

# On Premise Spark-as-a-Service on YARN

Jim Dowling

Associate Prof @ KTH, Stockholm  
Senior Researcher, SICS Swedish ICT  
CEO, Logical Clocks AB  
Twitter: @jim\_dowling

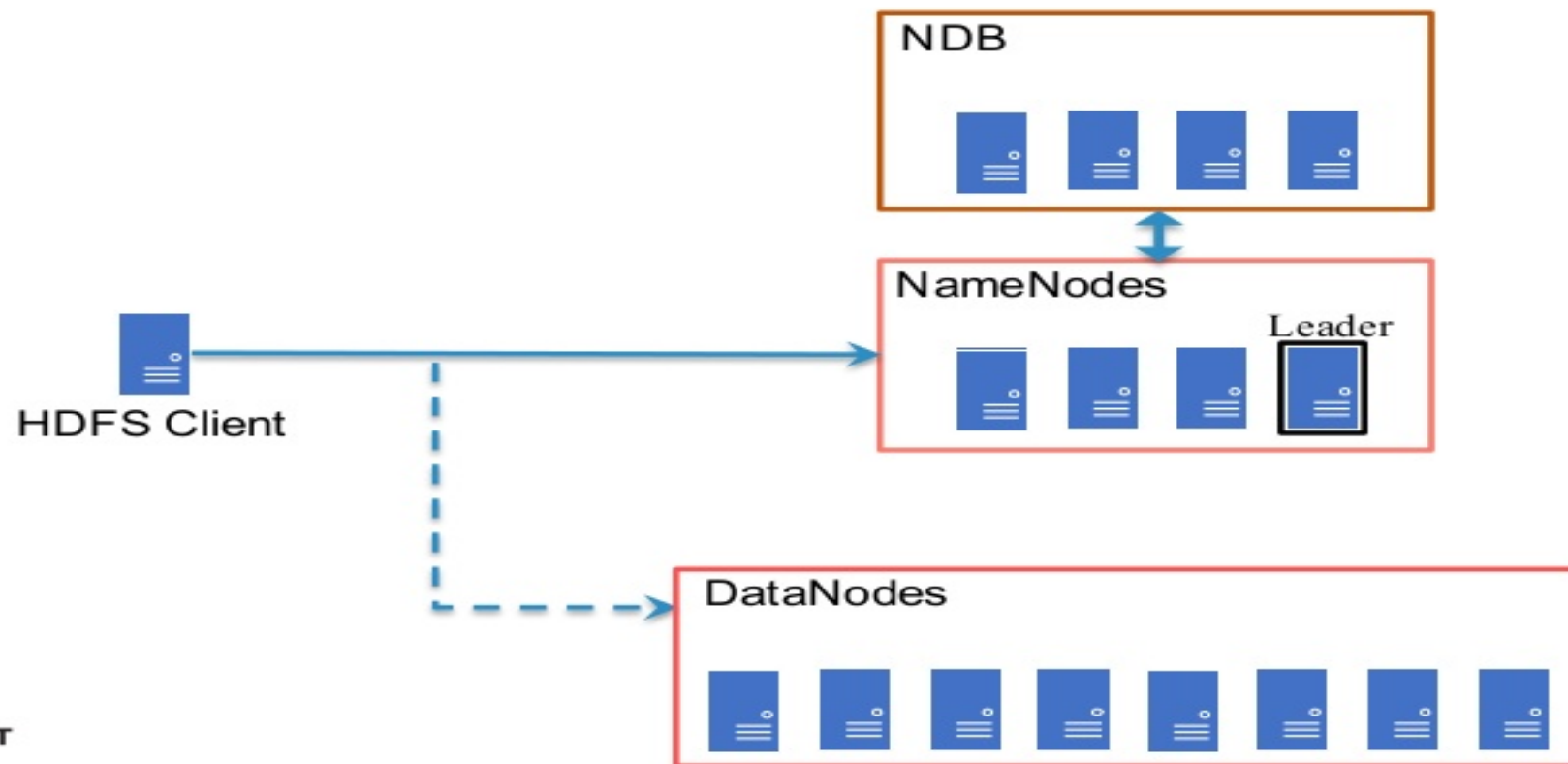


# Spark-as-a-Service in Sweden

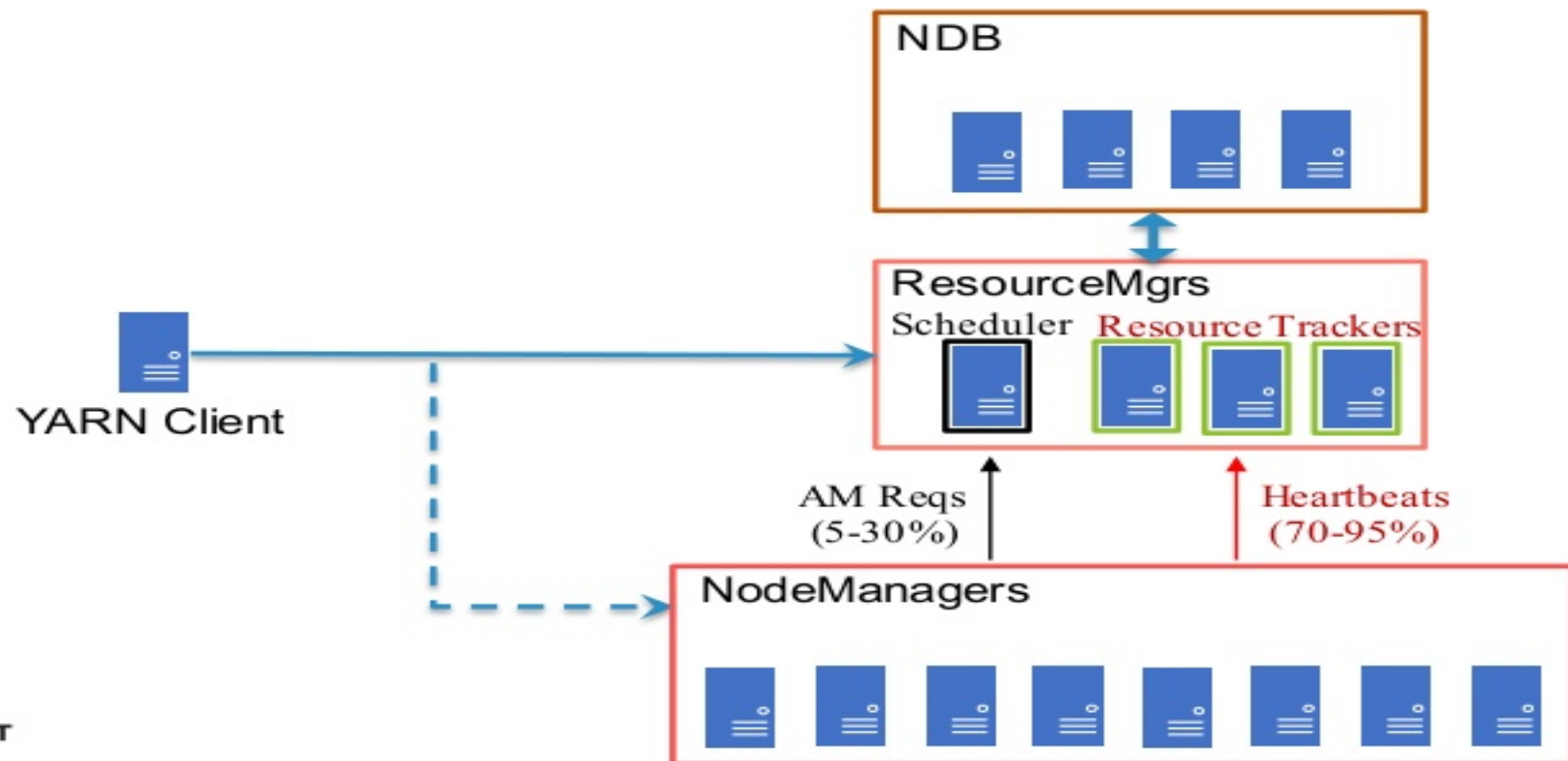
- **SICS ICE:** datacenter research and test environment
- **Hopsworks:** Spark/Kafka/Flink/Hadoop-as-a-service
  - Built on Hops Hadoop ([www.hops.io](http://www.hops.io))
  - Over 100 active users
  - Spark the platform of choice



