Apache Spark and Jupyter

Machiel Jansen machiel.jansen@surfsara.nl

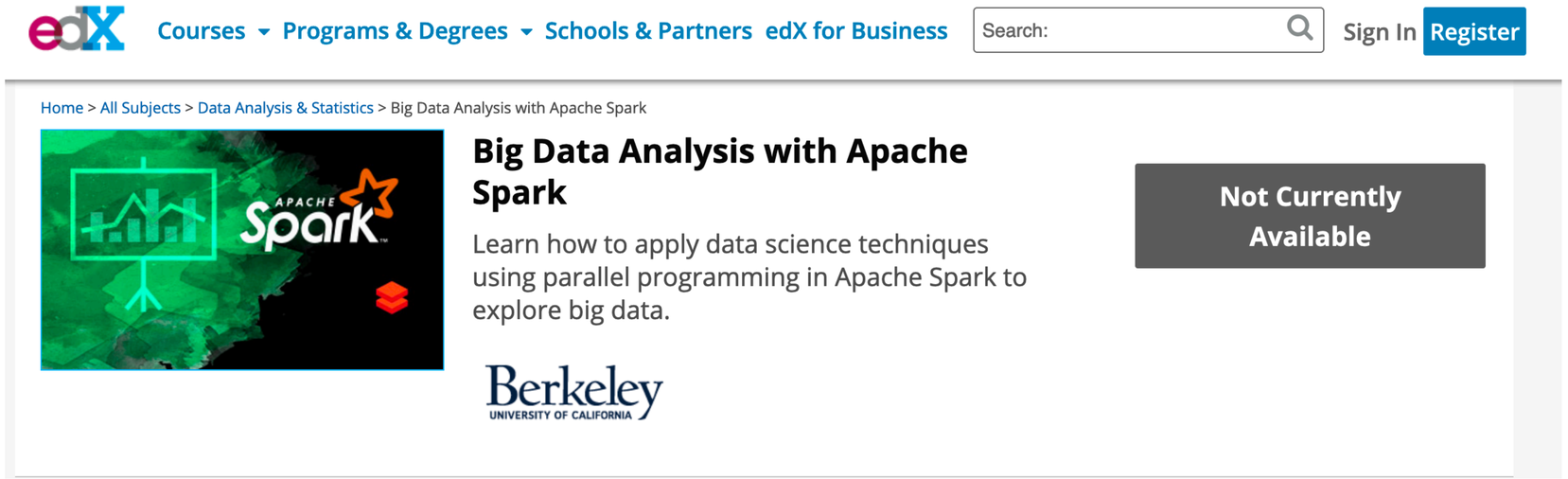
SURF SARA

The next 40(ish) minutes

- Intro Jupyter at SURFsara
- Jupyter and Docker (Stacks)
- What is Spark and why?
- How to use it from within Jupyter

