

# 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

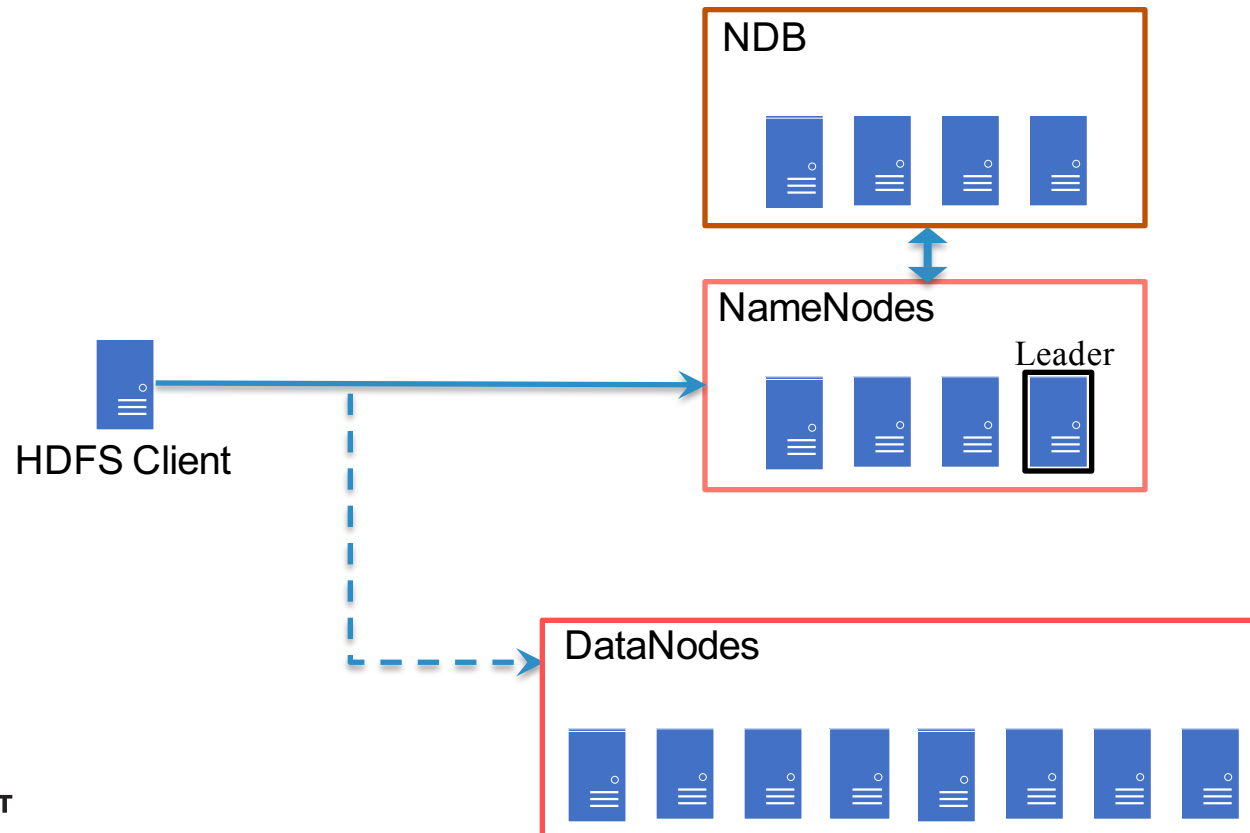


# Spark-as-a-Service in Sweden

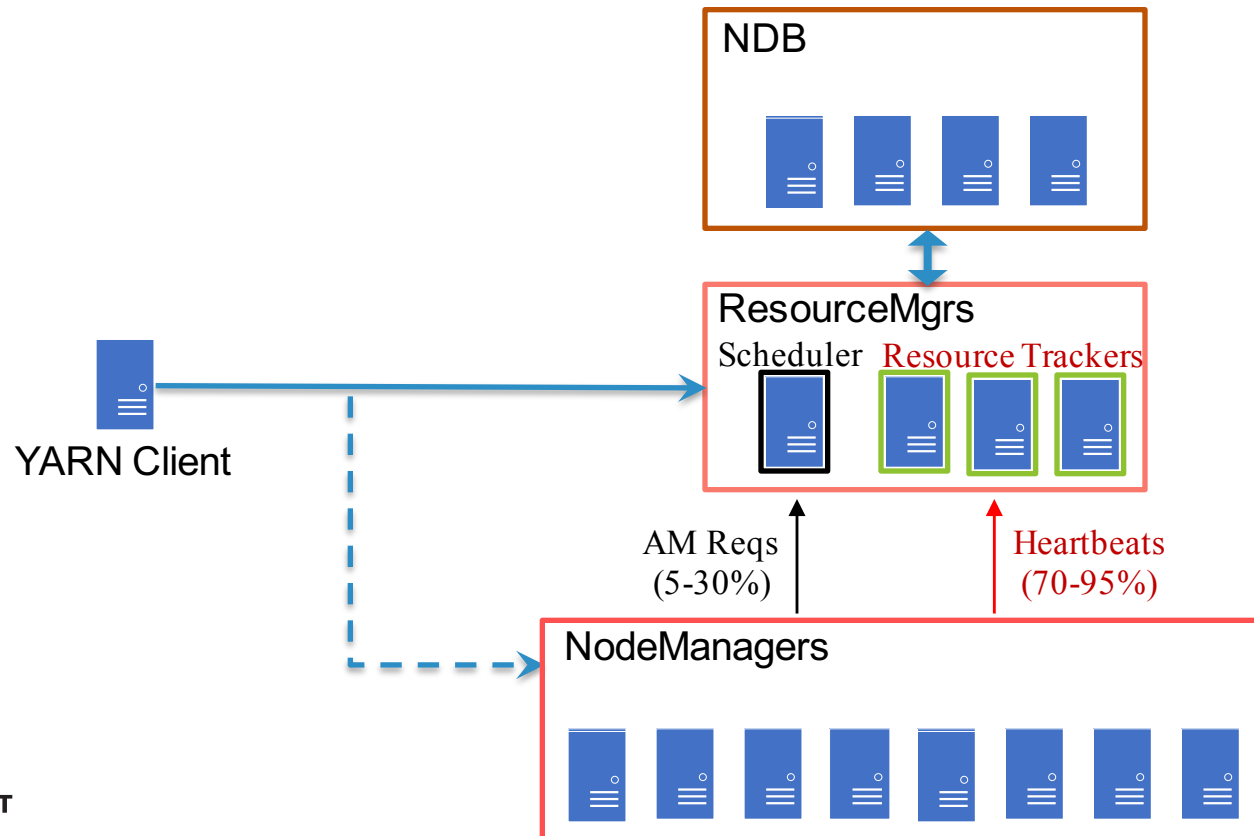
- **SICS ICE:** datacenter research and test environment
- **Hopsworks:** Spark/Kafka/Flink/Hadoop-as-a-service
  - Built on Hops Hadoop ([www.hops.io](http://www.hops.io))
  - Over 100 active users
  - Spark the platform of choice