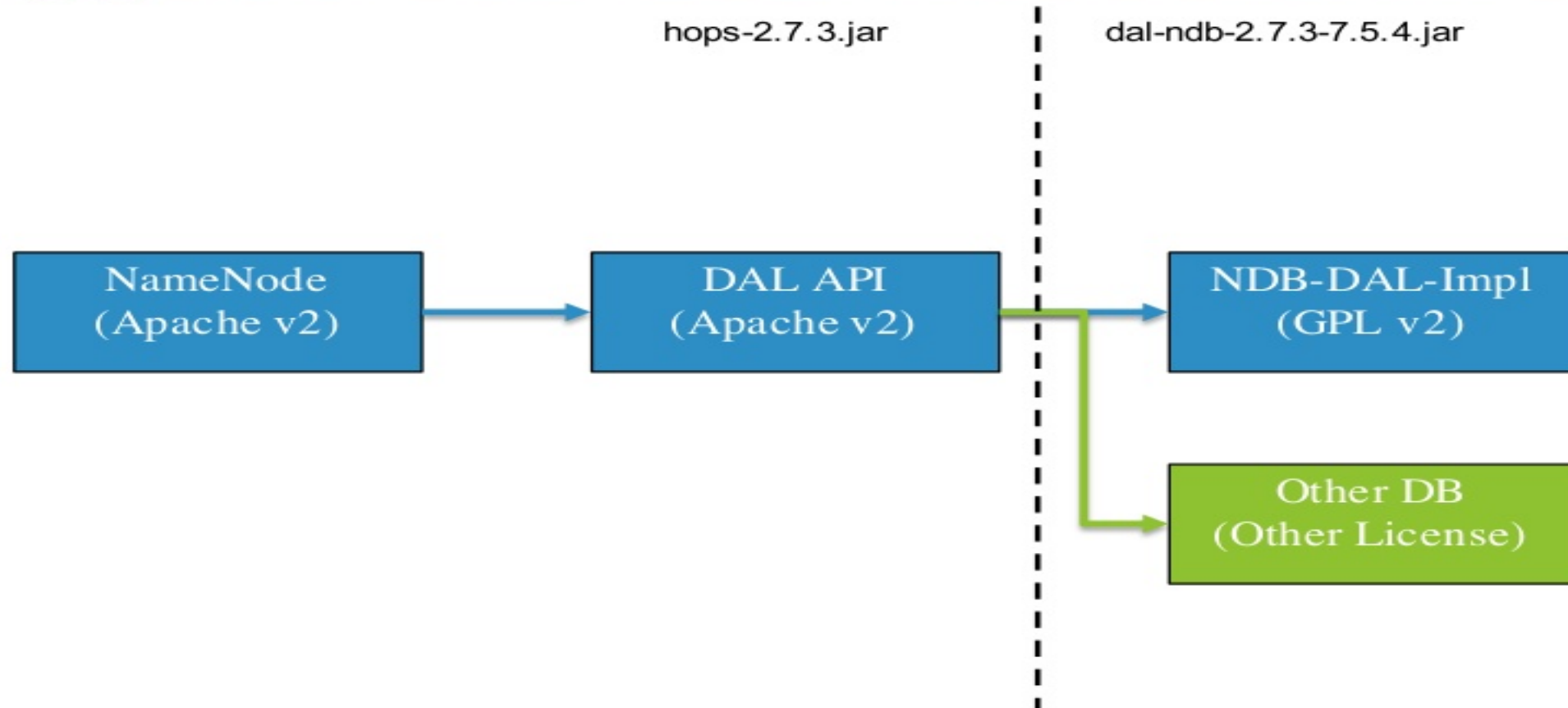
# HopsFS Architecture



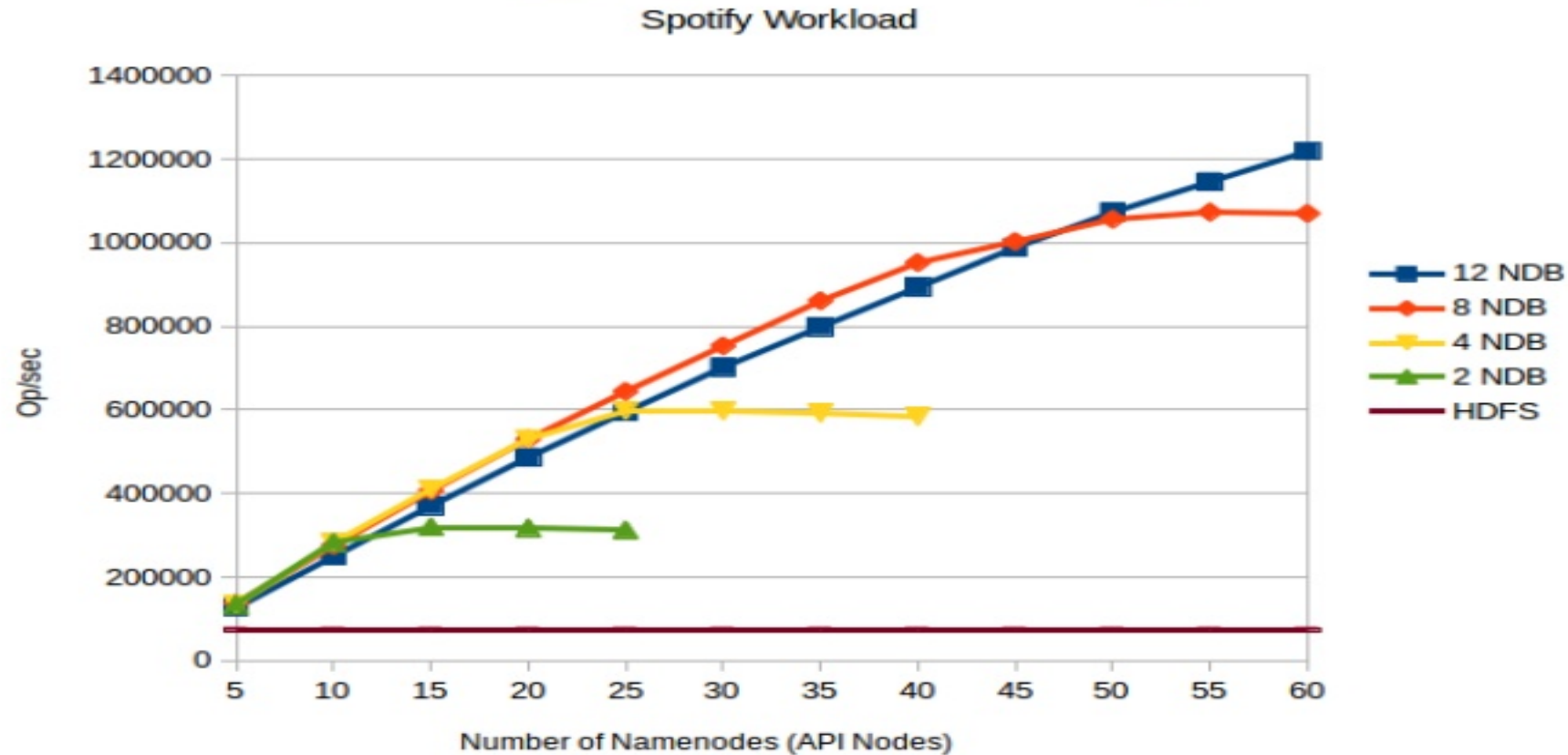
# Hops-YARN Architecture



# Pluggable DB: Data Abstraction Layer



# HopsFS Throughput vs Apache HDFS



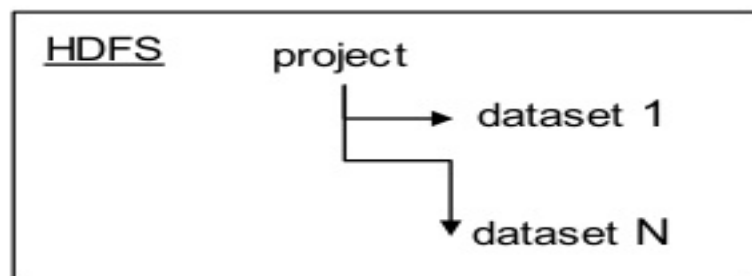
NDB Setup: Nodes using Xeon E5-2620 2.40GHz Processors and 10GbE.  
NameNodes: Xeon E5-2620 2.40GHz Processors machines and 10GbE.





