot shows the H2O Flow interface running on a Mac OS X system. The window title is "H2O FLOW" and the URL is "172.16.2.223". The interface has several tabs at the top: "Can Scala ha...", "BI and SQL A...", "Monitoring A...", "algorithms - L...", "Score -", "Admin -", and "Help -". Below the tabs, there's a toolbar with various icons. The main area is divided into sections:

- LendingClub**: A section containing a list of steps:
 - + extract data which we need for model training
- Extract columns**: A code editor showing Scala code:

```
1 val cols = array("loan_status","loan_amnt","term","int_rate","emp_length","home_ownership","annual_inc","annual_inc_joint",
2 "purpose","addr_state","dti","dti_joint","dti_bc_gt_dt","earliest_cr_line","total_rev_hi_lim","total_acc","verification_status",
3 // selected columns
4 val loanDetail = new H2OFrame("%loadframe=loanDetail")
```
- STATUS: SUCCESS**
- CONSOLE OUTPUT:**
- SCALA RESPONSE:**
- Show Scala Response**
- Publish H2O Frame as Spark DataFrame**: Another code editor showing Scala code:

```
1 val h2oContext = H2OContext.getOrCreate(sc)
2
3 val loanDetailDF = h2oContext.asDataFrame(loanDetail)
4 loanDetailDF.getSchema
```
- STATUS: SUCCESS**
- CONSOLE OUTPUT:**
- root**: A list of schema fields:
 - loan_status: string (nullable = true)
 - loan_amnt: integer (nullable = true)
 - term: string (nullable = true)
 - int_rate: string (nullable = true)
 - emp_length: string (nullable = true)
 - home_ownership: string (nullable = true)
 - annual_inc: double (nullable = true)
 - annual_inc_joint: string (nullable = true)
 - purpose: string (nullable = true)
 - addr_state: string (nullable = true)
 - dti: double (nullable = true)
 - dti_bc_gt_dt: string (nullable = true)
 - earliest_cr_line: string (nullable = true)
 - total_rev_hi_lim: string (nullable = true)
 - total_acc: byte (nullable = true)
 - verification_status: string (nullable = true)
 - verification_status_joint: string (nullable = true)
 - desc: string (nullable = true)
- SCALA RESPONSE:**
- Show Scala Response**
- Transformations**: A section showing a single transformation named "Ready".

H2O Frame as Spark's Datasource

- Use native Spark API to load and save data
- Spark can optimise the queries when loading data from H2O Frame
- Use of Spark query optimiser
- Let's see it in practise!