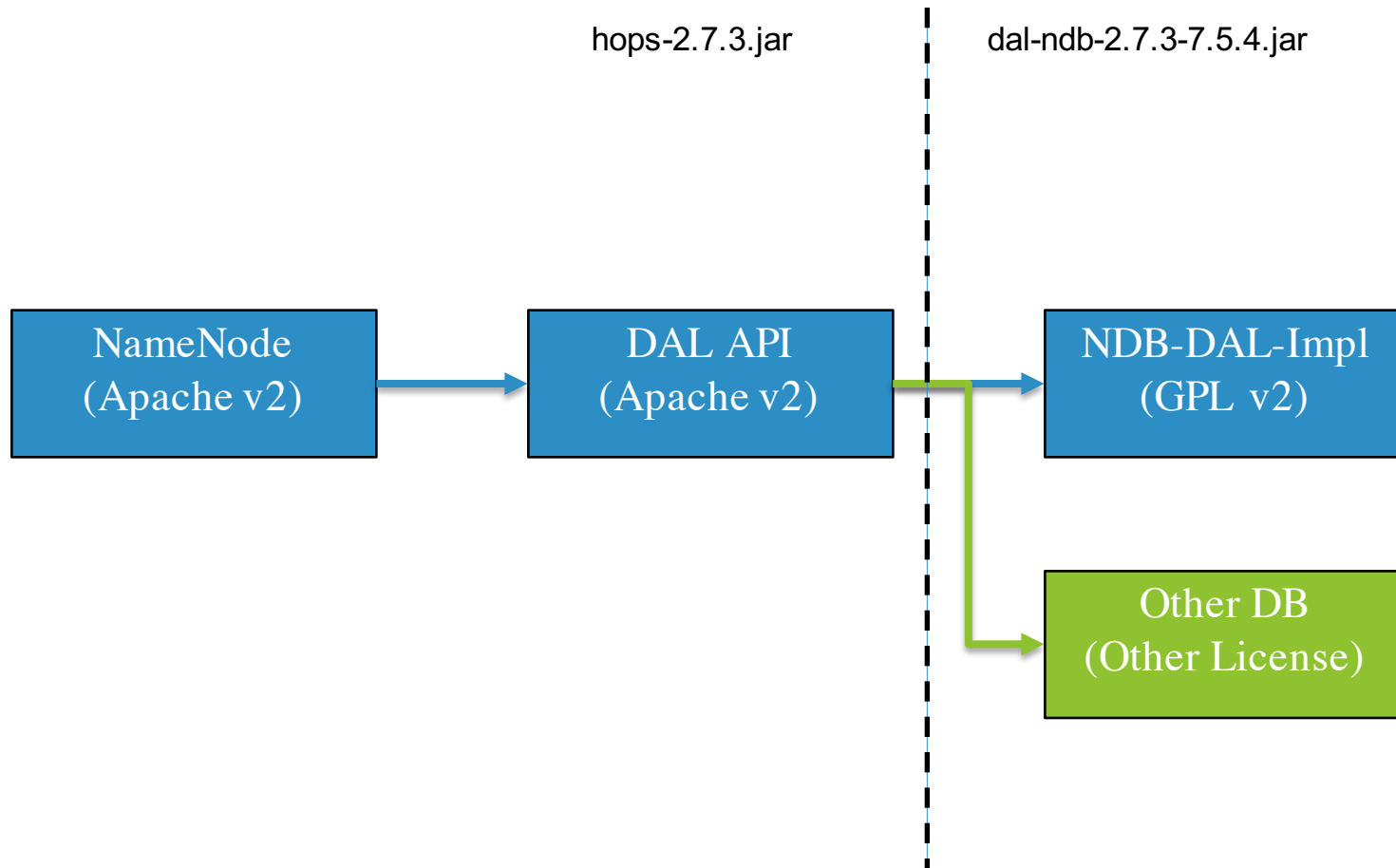
# HopsFS Architecture



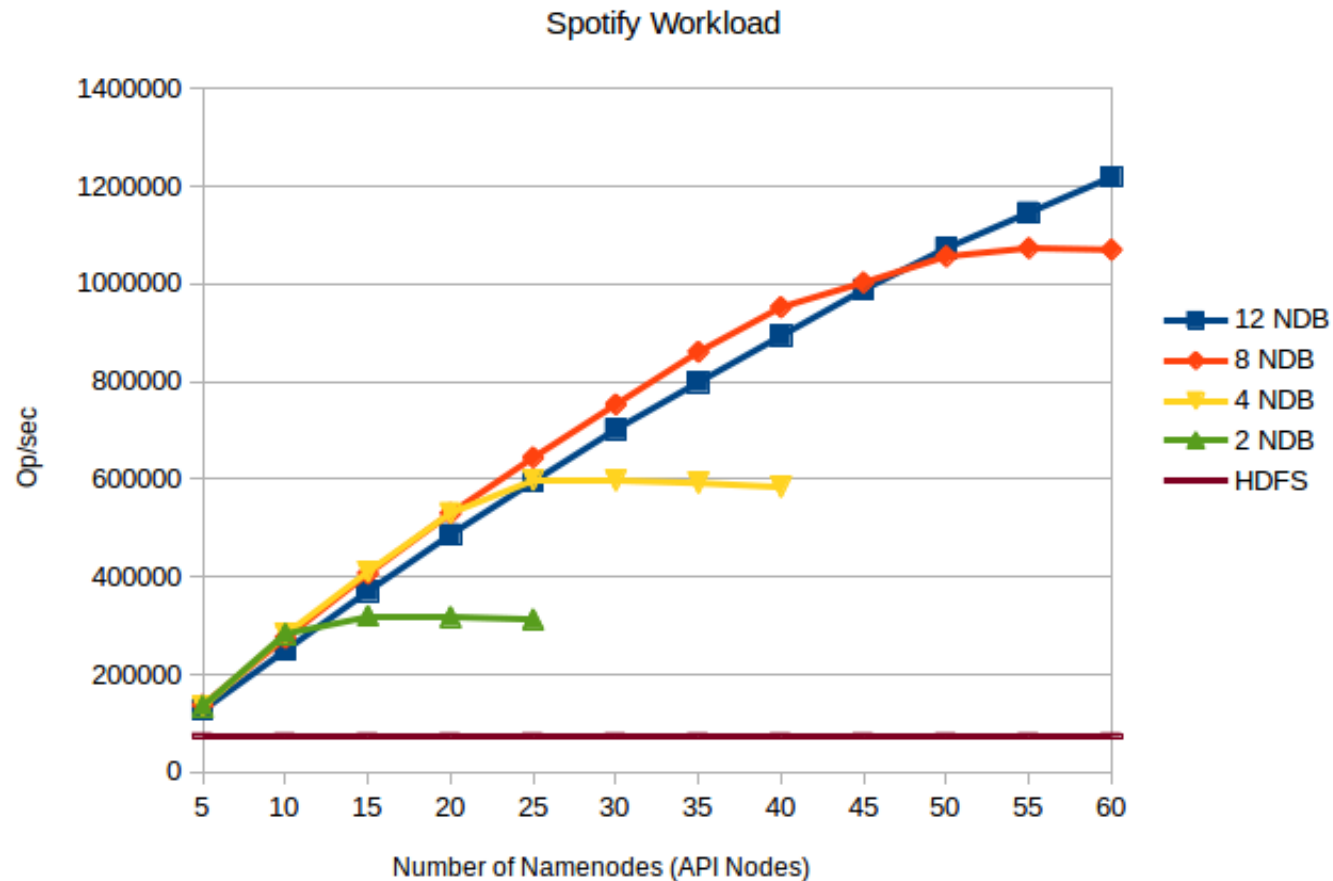
# Hops-YARN Architecture



# Pluggable DB: Data Abstraction Layer



# HopsFS Throughput vs Apache HDFS

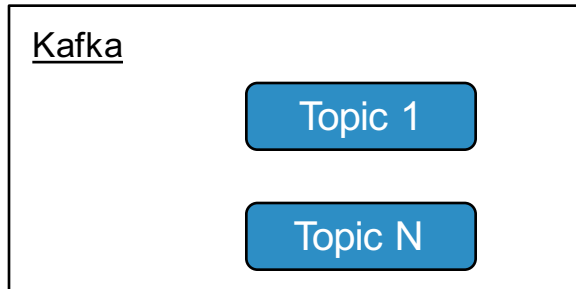
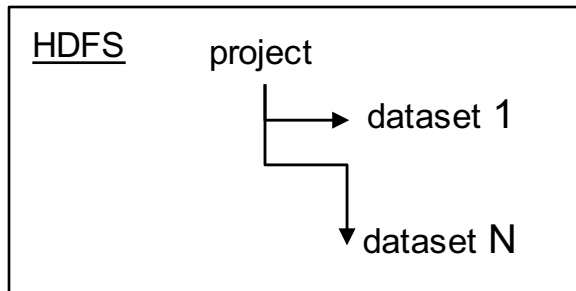
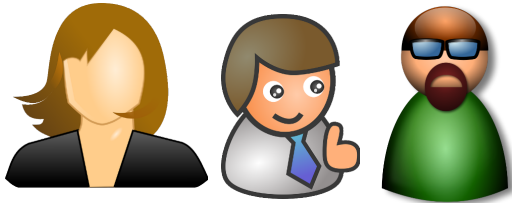


NDB Setup: Nodes using Xeon E5-2620 2.40GHz Processors and 10GbE.  
NameNodes: Xeon E5-2620 2.40GHz Processors machines and 10GbE.