# HOPSWORKS

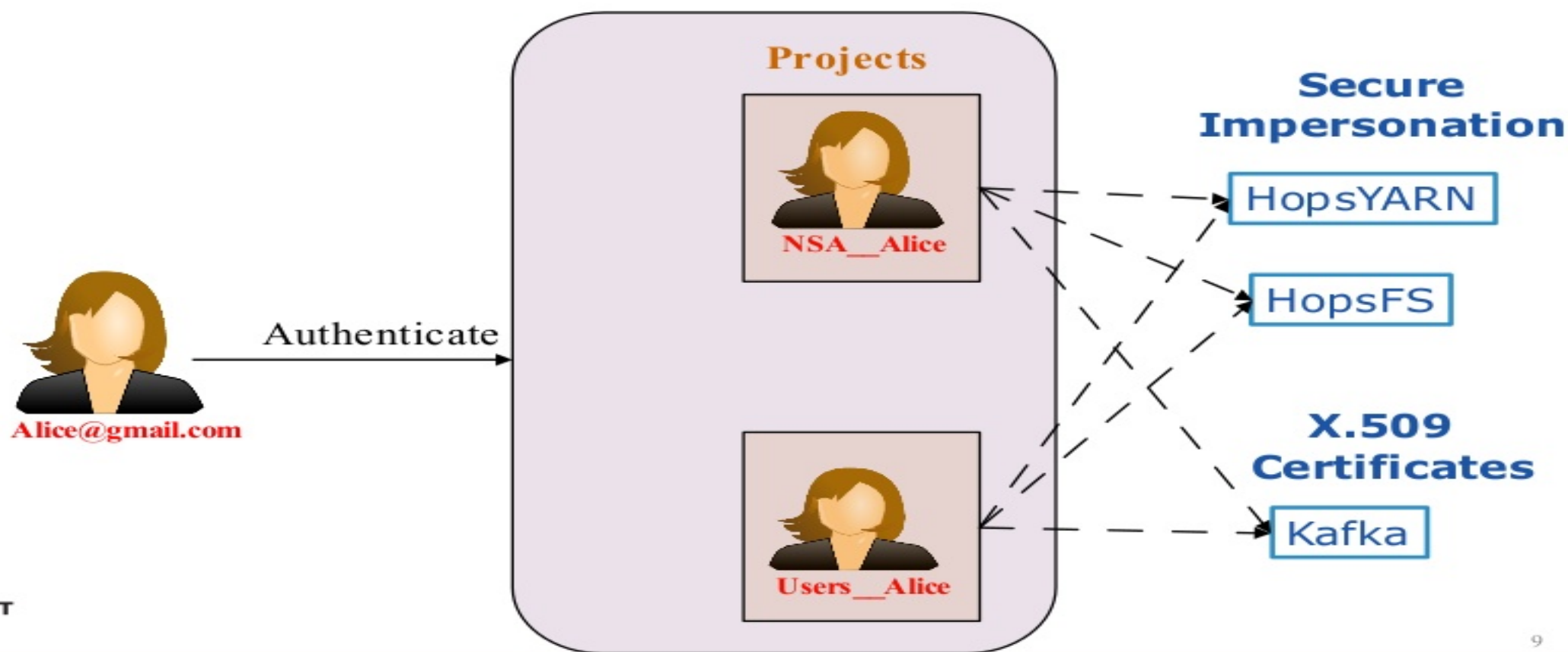
# Project-Based Multi-Tenancy



- A project is a collection of
  - Users with Roles
  - HDFS DataSets
  - Kafka Topics
  - Notebooks, Jobs
- Per-Project quotas
  - Storage in HDFS
  - CPU in YARN
    - Uber-style Pricing
- Sharing across Projects
  - Datasets/Topics