Jupyter at SURFsara

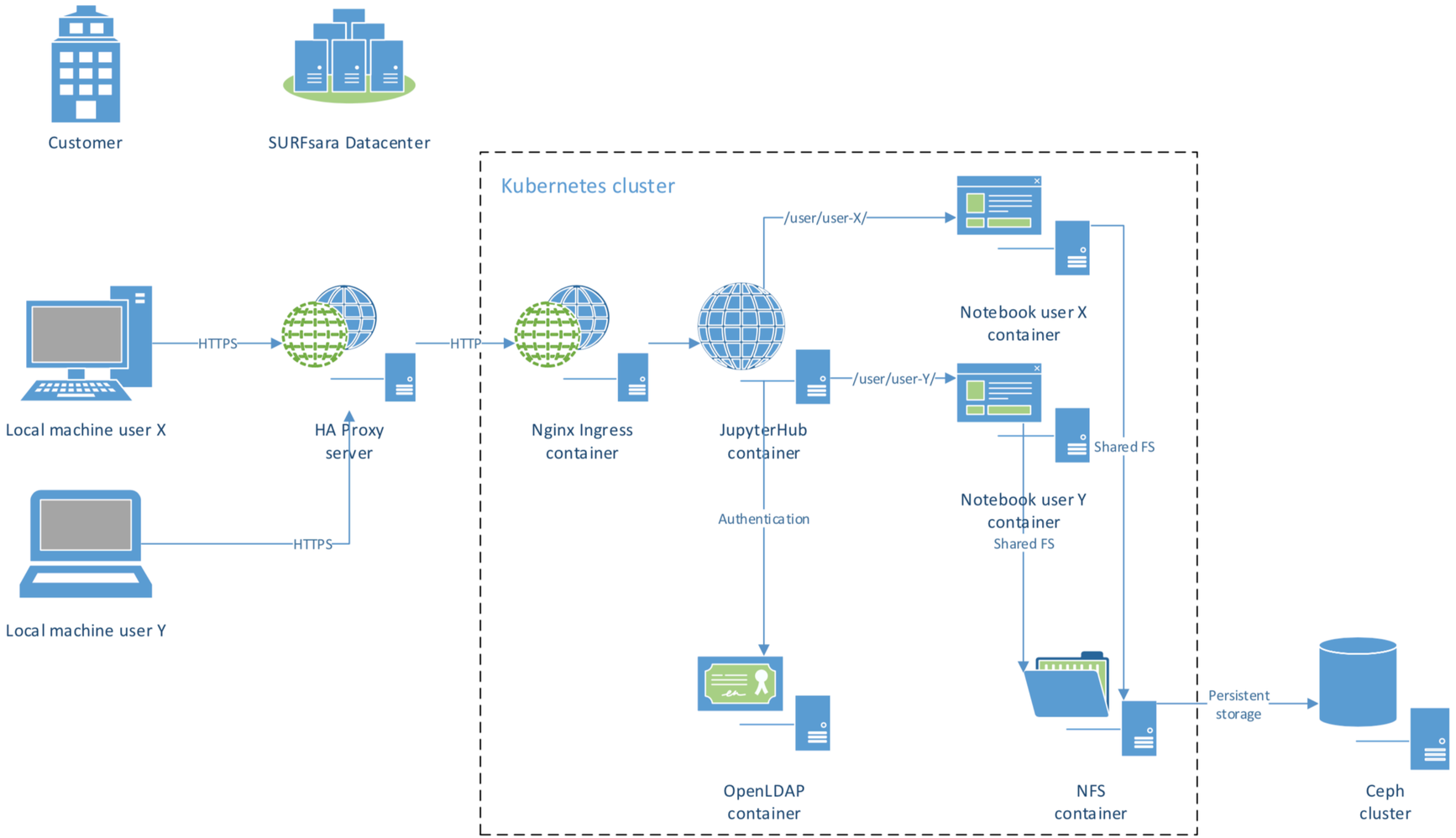
- Used in workshops for Apache Spark
- After EDX MOOC on Spark by Berkeley Uni gave us the idea



