Jakub Háva
jakub@h2o.ai

Sparkling Water 2.0

H2O & Booking.com, Amsterdam,
April 6, 2017

SPARKLING
WATER

Who am I

- Finishing high-performance cluster monitoring tool for JVM based languages (JNI, JVMTI, instrumentation)
- Finishing Master’s at Charles Uni in Prague
- Software engineer at H2O.ai - Core Sparkling Water
- Climbing & 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

H_2O.ai

Machine Intelligence

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?

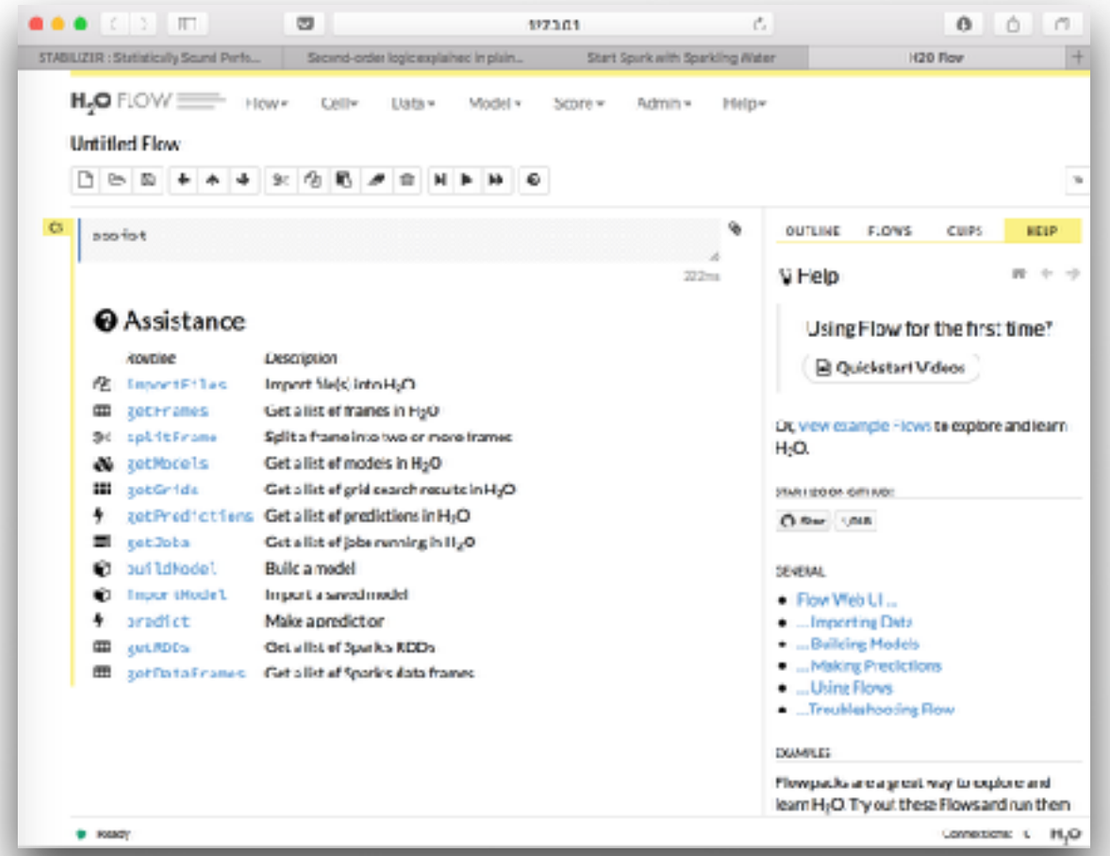
H_2O.ai

Machine Intelligence

Start spark with Sparkling Water

start.sh — Raw

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

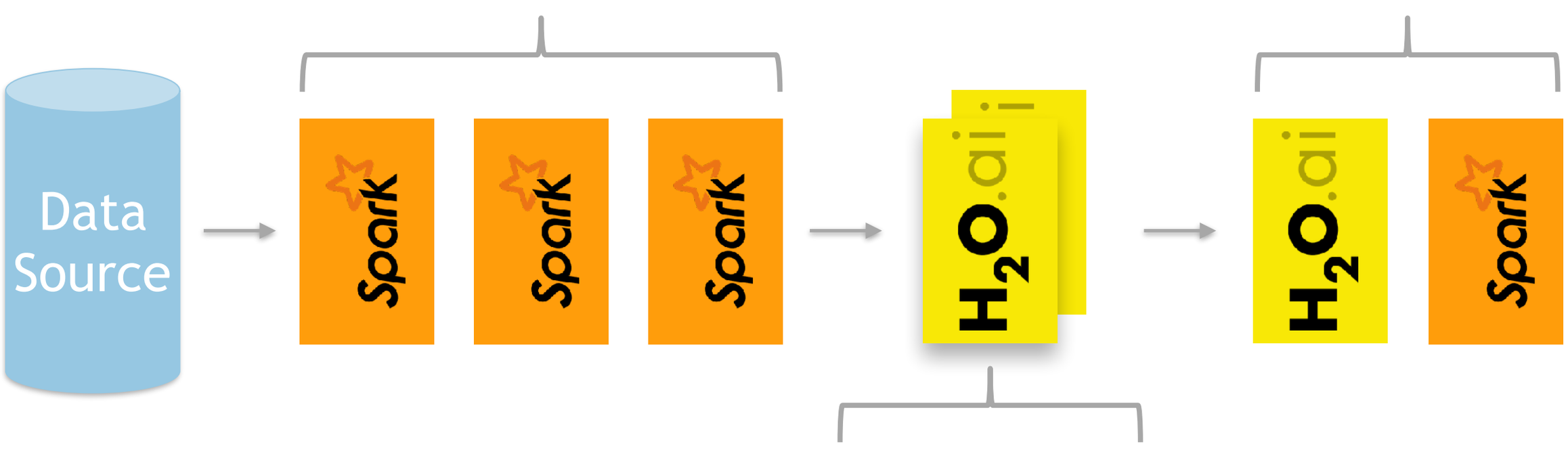


Model Building

Data munging

Modelling

Prediction processing

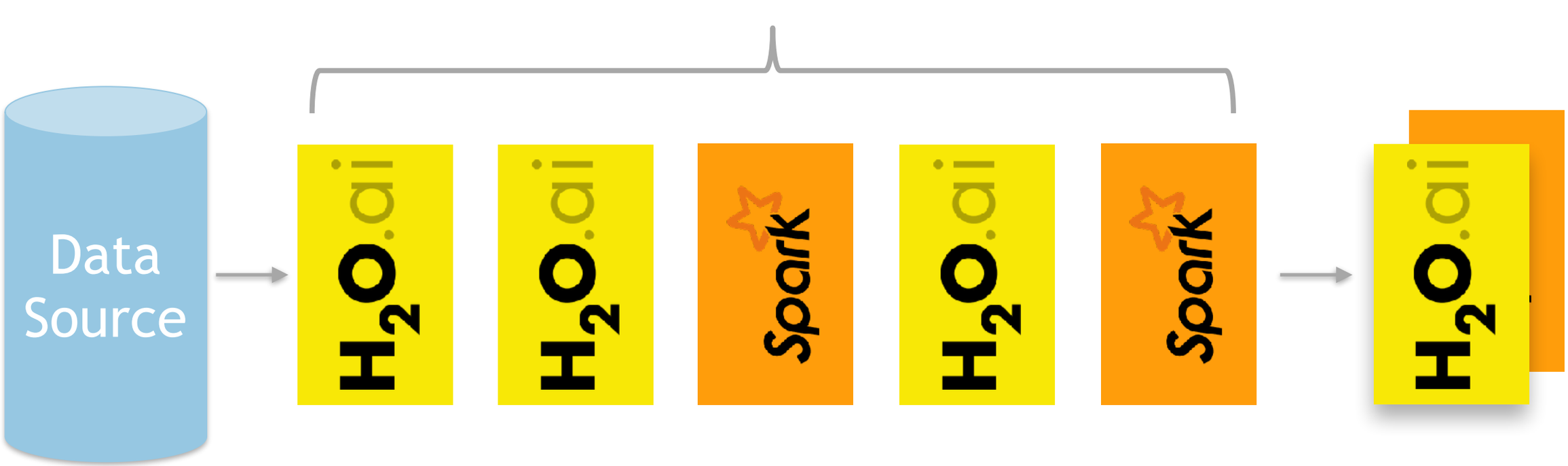


Deep Learning, GBM
DRF, GLM, GLRM
K-Means, PCA
CoxPH, Ensembles

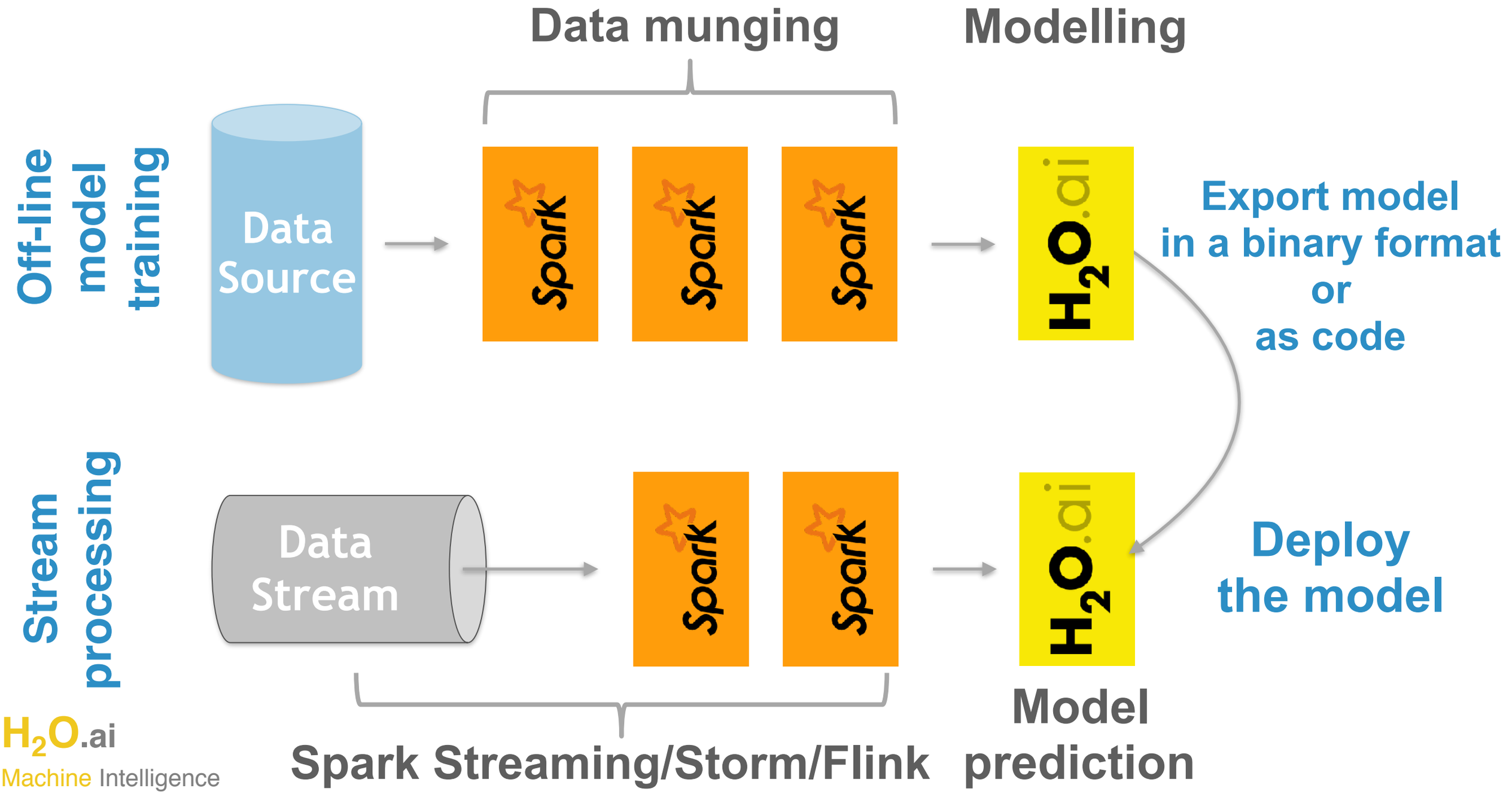
