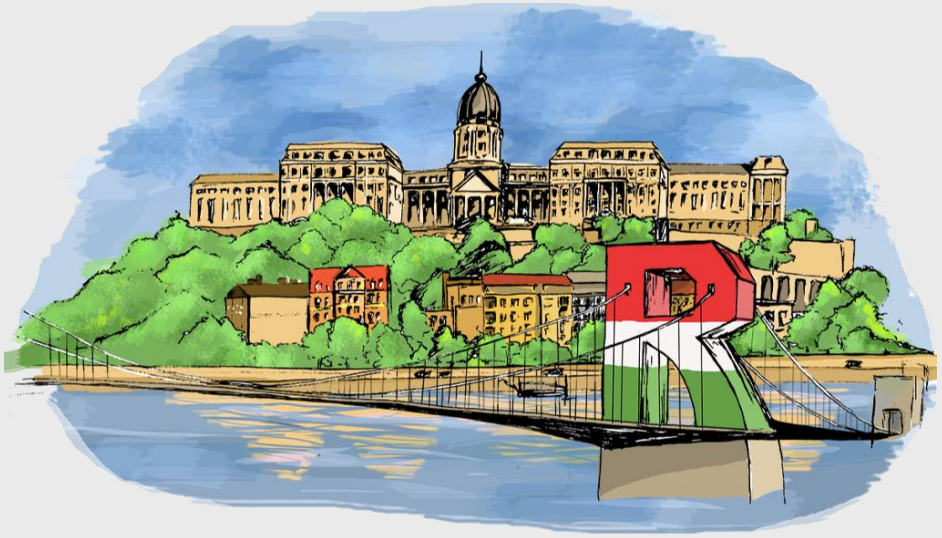
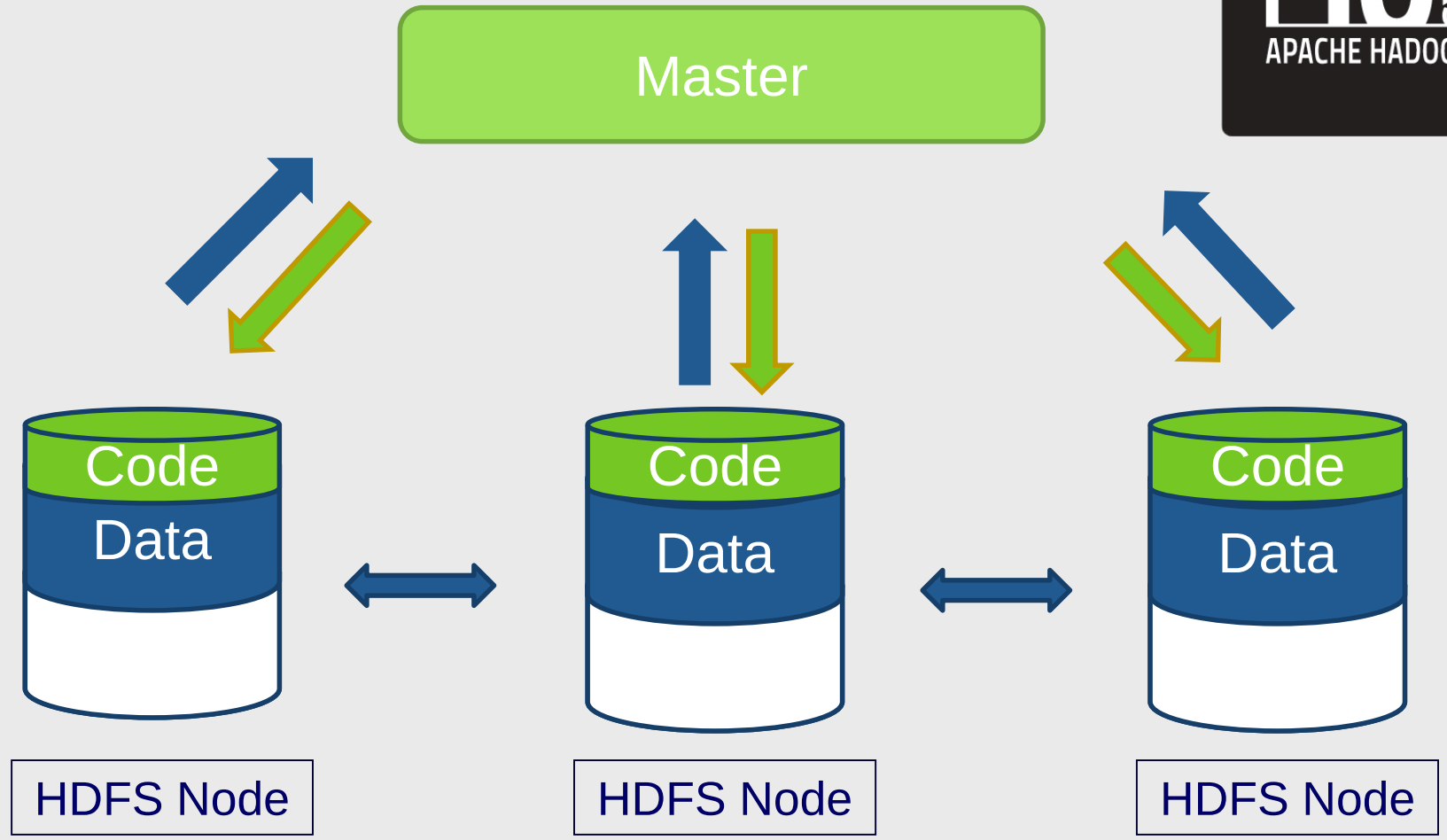