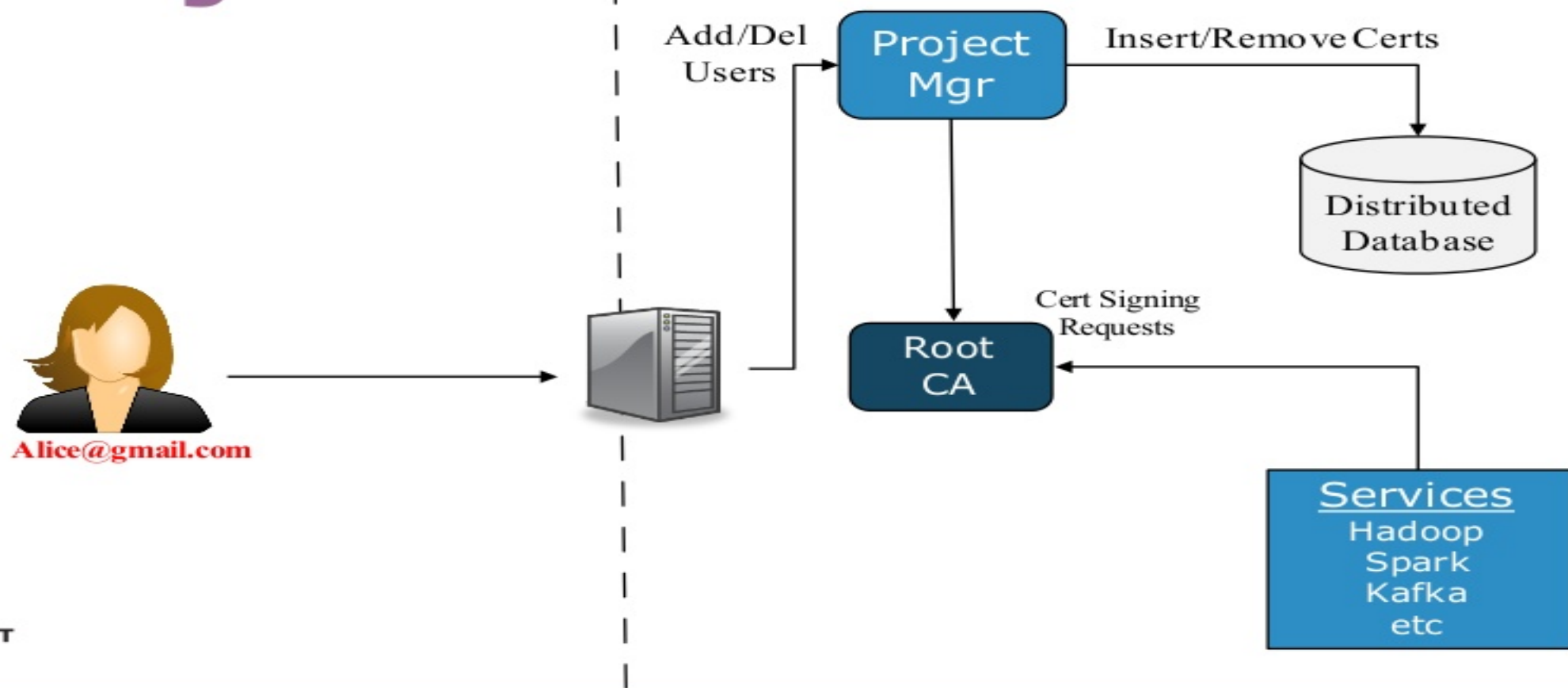
# Dynamic Roles for Hadoop/Kafka



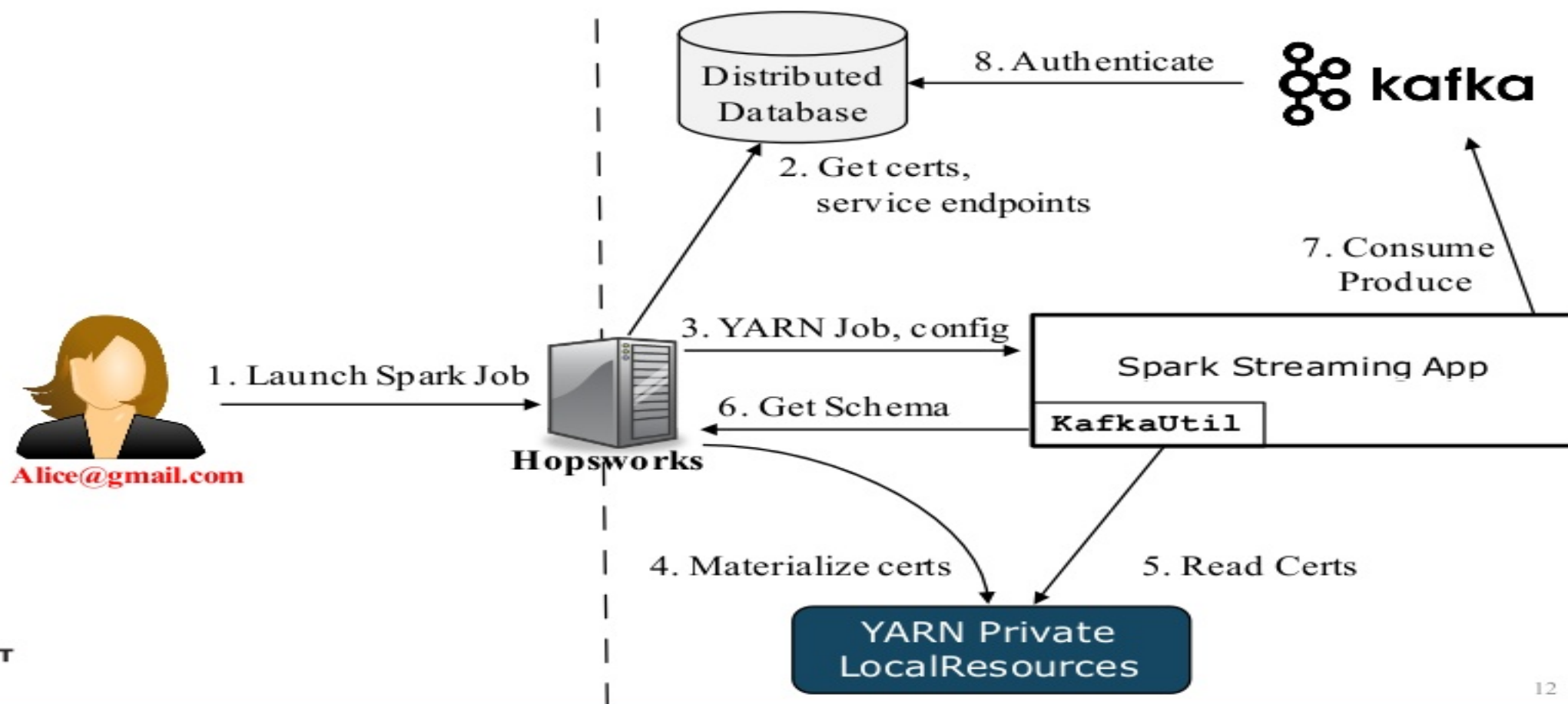
# Look Ma, No Kerberos!