Data Munging

Data load/munging/ exploration

Modelling



Stream processing



H2O.ai
Machine Intelligence

Features

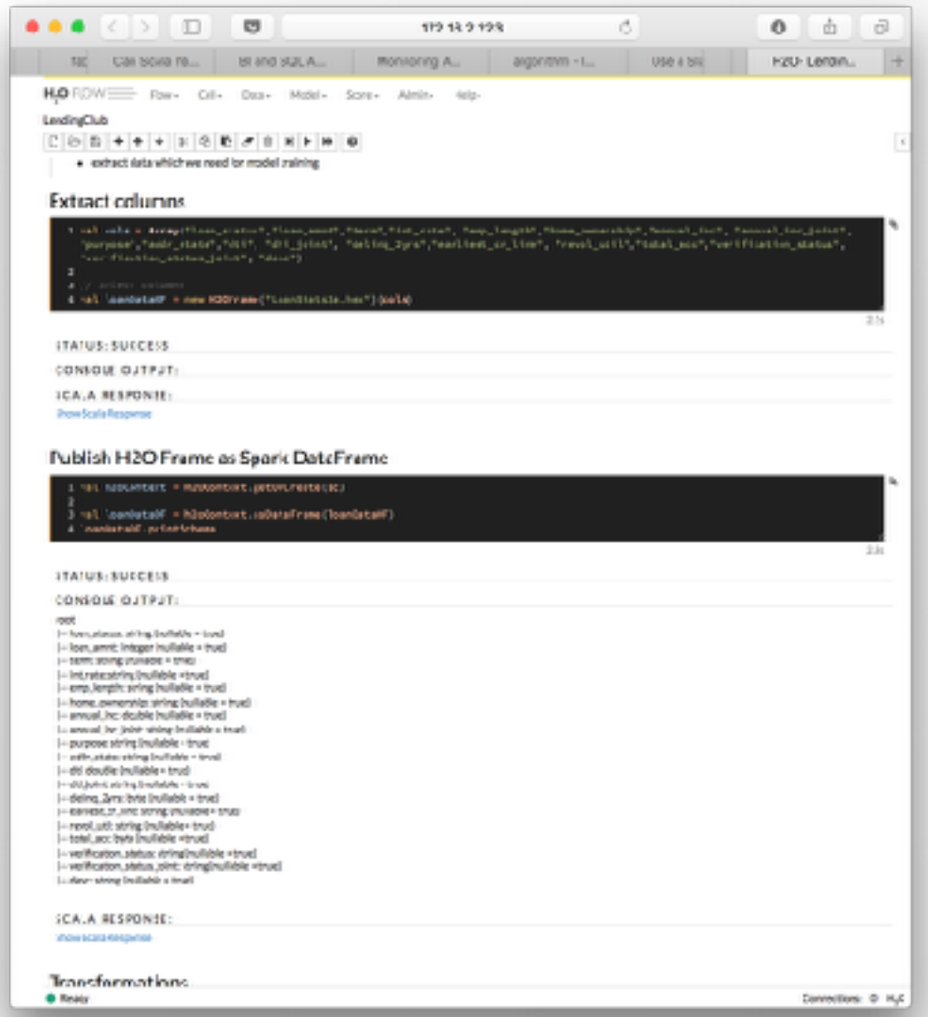
Overview

H_2O.ai

Machine Intelligence

Scala code in H2O Flow

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



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

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

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

And others!