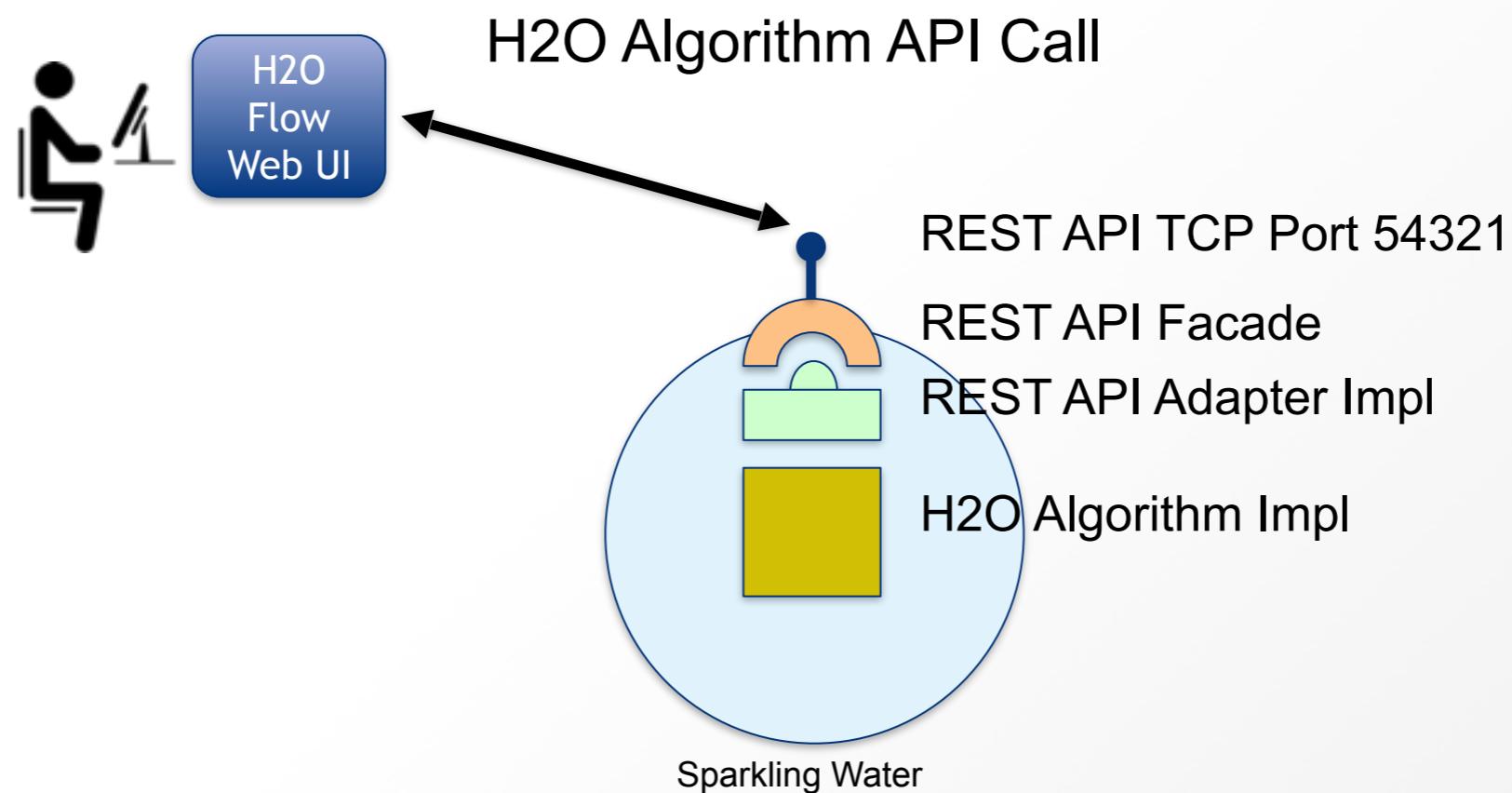
Machine learning pipelines

- Wrap our algorithms as Transformers and Estimators
- Support for embedding them into Spark ML Pipelines
- Can serialise fitted/unfitted pipelines
- Unified API => Arguments are set in the same way for Spark and H2O Models