# HOPSWORKS

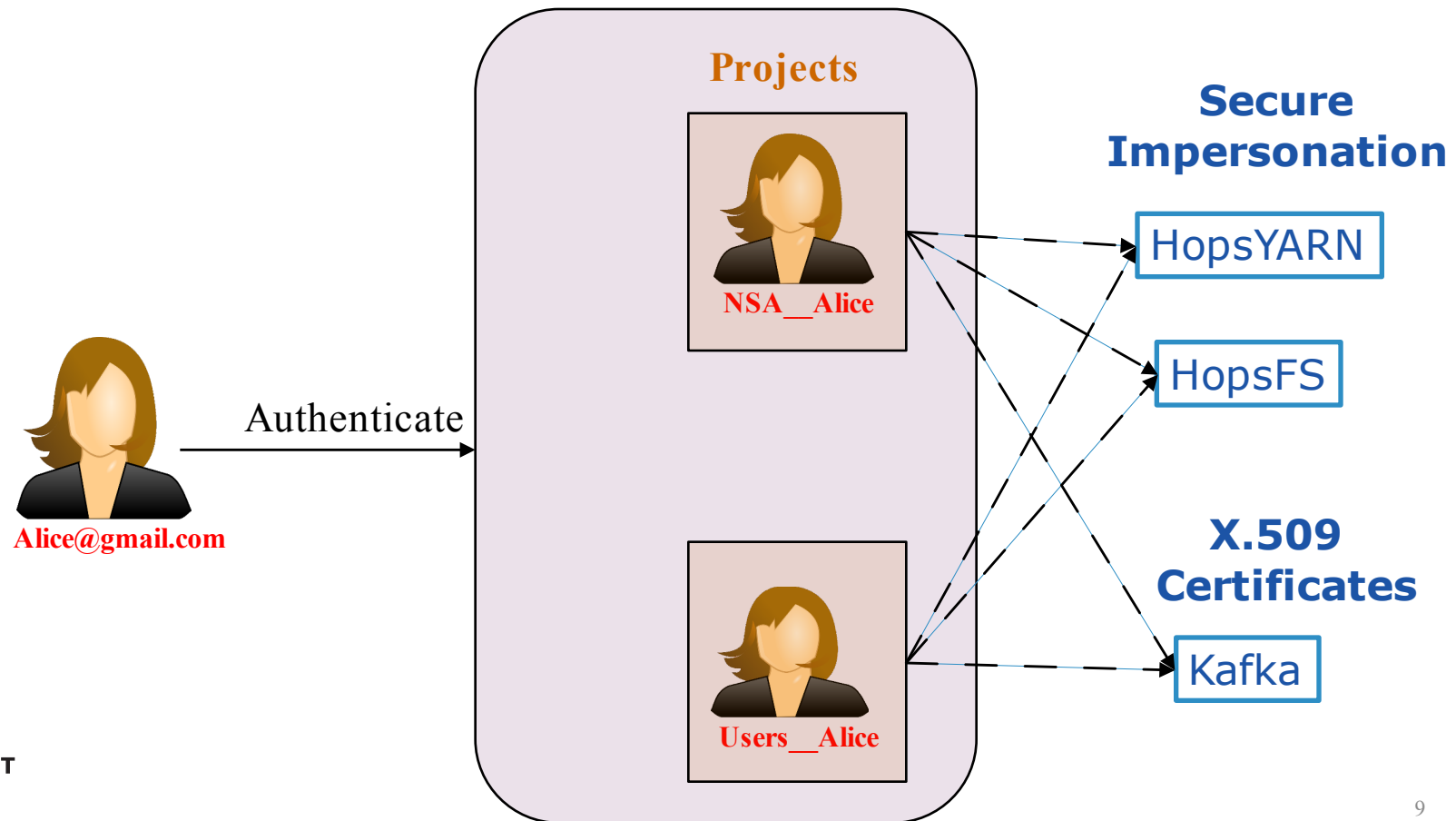
# Project-Based Multi-Tenancy



- 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