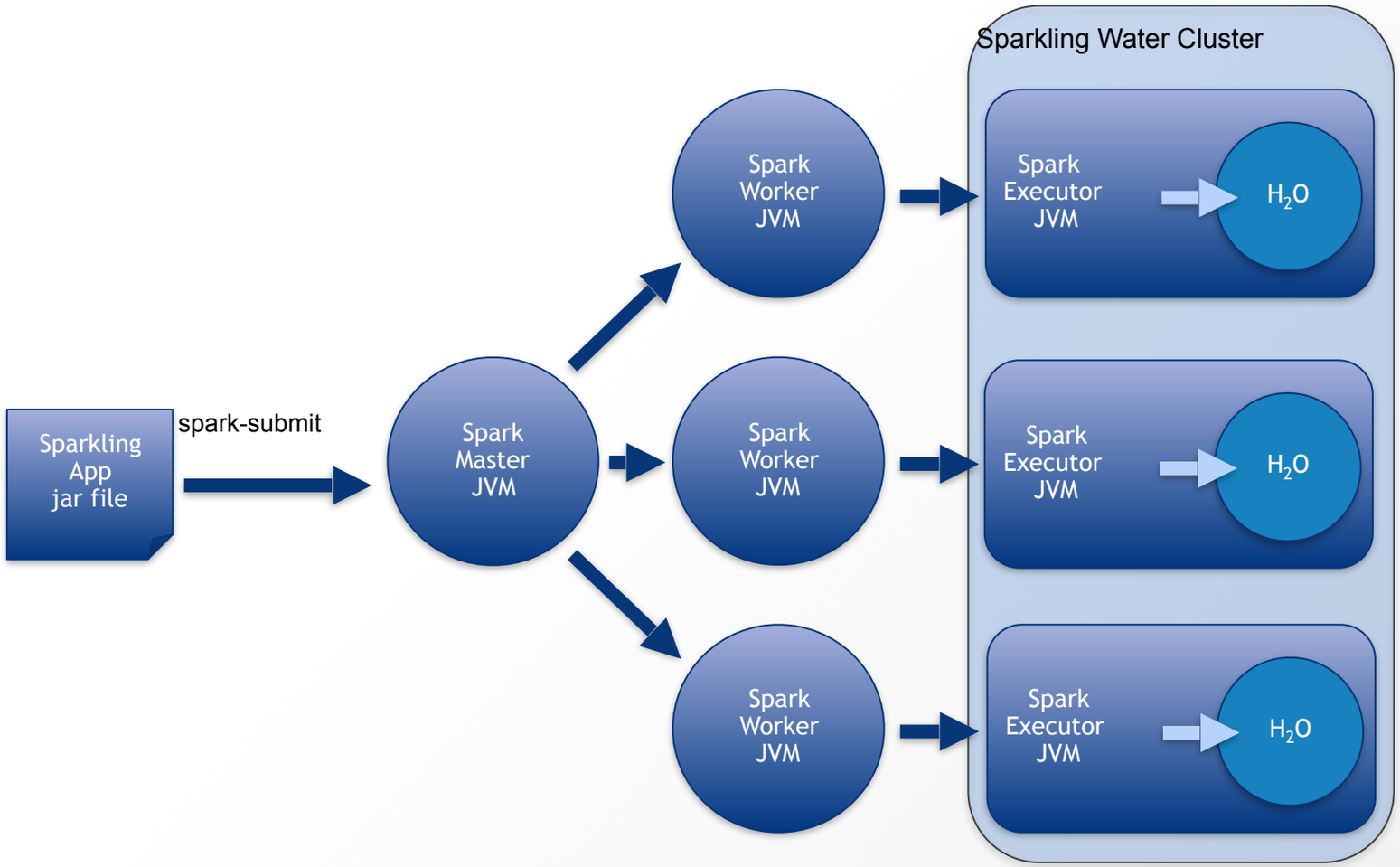
- Support for Datasets
- RSparkling
- Zeppelin notebook support
- Integration with TensorFlow (via DeepWater)
- Support for high-cardinality fast joins
- Secure Communication - SSL
- Bug fixes..