Jupyter at SURFsara

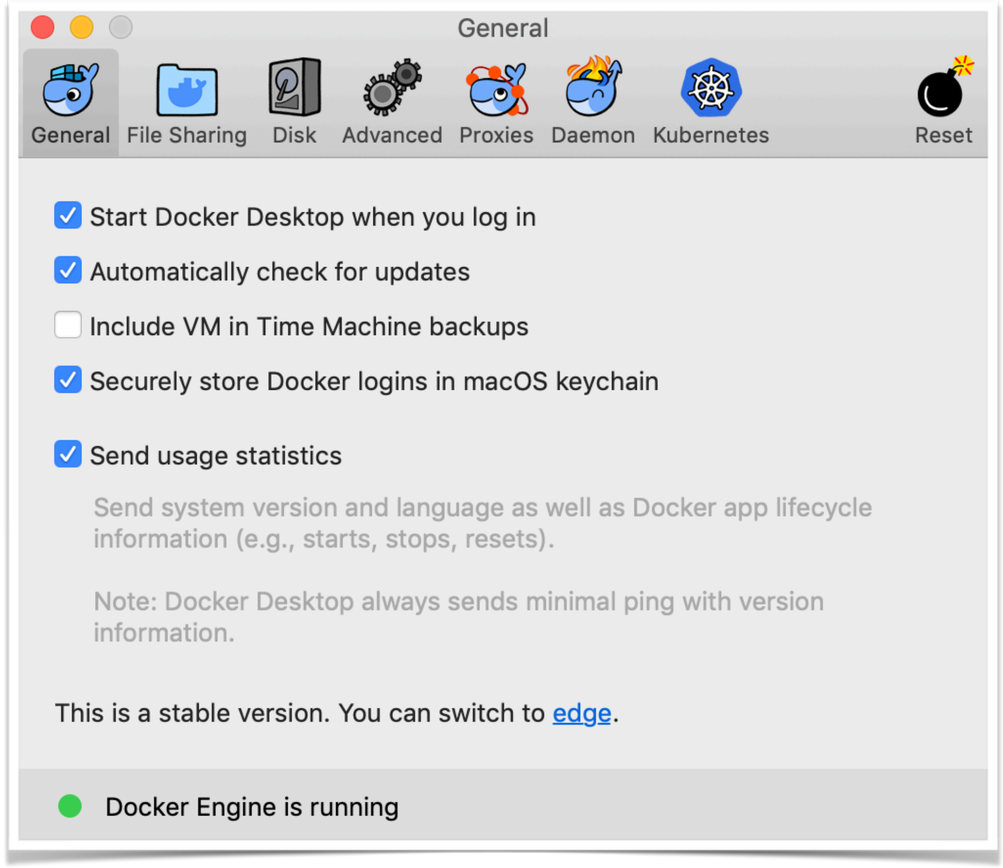
- JupyterHub for education and data science
- Multi user, coming with LDAP and use management portal & API
- Using Kubernetes spawner
- CEPH as persistent storage
- Currently experiments on SURF ResearchDrive and AWS

Kubernetes spawner



Jupyter in Docker

- Easy to share, to run elsewhere
  - (MacOS, Linux, Windows)
- Easy to scale
- Try different images



SURF SARA

Jupyter Docker Stacks

Jupyter Docker Stacks are a set of ready-to-run Docker images containing Jupyter applications and interactive computing tools. You can use a stack image to do any of the following (and more):

- Start a personal Jupyter Notebook server in a local Docker container
- Run JupyterLab servers for a team using JupyterHub
- Write your own project Dockerfile

Quick Start

You can try a recent build of the jupyter/base-notebook image on mybinder.org by simply clicking the preceding link. Otherwise, the two examples below may help you get started if you have Docker installed, know which Docker image you want to use, and want to launch a single Jupyter Notebook server in a container.

The other pages in this documentation describe additional uses and features in detail.

Example 1: This command pulls the `jupyter/scipy-notebook` image tagged `17aba6048f44` from Docker Hub if it is not already present on the local host. It then starts a container running a Jupyter Notebook server and exposes the server on host port 8888. The server logs appear in the terminal. Visiting `http://<hostname>:8888/?token=<token>` in a browser loads the Jupyter Notebook dashboard page, where `hostname` is the name of the computer running docker and `token` is the secret token printed in the console. The container remains intact for restart after the notebook server exits.:

```
docker run -p 8888:8888 jupyter/scipy-notebook:17aba6048f44
```

Apache Spark

Specific Docker Image Options

- `-p 4040:4040` - The `jupyter/pyspark-notebook` and `jupyter/all-spark-notebook` images open SparkUI (Spark Monitoring and Instrumentation UI) at default port `4040`, this option map `4040` port inside docker container to `4040` port on host machine . Note every new spark context that is created is put onto an incrementing port (ie. 4040, 4041, 4042, etc.), and it might be necessary to open multiple ports. For example: `docker run -d -p 8888:8888 -p 4040:4040 -p 4041:4041 jupyter/pyspark-notebook`

Usage Examples

The `jupyter/pyspark-notebook` and `jupyter/all-spark-notebook` images support the use of Apache Spark in Python, R, and Scala notebooks. The following sections provide some examples of how to get started using them.