- For each project, a user is issued with a X.509 certificate, containing the project-specific userID.
  - Inspired by Netflix' BLESS system.
- Services are also issued with X.509 certificates.
  - Both user and service certs are signed with the same CA.
  - Services extract the userID from RPCs to identify the caller.

# Project-User Certificates



# Spark Streaming on YARN with Hopsworks



# Spark Stream Producer in Secure Kafka

```
SparkConf sparkConf = ...  
JavaSparkContext jsc = ...
```

1. **Discover:** Schema Registry and Kafka Broker Endpoints
2. **Create:** Kafka Properties file with certs and broker details
3. **Create:** producer using Kafka Properties Developer
4. **Download:** the Schema for the Topic from the Schema Registry
5. **Distribute:** X.509 certs to all hosts on the cluster
6. **Cleanup securely** Operations

```
// write to Kafka
```

# Spark Streaming Producer in Hopsworks

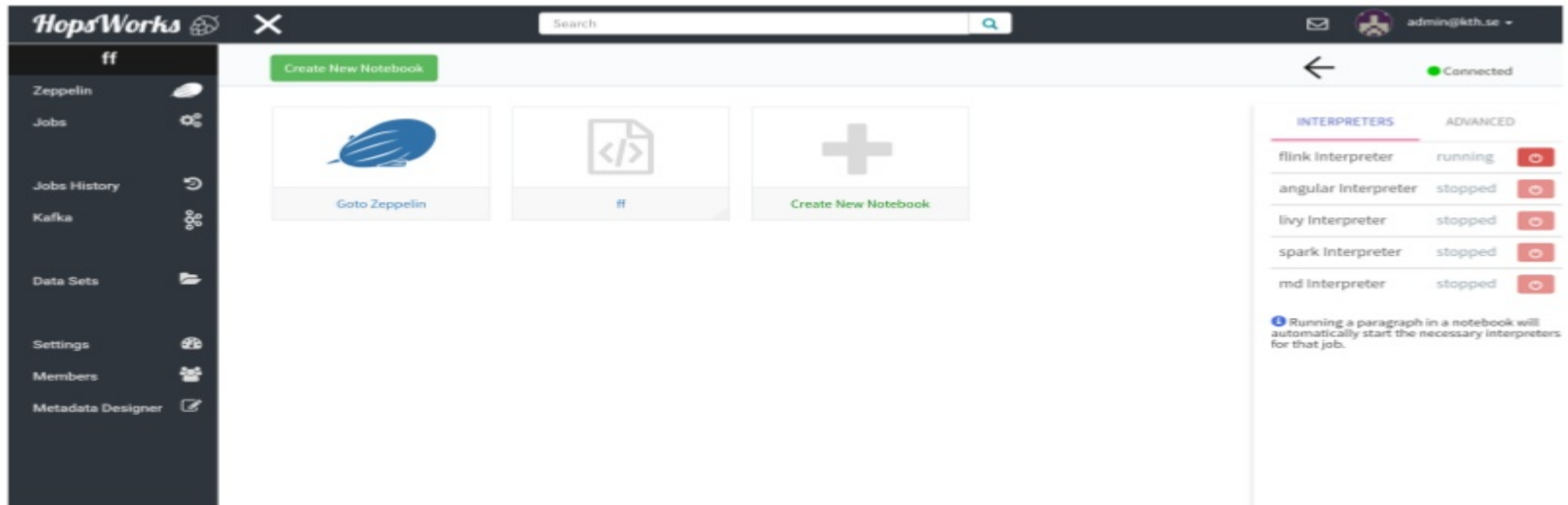
```
List<String> topics = KafkaUtil.getTopics();  
...  
SparkProducer sparkProducer =  
    KafkaUtil.getSparkProducer(topic);  
...  
Map<String, String> message = ...  
sparkProducer.produce(message);  
...  
sparkProducer.close();
```








# Spark Streaming Consumer in Hopsworks

```
JavaStreamingContext jssc = ...  
List<String> topics = KafkaUtil.getTopics();  
...  
SparkConsumer consumer = KafkaUtil.getSparkConsumer(jssc, topics);  
...  
// Avro schema downloaded by framework here  
GenericRecord genericRecord = KafaUtil.getRecordInjections()  
    .get(topic);  
...  
jssc.start();  
jssc.awaitTermination();
```