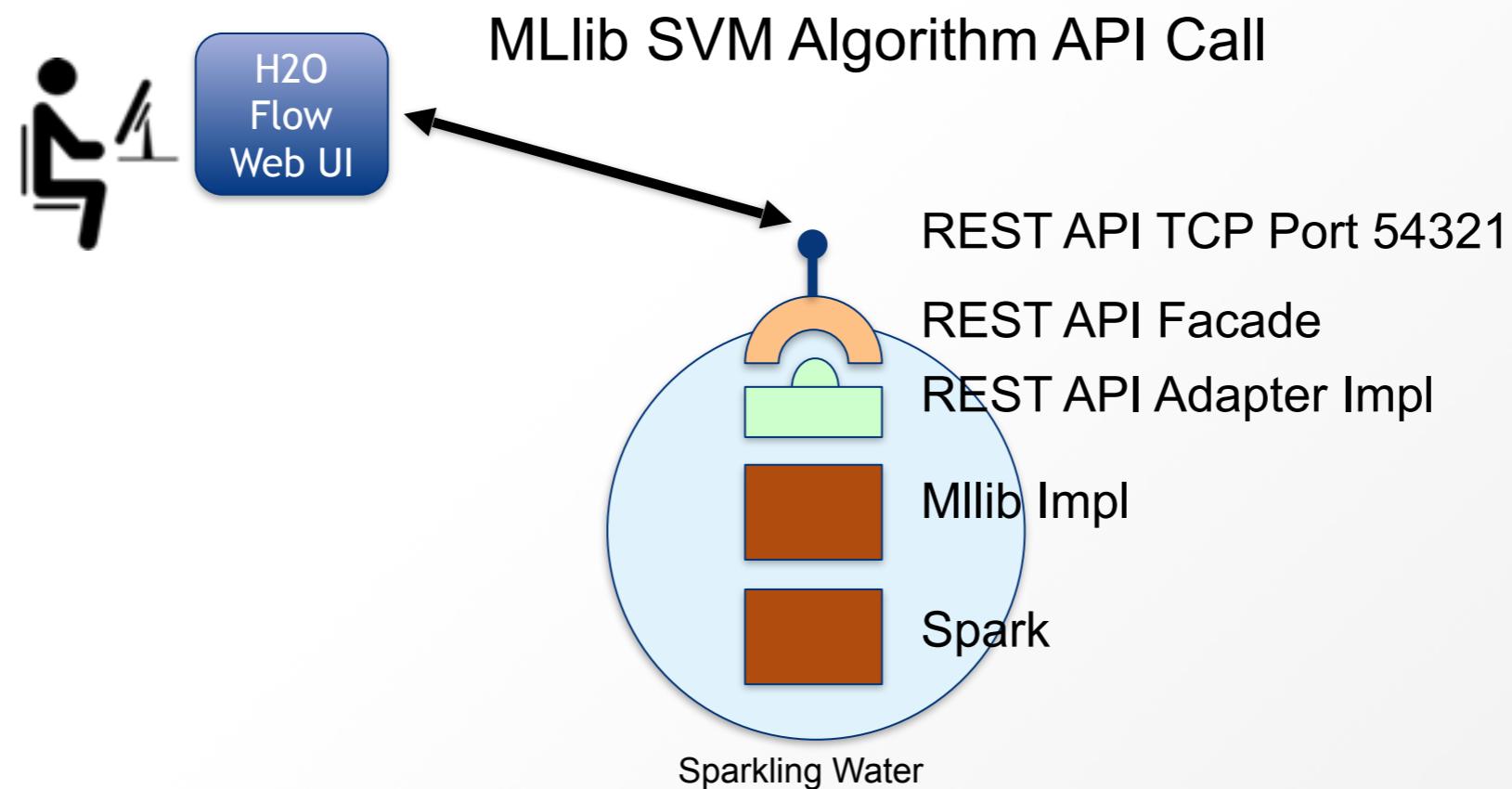
MLlib Algorithms in Flow UI

- Can examine them in H2O Flow
- Can generate POJO out of them
- Let's see Support Vector Machines (SVM)!