Coming features

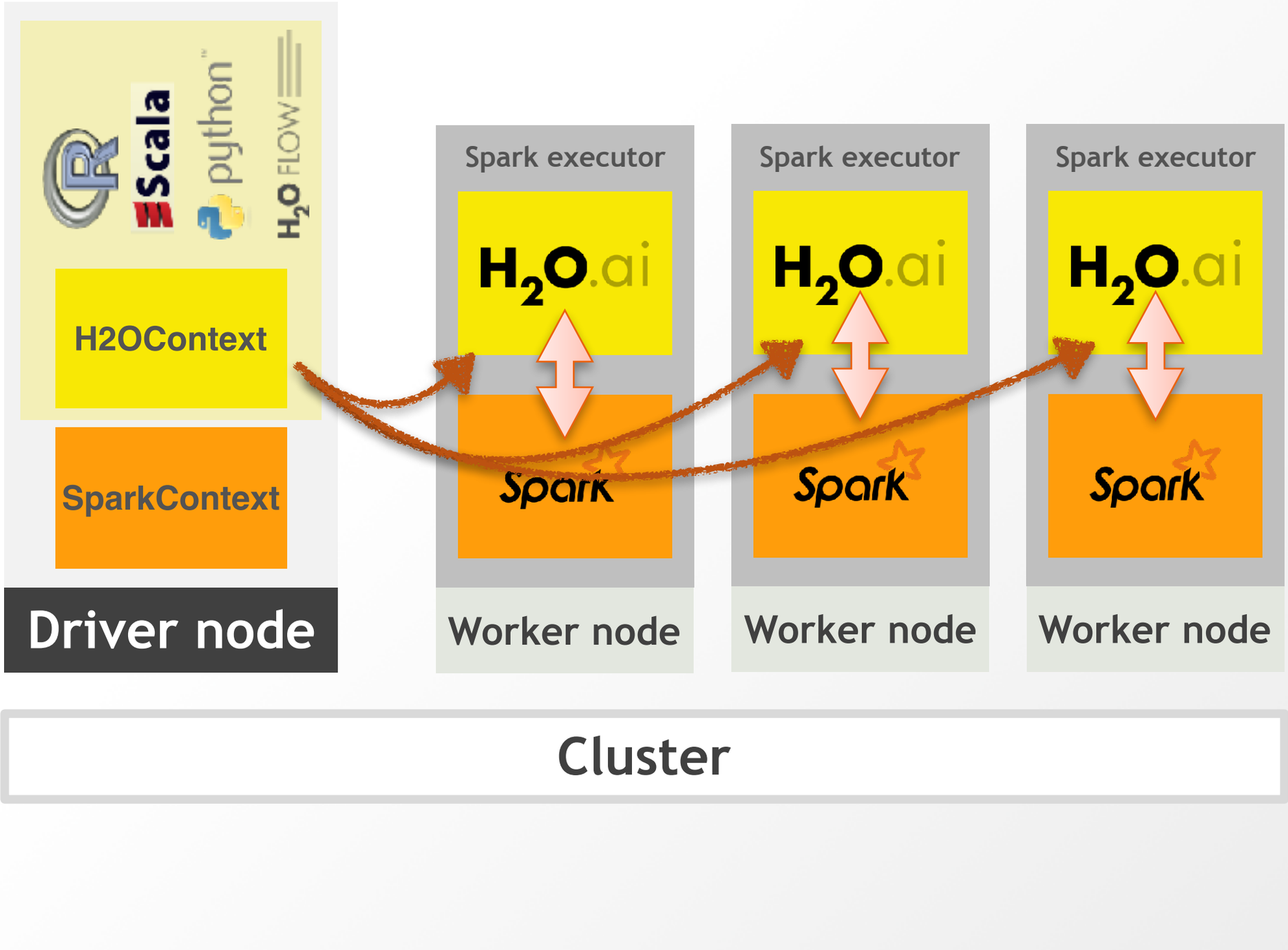
- Support for more MLlib algorithms in Flow
- Python cell in the H2O Flow
- Integration with Steam
- ...