Using Spark Local Mode

Spark local mode is useful for experimentation on small data when you do not have a Spark cluster available.

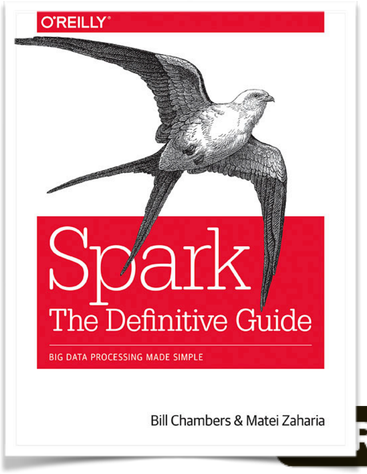
In a Python Notebook

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("SimpleApp").getOrCreate()
# do something to prove it works
spark.sql('SELECT "Test" as c1').show()
```

```
docker run -d -p 8888:8888 -p 4040:4040 -v /Users/mgjansen/jupyter/mount:/home/jovyan/work jupyter/all-spark-notebook
```

What is Apache Spark?

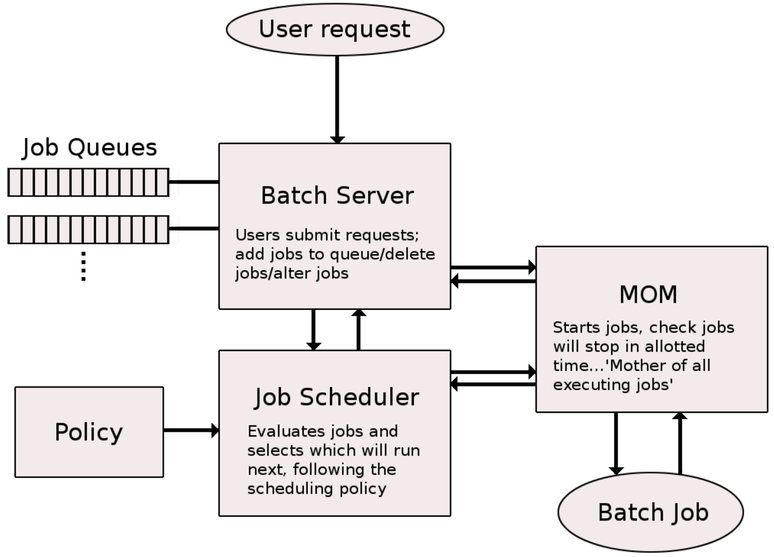
- Spark is a software development framework - not an environment in which you can (easily) run binary (unix) programs