H2O Algo from Flow



Pure MLlib Algo from Flow



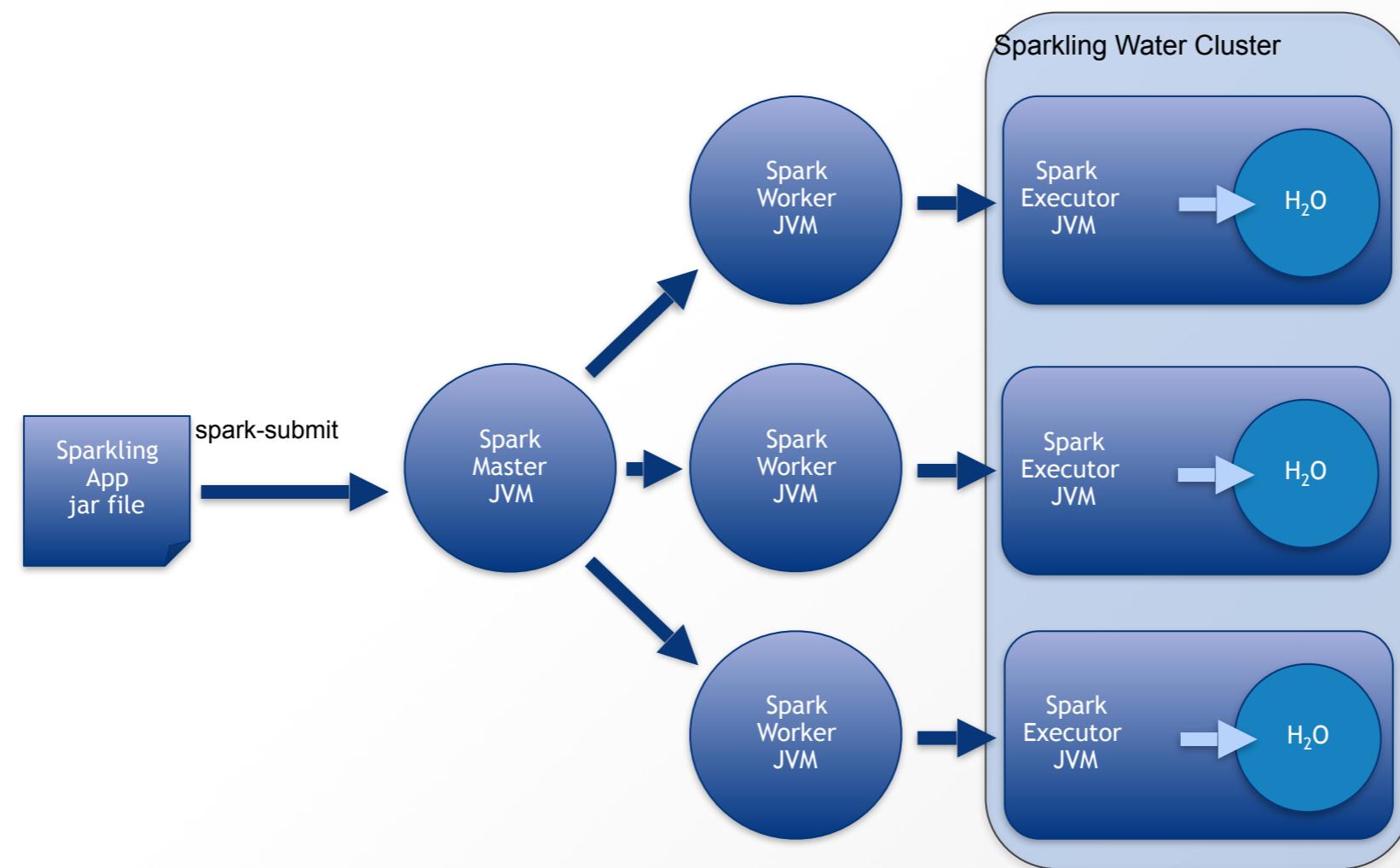
PySparkling made easy

- PySparkling now in PyPi
- Contains all H2O and Sparkling Water dependencies, no need to worry about them
- Just add in on your Python path and that's it