Internal Backend

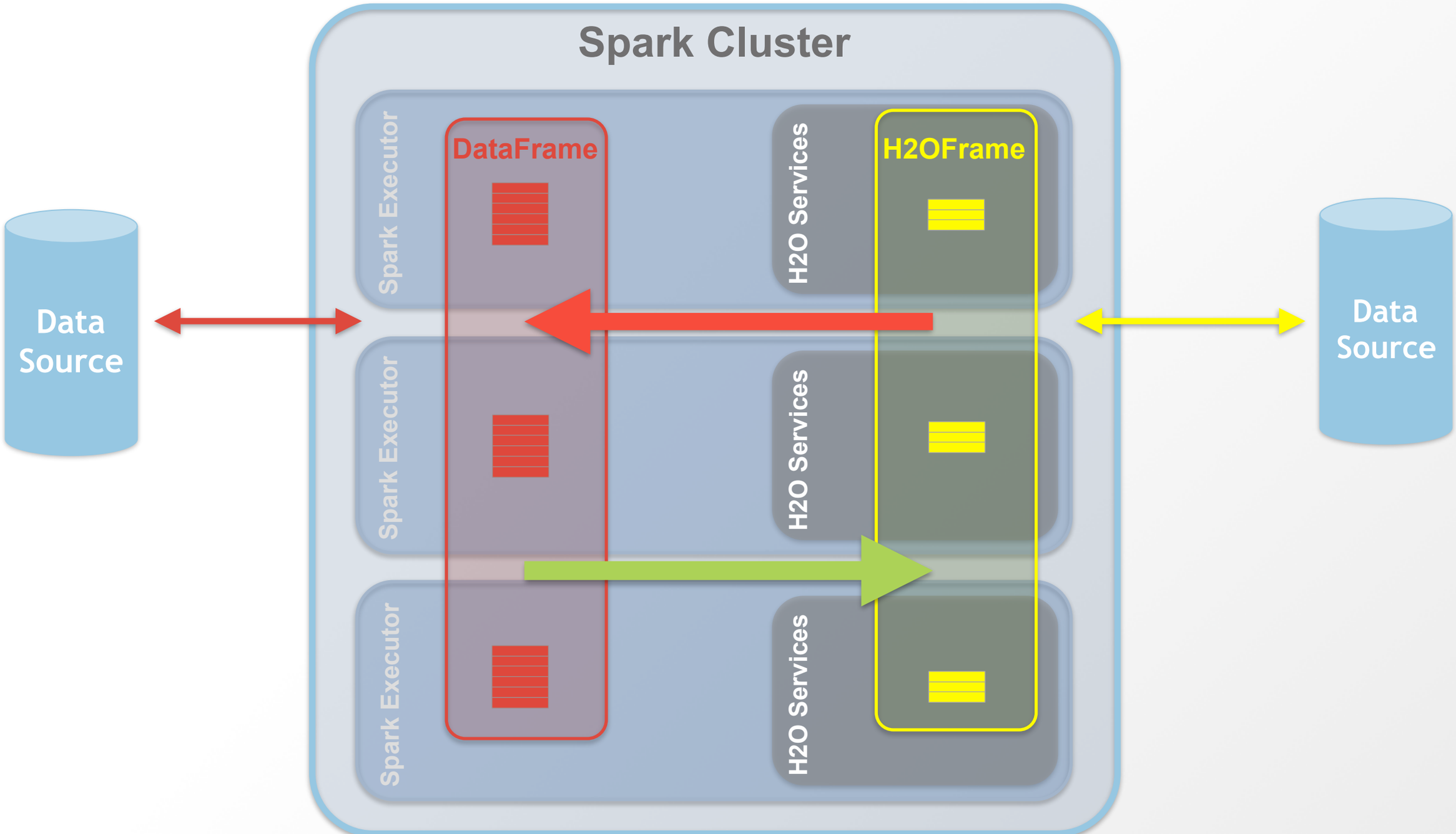
Sparkling Water Internal Backend



Internal Backend



Internal Backend



Pros & Cons

- Advantages
  - Easy to configure
  - Faster (no need to send data to different cluster)
- Disadvantages
  - Spark kills or joins new executor => H2O goes down
  - No way how to discover all executors

When to use

- Have a few Spark Executors
- Stable environment - not many Spark jobs need to be repeated
- Quick jobs

When not To Use

- Unstable environment
- Big number of executors
- Big data, long running tasks which often fails in the middle

How to use

```
from pysparkling import *

conf = H2OConf(sc)
 .set_internal_cluster_mode()
hc = H2OContext.getOrCreate(sc, conf)
```

External Backend

Overview

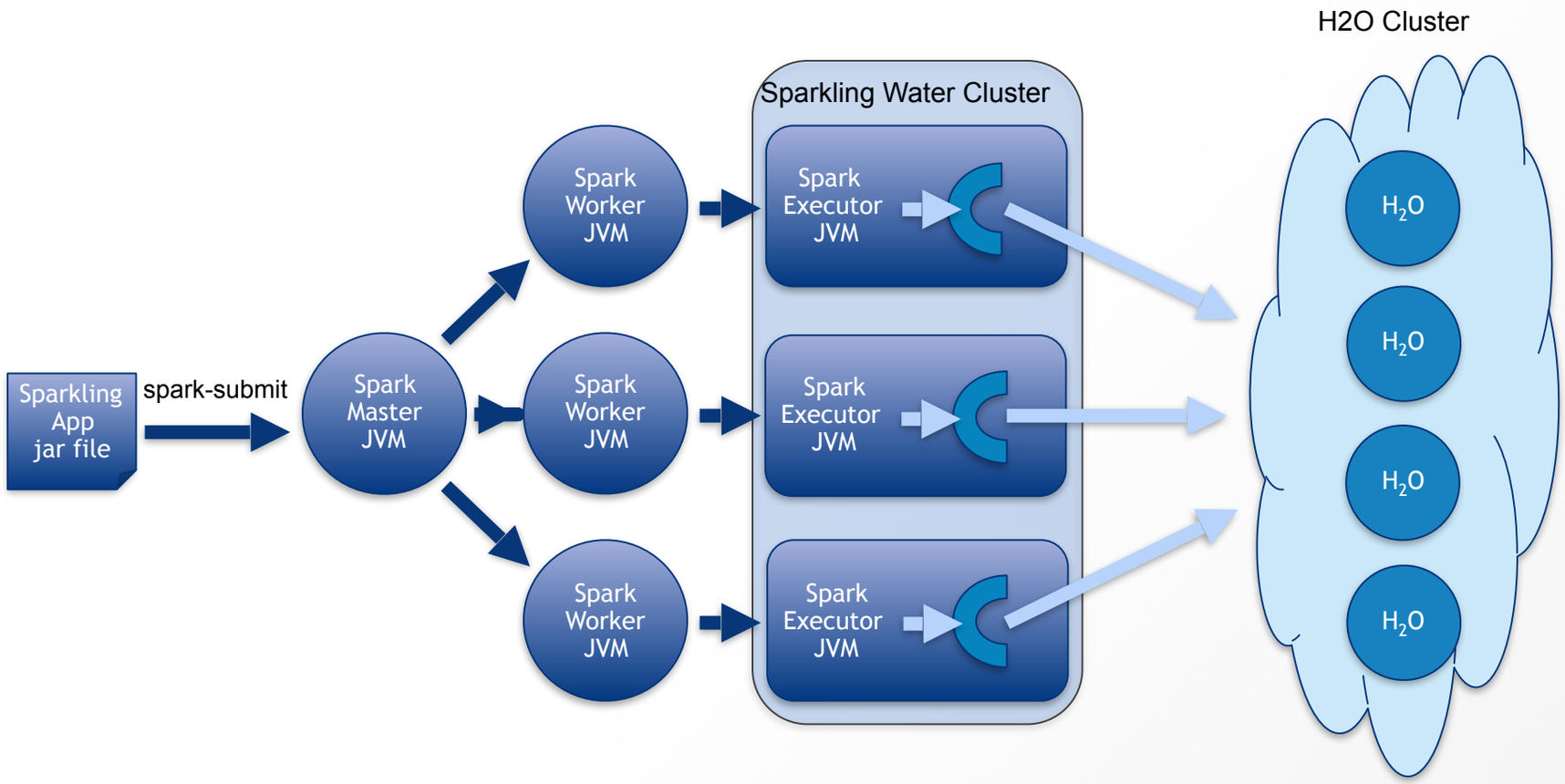
- Sparkling Water is using external H2O cluster instead of starting H2O in each executor
- Spark executors can come and go and H2O won't be affected
- Start H2O cluster on YARN automatically