- It provides a simplified (limited) and therefore easier way of writing distributed data intensive applications
- Spark runs on commodity hardware or in cloud environments



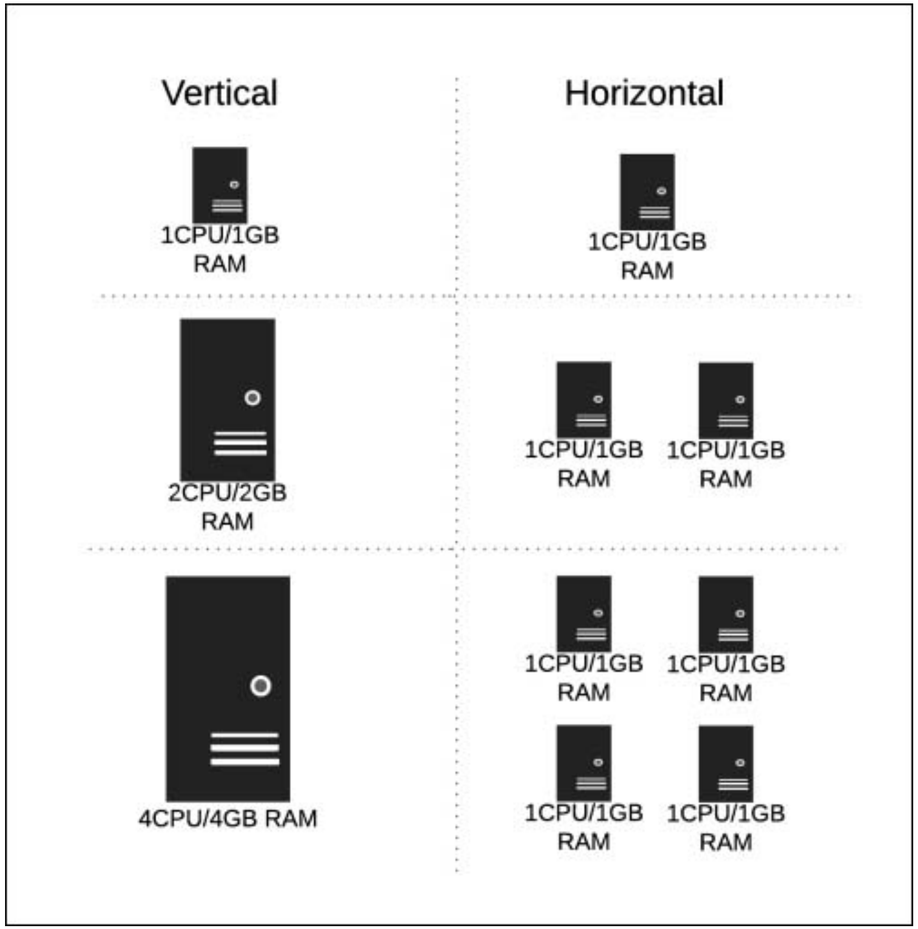
SARA

Scaling “traditional” applications

- Now the one running the application needs to:
  - Distribute and split data
  - Handle faults and errors inherent with scale
  - Submit and track applications
  - Use files or relational database (fixed schema's)



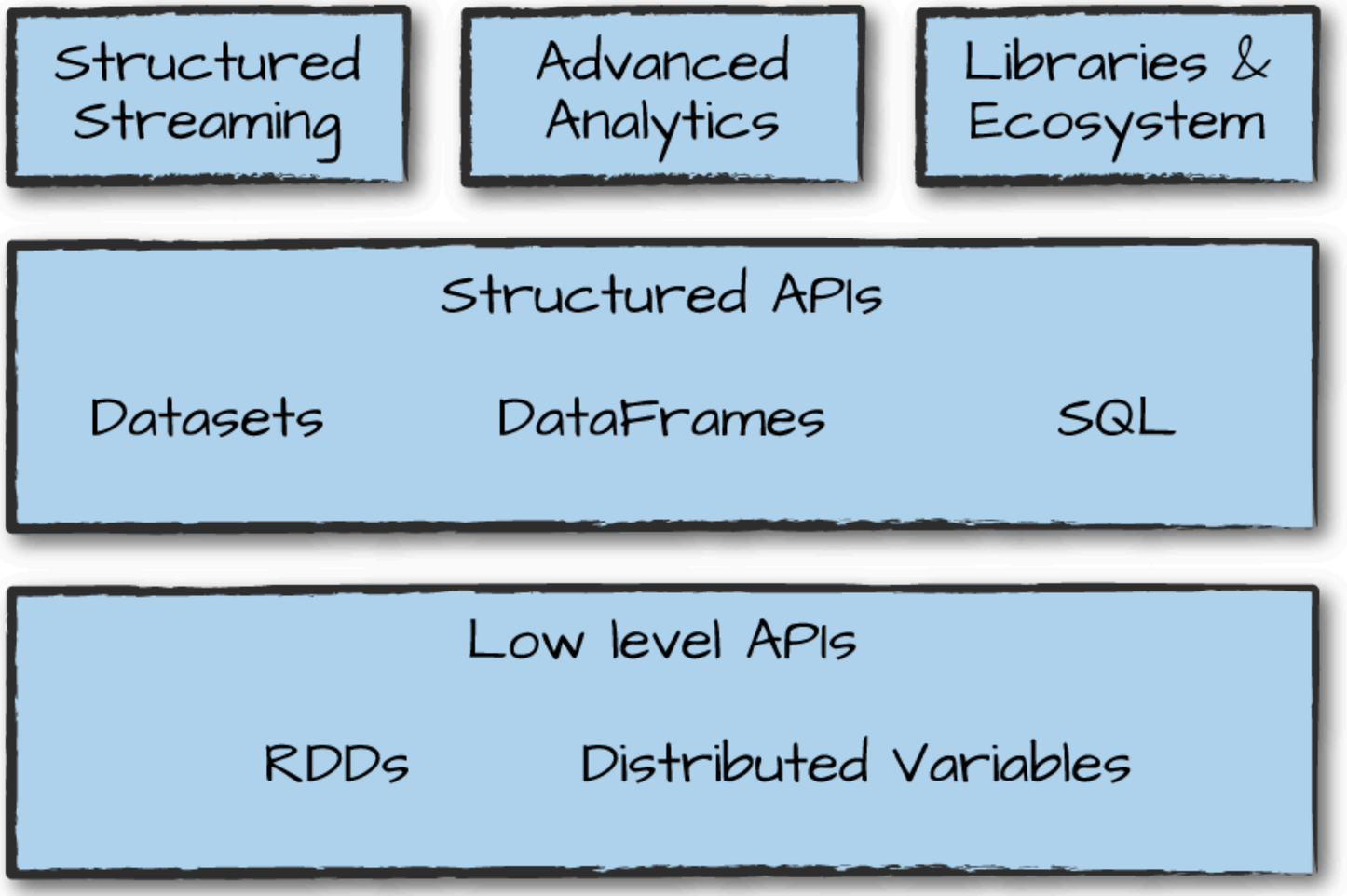
Scaling up or out



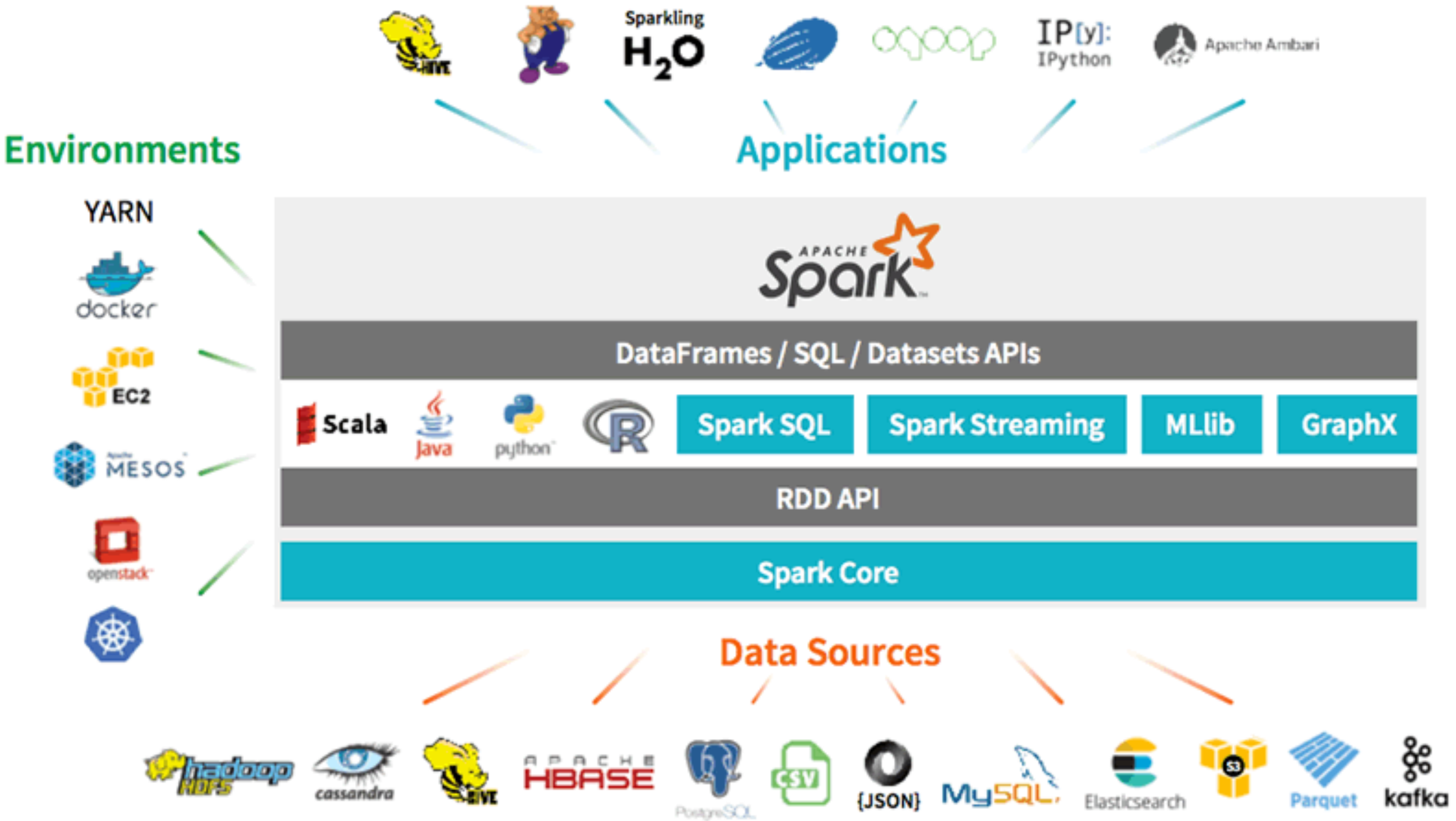