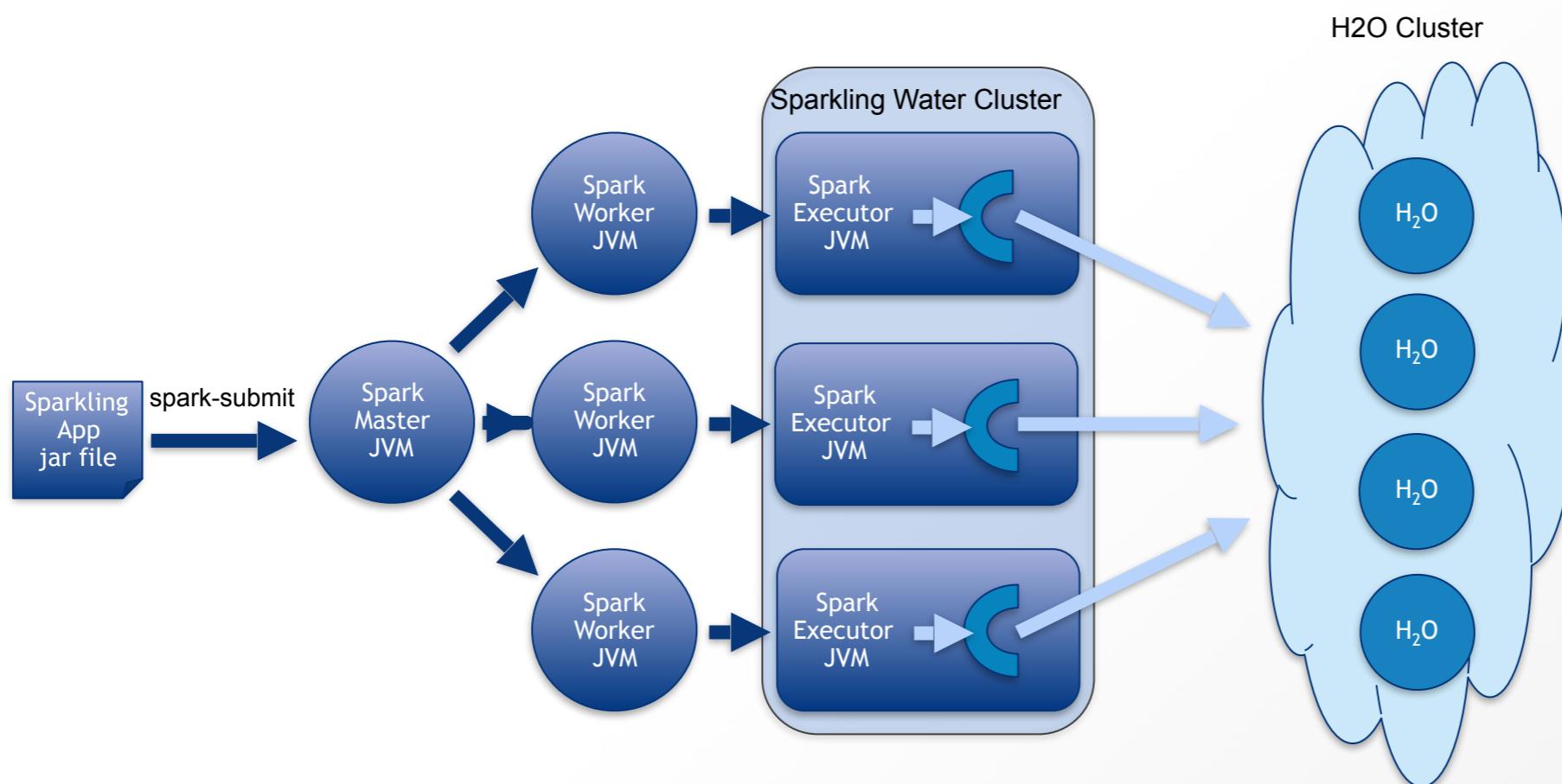
Sparkling Water high-availability

- New solution
- Not yet in master, in testing phase
- Sparkling Water is using external H2O cluster instead of starting H2O in each executor
- Spark executors can come and go and H2O won't be affected

Sparkling Water Internal Backend



Sparkling Water External Backend



And others!

- Support for Datasets
- Zeppelin notebook support
- Integration with TensorFlow
- Support for fast joins
- A Lots of bug fixes..