Separation Approach

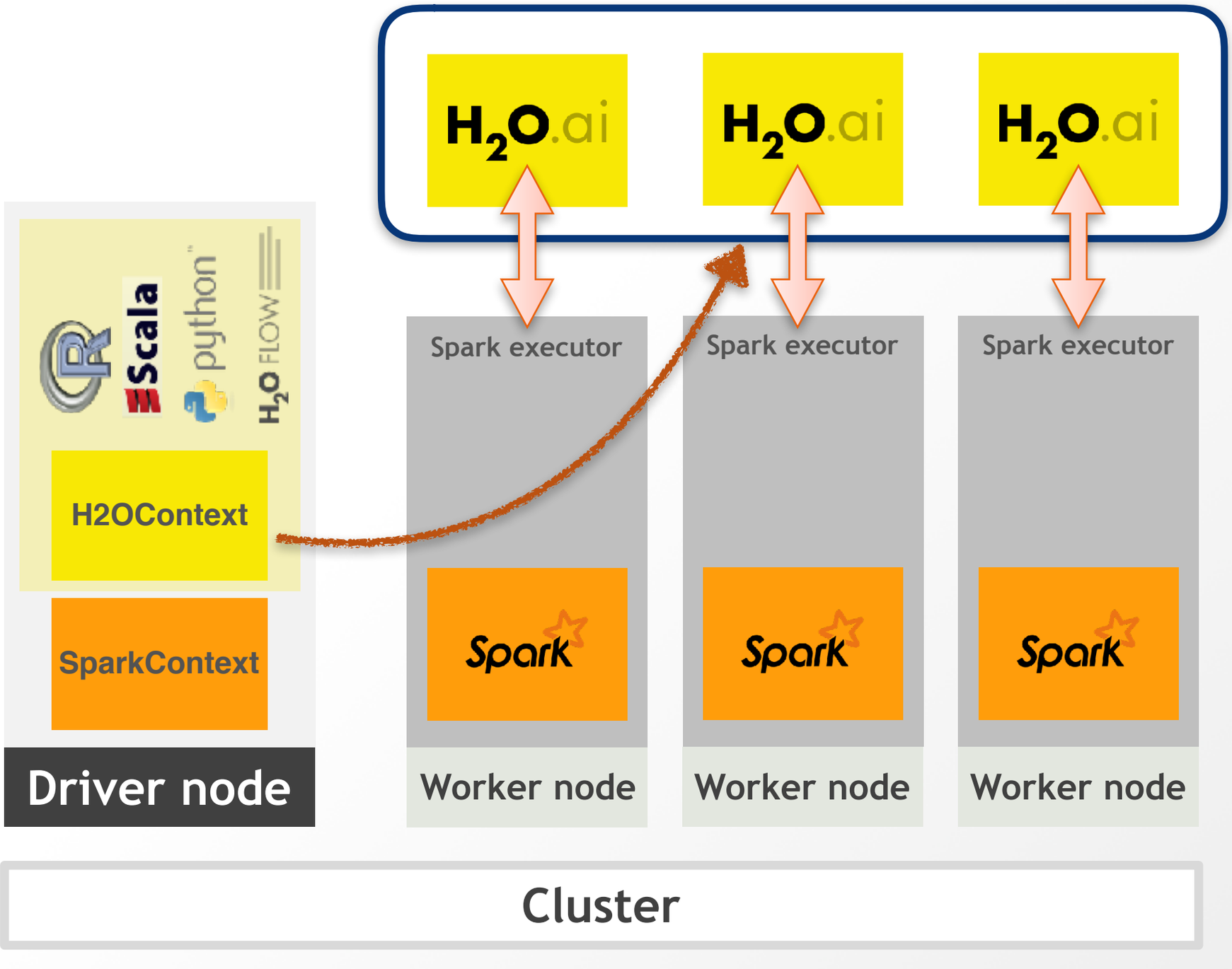
- **Separating Spark and H2O**
  - But preserving same API:
    val h2oContext = H2OContext.getOrCreate("http://h2o:54321/)

- Spark and H2O can be submitted as Yarn job and controlled in separation
  - But H2O still needs non-elastic environment (H2O itself does not implement HA yet)

Sparkling Water External Backend



External Backend



Pros & Cons

- **Advantages**
  - H2O does not crash when Spark executor goes down
  - Better resource management since resources can be planned per tool

- **Disadvantages**
  - Transfer overhead between Spark and H2O processes
    - under measurement with cooperation of a customer

Extended H2O Jar

- External cluster requires H2O jar file to be extended by sparkling water classes
- ./gradlew extendJar -PdownloadH2O=cdh5.8