Spark: A General Framework

- Spark aims to generalise MapReduce to support new applications with a more efficient engine, and simpler for the end users.
- Write programs in terms of distributed datasets and operations on them.
- Accessible from multiple programming languages:
  - Scala
  - Java
  - Python
  - R (only via dataframes)

Spark components



Spark extras



Two main API's

- Low level

RDDs - Python, Scala , Java - no R support

- Structured API

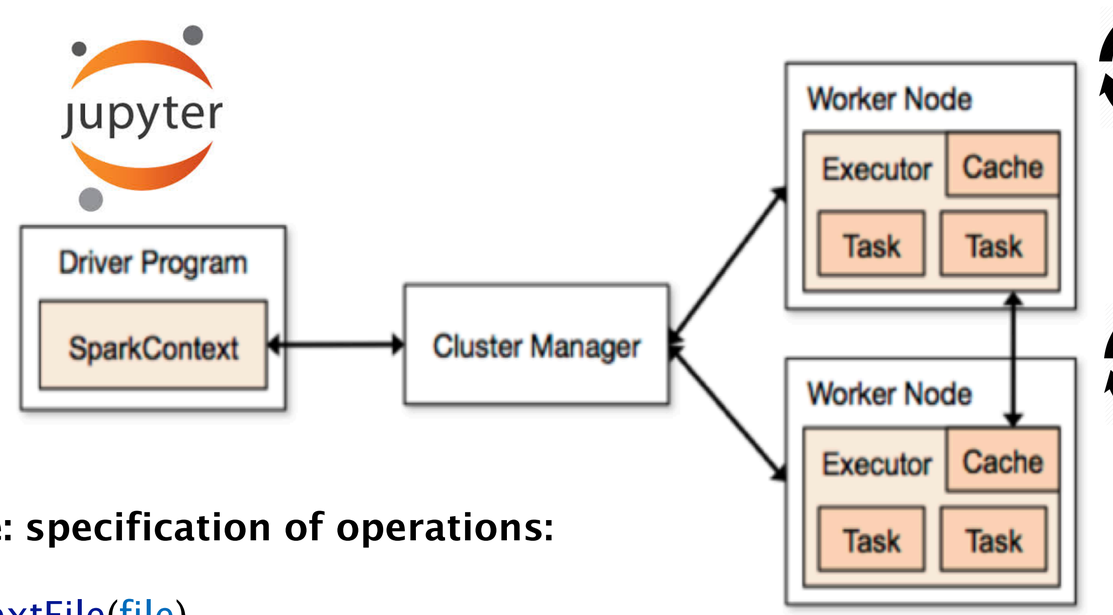
DataFrames - higher level - R support, SQL and ML

Spark modes