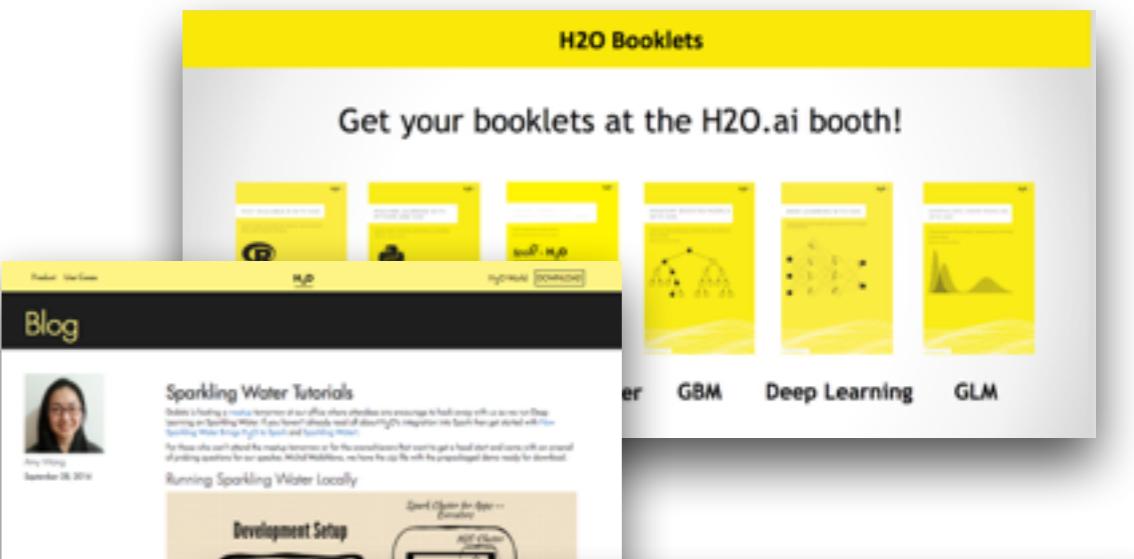
Coming features

- RSparkling
- Support for more MLlib algorithms in Flow
- Python cell in the H2O Flow
- Secure Communication - SSL
- ...

More info

Checkout **H2O.ai** Training Books

<http://h2o.ai/resources>



Checkout **H2O.ai** Blog

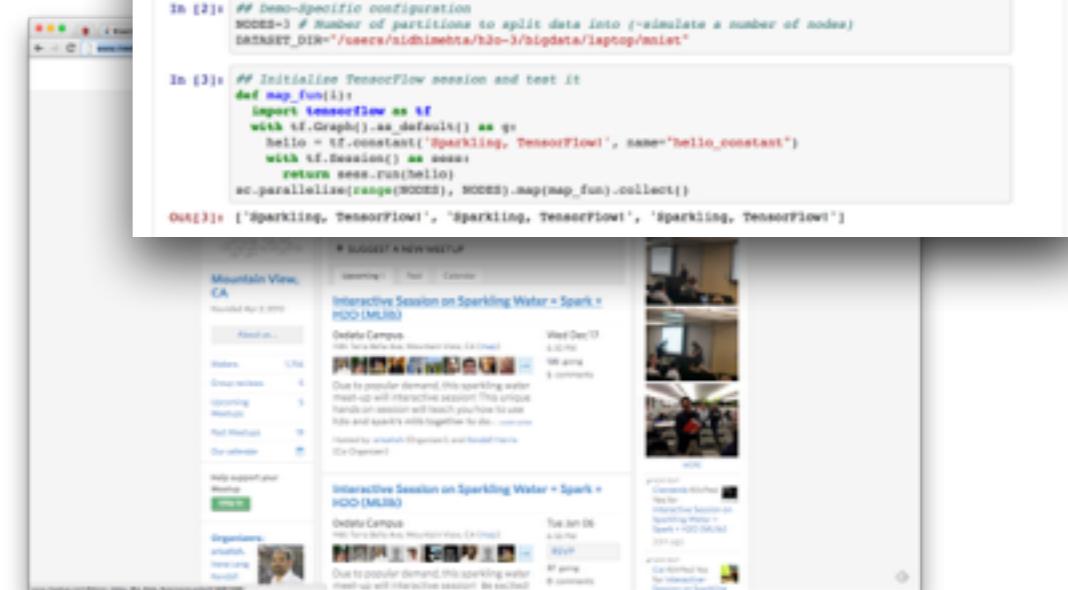
<http://h2o.ai/blog/>

Checkout **H2O.ai** Youtube Channel

<https://www.youtube.com/user/0xdata>

Checkout GitHub

<https://github.com/h2oai/sparkling-water>



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)