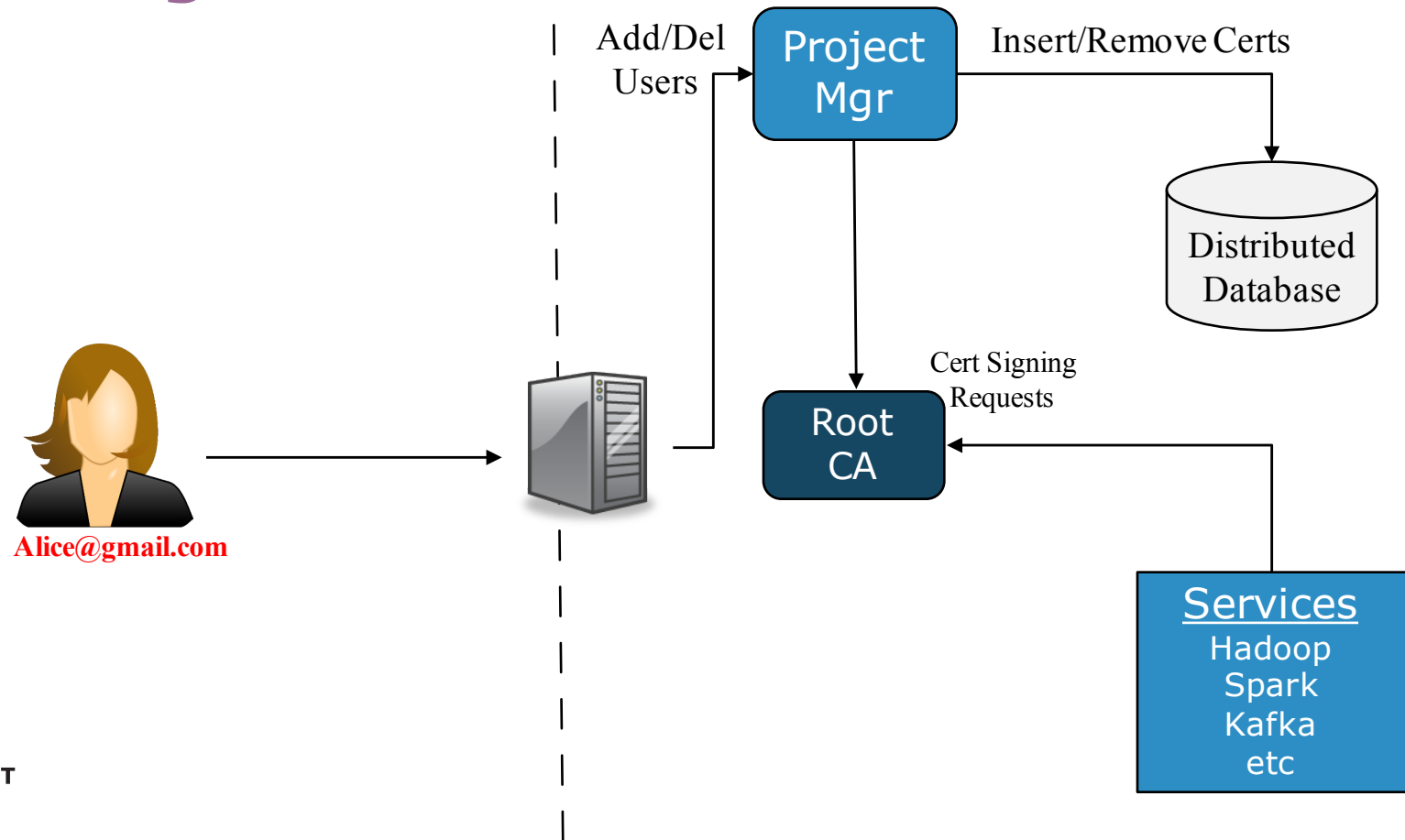
# Dynamic Roles for Hadoop/Kafka



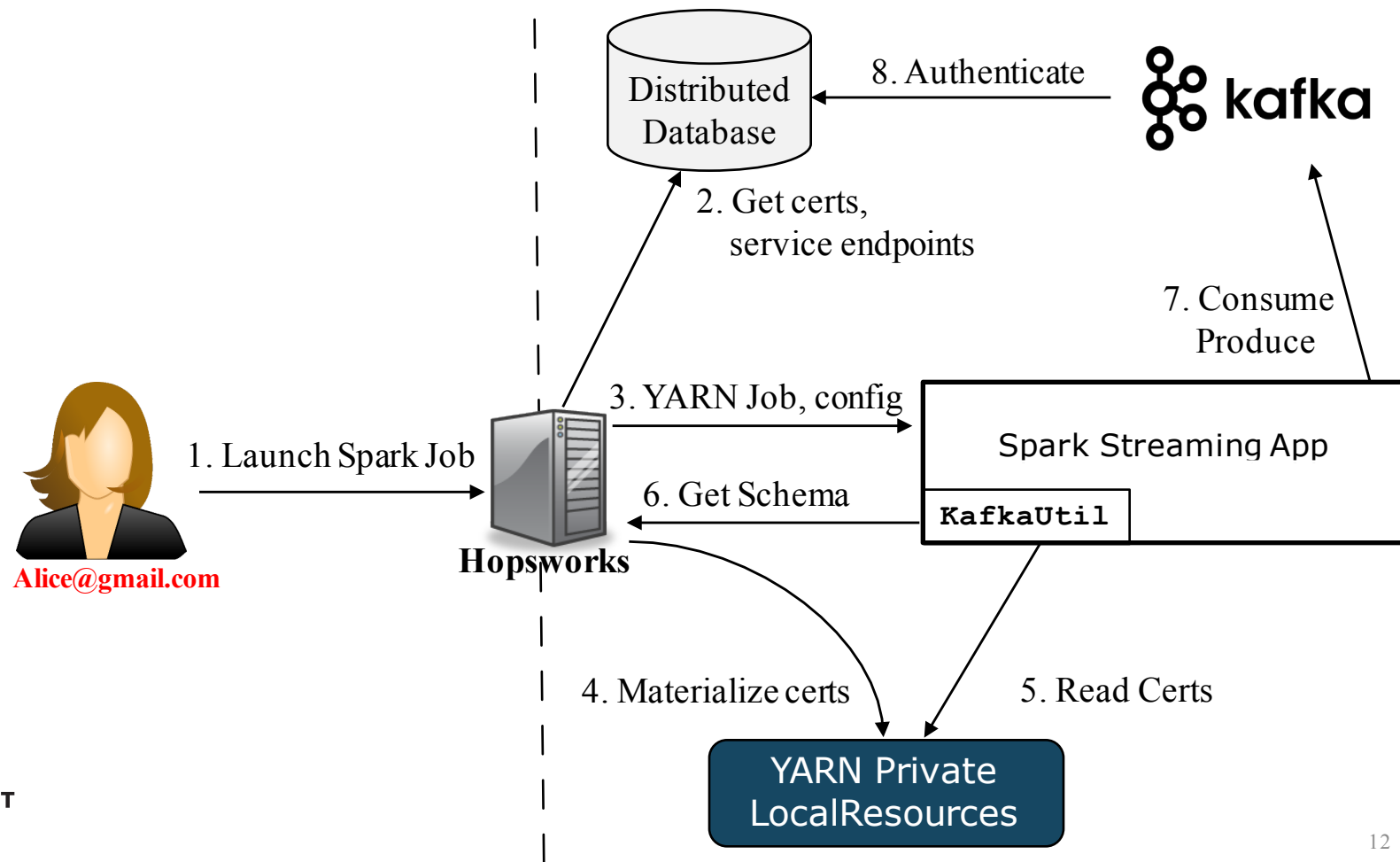
# Look Ma, No Kerberos!