- We are working on making this even easier so the extended Jars will be available online

Modes

- **Auto Start mode**
  - Start h2o cluster automatically on YARN
- **Manual Start Mode**
  - User is responsible for starting the cluster manually

Manual Cluster Start Mode

```
from pysparkling import *

conf = H2OConf(sc)
 .set_external_cluster_mode()
 .use_manual_cluster_start()
 .set_cloud_name(“test”)
 .set_h2o_driver_path(“path to h2o driver”)

hc = H2OContext.getOrCreate(sc, conf)
```

Auto Cluster Start Mode

```
from pysparkling import *

conf = H2OConf(sc)
 .set_external_cluster_mode()
 .use_auto_cluster_start()
 .set_h2o_driver_path(“path_to_extended_driver")
 .set_num_of_external_h2o_nodes(4)
 .set_mapper_xmx(“2G”)
 .set_yarn_queue(“abc”)

hc = H2OContext.getOrCreate(sc, conf)
```

Multi-network Environment

- Node with Spark driver & H2O client is connected to more networks
- Spark driver choose IP in different network then the external H2O cloud
- We need to manually configure H2O client IP to be on the same network as rest of the H2O cloud

Demo Time

More info

Checkout H2O.ai Training Books

http://h2o.ai/resources

Checkout H2O.ai Blog

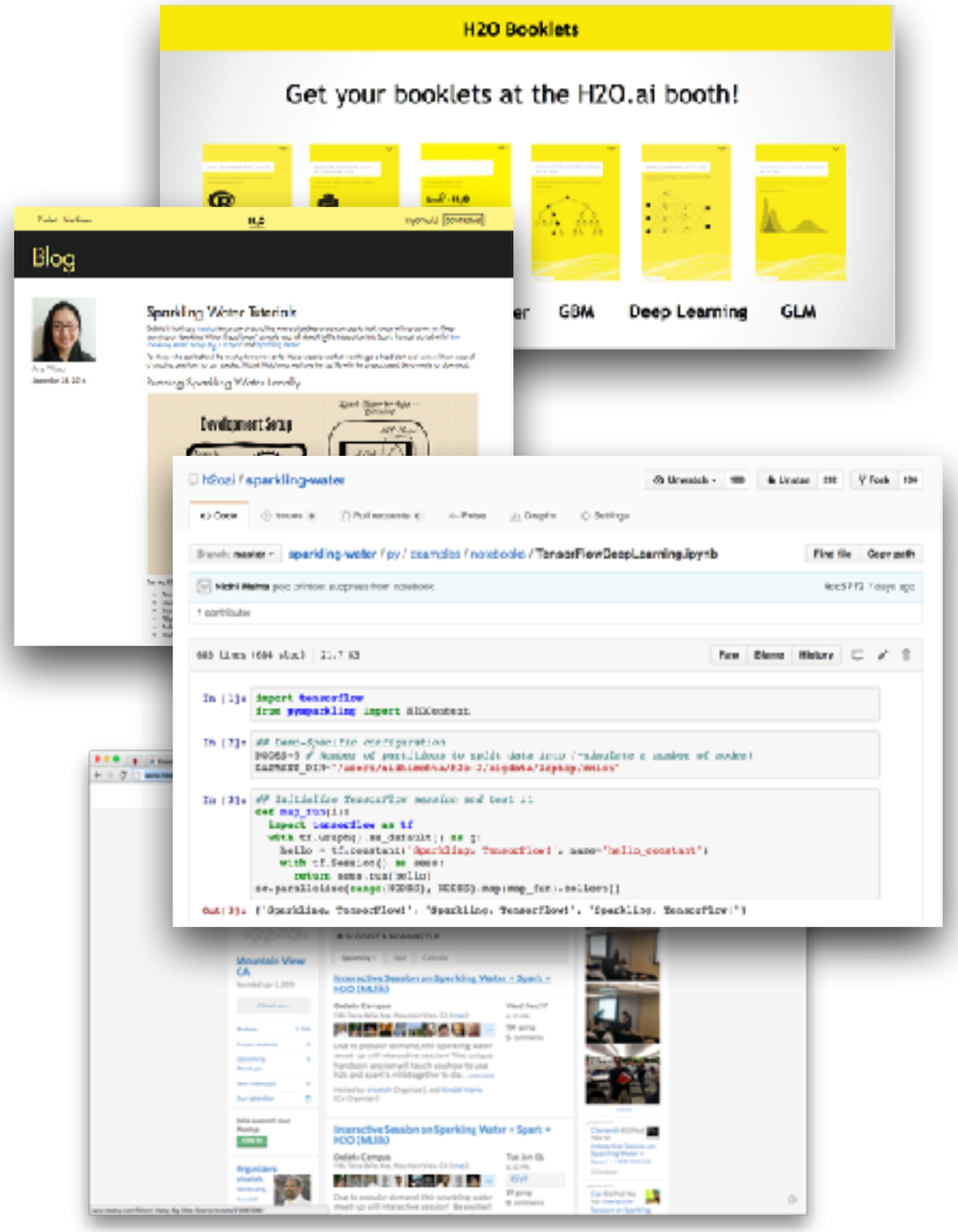
http://h2o.ai/blog/

Checkout H2O.ai Youtube Channel

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

Checkout GitHub

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



H2O.ai
Machine Intelligence

H2O.ai

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