# Zeppelin Support for Spark/Livy

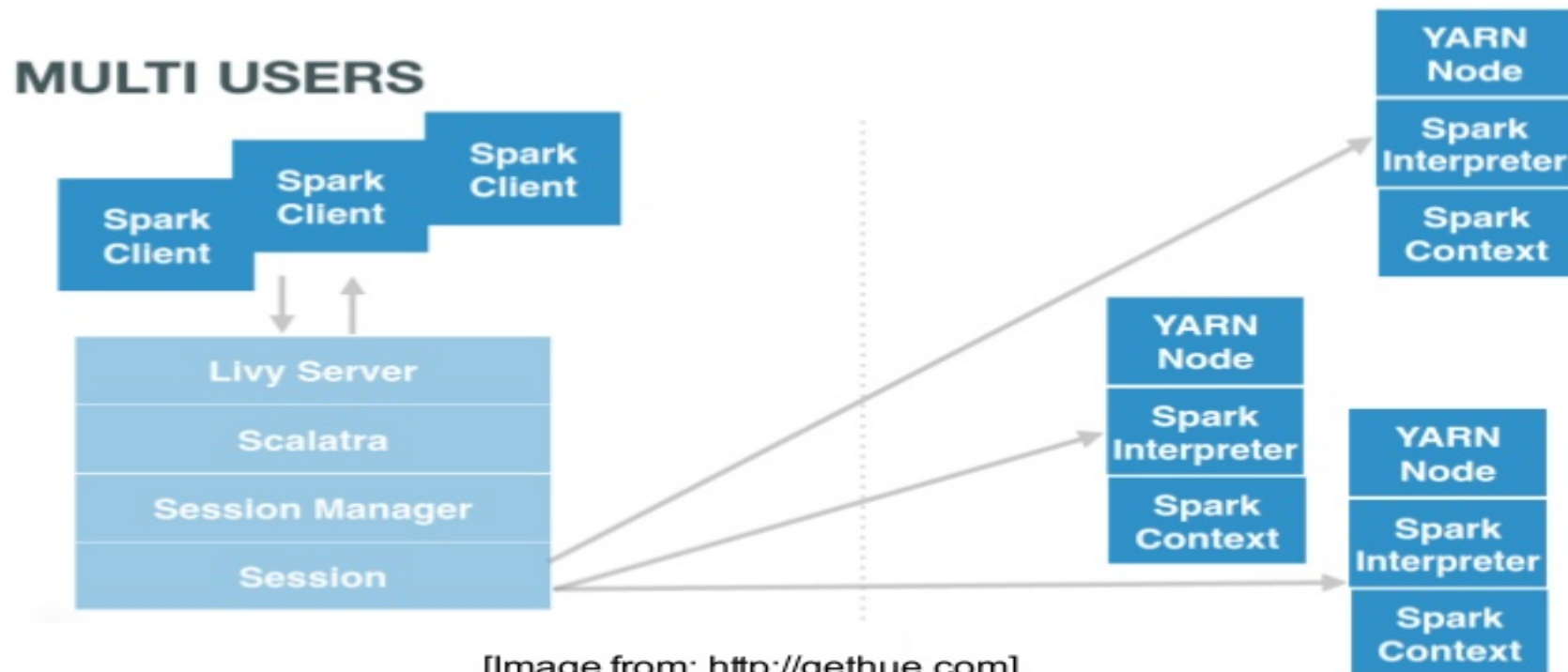


The screenshot displays the HopsWorks web interface. On the left is a dark sidebar with navigation links: Zeppelin, Jobs, Jobs History, Kafka, Data Sets, Settings, Members, and Metadata Designer. The main area features a 'Create New Notebook' button and three tiles: 'Goto Zeppelin', a code editor icon, and another 'Create New Notebook' button. On the right, the 'INTERPRETERS' panel is open, showing a table of active interpreters.

INTERPRETERS	ADVANCED
flink interpreter	running 
angular interpreter	stopped 
livy interpreter	stopped 
spark interpreter	stopped 
md interpreter	stopped 

Running a paragraph in a notebook will automatically start the necessary interpreters for that job.