SparkContext contains information about the cluster and is the linking pin between your code and the cluster.

- Local mode: single machine, using multiple cores. For testing and training purposes.
- Cluster mode: dedicated Spark cluster (also on clouds)
- Hadoop/cluster mode: Use Hadoop YARN to deploy cluster

An Executing Application



Executor: execution of operations:

On local lines
s.split()

Executor: execution of operations:

On local lines
s.split()

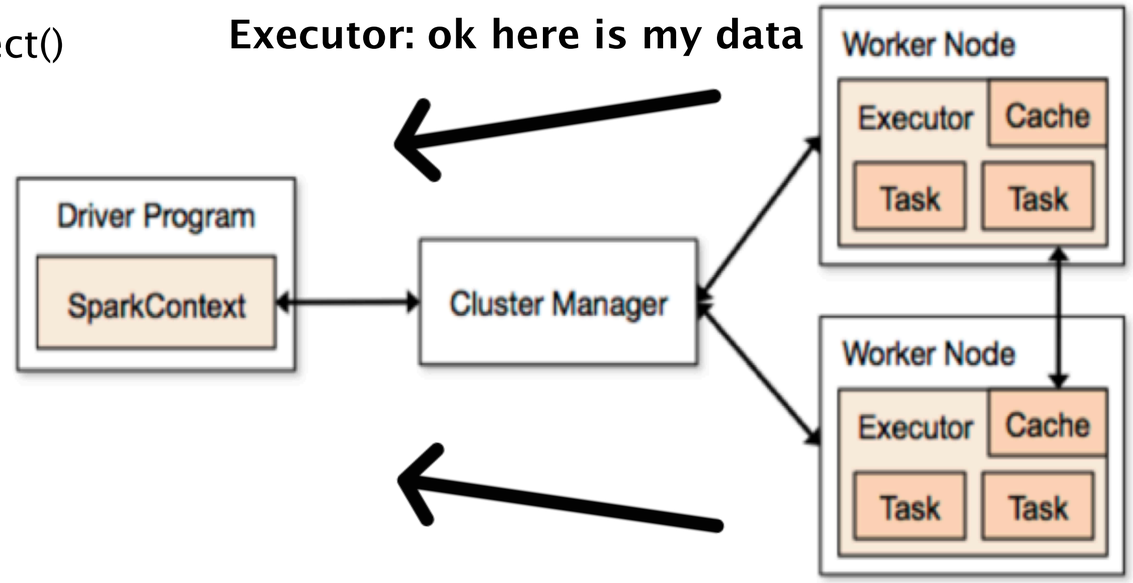
Driver code: specification of operations:

```
lines  = sc.textFile(file)
words  = lines.flatMap(lambda s: s.split())
```

An Executing Application

Driver code: some actions send data to driver

```
lines  = sc.textFile(file)
lines_local = lines.collect()
```



Resilient Distributed Dataset (RDD)

- Abstraction for a collection of objects/elements/records
- Spread over many machines
- Built through parallel transformations
- Immutable

DataFrames

- DataFrame is a distributed collection of data organized into named columns. Conceptually equivalent to a table in a relational database or a dataframe in R/Python.

- DataFrames can be constructed from a wide array of sources such as: structured data files, external databases, or existing RDDs.