- 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

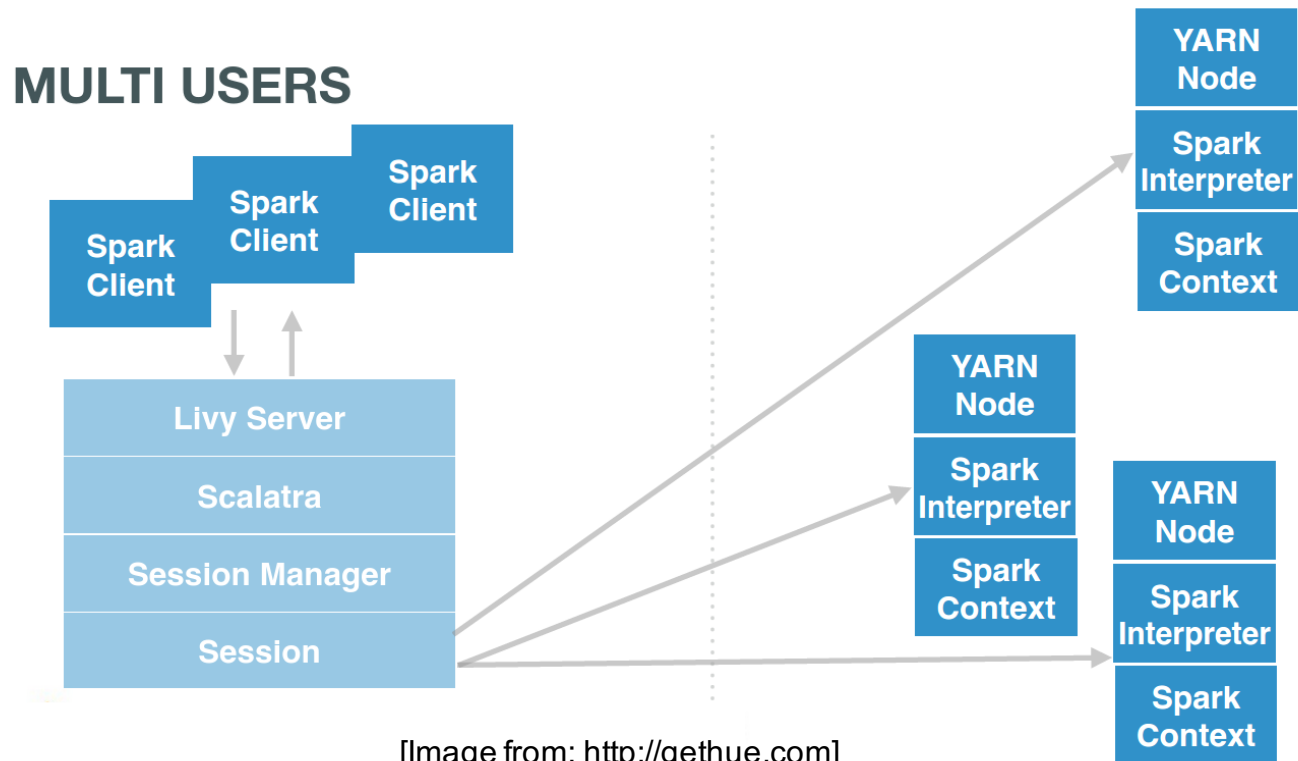
The screenshot displays the HopsWorks Zeppelin web interface. The top navigation bar includes the HopsWorks logo, a search bar, and a user profile for 'admin@kth.se'. The left sidebar lists various features: Zeppelin, Jobs, Jobs History, Kafka, Data Sets, Settings, Members, and Metadata Designer. The main workspace area contains three buttons: 'Goto Zeppelin' (with a blue leaf icon), 'ff' (with a code icon), and 'Create New Notebook' (with a plus icon). A green 'Create New Notebook' button is also located at the top of the workspace. On the right side, the 'INTERPRETERS' panel is visible, showing a list of interpreters and their status.