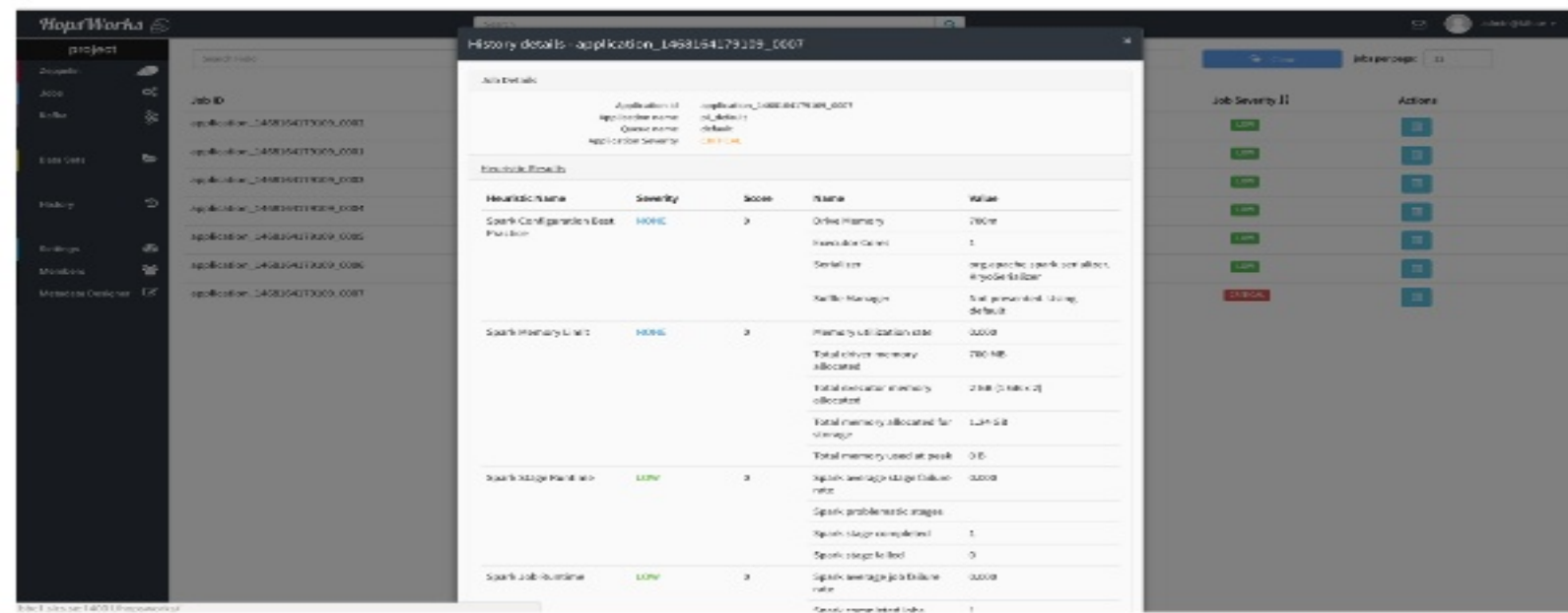
# Livy to launch Spark 2.0 Jobs



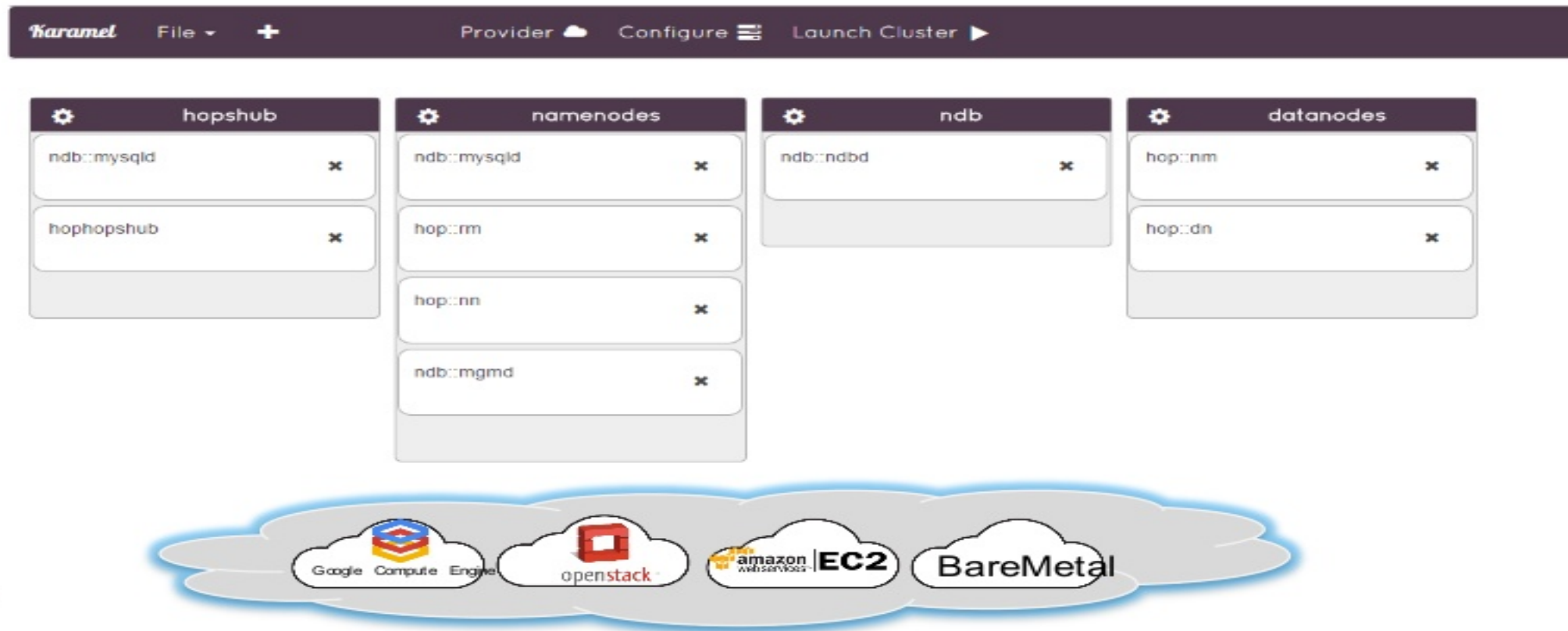
[Image from: <http://gethue.com>]

# Debugging Spark with DrElephant

- Project-specific view of performance/correctness issues for completed Spark Jobs
- Customizable heuristics
- Doesn't show killed jobs



# Karamel/Chef for Automated Installation



# Demo



# Summary

- Hopsworks provides first-class support for Spark-as-a-Service
  - Streaming or Batch Jobs
  - Zeppelin Notebooks
- Hopworks simplifies writing secure SparkStreaming applications with Kafka



Hops

[Hadoop For Humans]

<http://github.com/hopshadoop>

# Hops Team

**Active:** Jim Dowling, Seif Haridi, Tor Björn Minde, Gautier Berthou, Salman Niazi, Mahmoud Ismail, Theofilos Kakantousis, Konstantin Popov, Antonios Kouzoupis, Ermias Gebremeskel.

**Alumni:** Vasileios Giannokostas, Johan Svedlund Nordström, Rizvi Hasan, Paul Mälzer, Bram Leenders, Juan Roca, Misganu Dessalegn, K “Sri” Srijeeyanthan, Jude D’Souza, Alberto Lorente, Andre Moré, Ali Gholami, Davis Jaunzems, Stig Viaene, Hooman Peiro, Evangelos Savvidis, Steffen Grohsschmiedt, Qi Qi, Gayana Chandrasekara, Nikolaos Stanogias, Ioannis Kerkinos, Peter Buechler, Pushparaj Motamari, Hamid Afzali, Wasif Malik, Lalith Suresh, Mariano Valles, Ying Lieu.



# THANK YOU.

[www.hops.io](http://www.hops.io)