| 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



# Debugging Spark with DrElephant

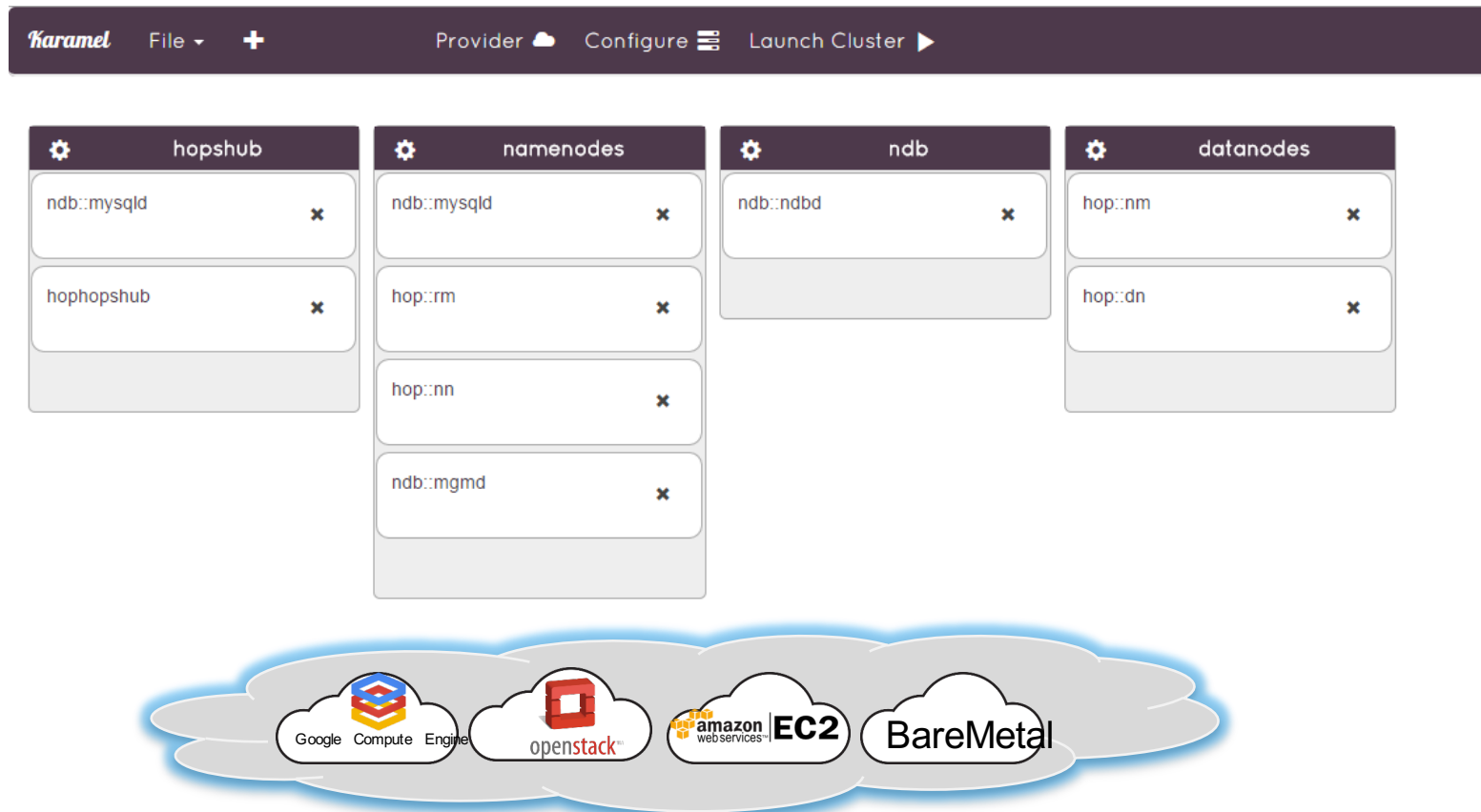
- Project-specific view of performance/correctness issues for completed Spark Jobs
- Customizable heuristics
- Doesn't show killed jobs

The screenshot displays the HopsWorks DrElephant interface. On the left, a sidebar shows navigation options: project, Zppelin, Jobs, Kafka, Data Sets, History, Settings, Members, and Metadata Designer. The main panel shows 'History details - application\_1468164179109\_0007'. It includes a 'Job Details' section with application information and a 'Heuristic Results' table.

| Heuristic Name                    | Severity | Score | Name                               | Value                                      |
|-----------------------------------|----------|-------|------------------------------------|--------------------------------------------|
| Spark Configuration Best Practice | NONE     | 0     | Drive Memory                       | 700m                                       |
|                                   |          |       | Executor Cores                     | 1                                          |
|                                   |          |       | Serializer                         | org.apache.spark.serializer.KryoSerializer |
|                                   |          |       | Shuffle Manager                    | Not presented. Using default               |
| Spark Memory Limit                | NONE     | 0     | Memory utilization rate            | 0.000                                      |
|                                   |          |       | Total driver memory allocated      | 700 MB                                     |
|                                   |          |       | Total executor memory allocated    | 2 GB (1 GB x 2)                            |
|                                   |          |       | Total memory allocated for storage | 1.24 GB                                    |
|                                   |          |       | Total memory used at peak          | 0 B                                        |
| Spark Stage Runtime               | LOW      | 0     | Spark average stage failure rate   | 0.000                                      |
|                                   |          |       | Spark problematic stages           |                                            |
|                                   |          |       | Spark stage completed              | 1                                          |
|                                   |          |       | Spark stage failed                 | 0                                          |
| Spark Job Runtime                 | LOW      | 0     | Spark average job failure rate     | 0.000                                      |
|                                   |          |       | Spark recommended solve            | 1                                          |

On the right, a 'Job Severity' section shows a list of jobs with severity levels (LOW, CRITICAL) and action buttons.

# Karamel/Chef for Automated Installation



# Demo

# Summary

- Hopsworks provides first-class support for Spark-as-a-Service
  - Streaming or Batch Jobs
  - Zeppelin Notebooks



Hops

[Hadoop For Humans]

- Hopworks simplifies writing secure SparkStreaming applications with Kafka

# 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” Srijeanthan, 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)