The Big Connection - R and Big Data

Bence Arató

arato@biconsulting.hu

rstats.budapestbi.hu


Master
Code
Data
Code
Data
Code
Data
HDFS Node
HDFS Node
HDFS Node

- SQL on Hadoop – why?
  - Familiar interface for most users
  - BI tools (like Tableau, Power BI etc) also uses SQL to connect
- Many different engines
  - Hive, Impala, Drill, Presto, …
  - Most offers ODBC/JDBC driver, usable from R

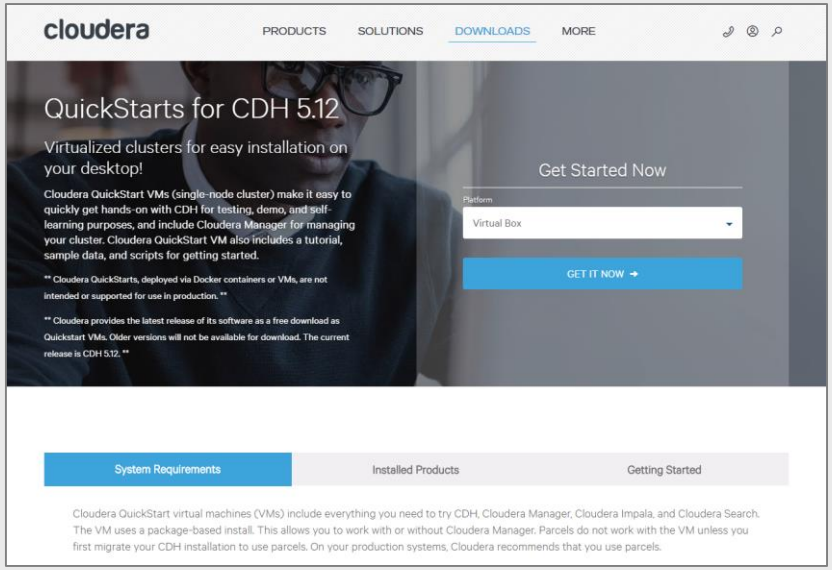
Hadoop Tracker

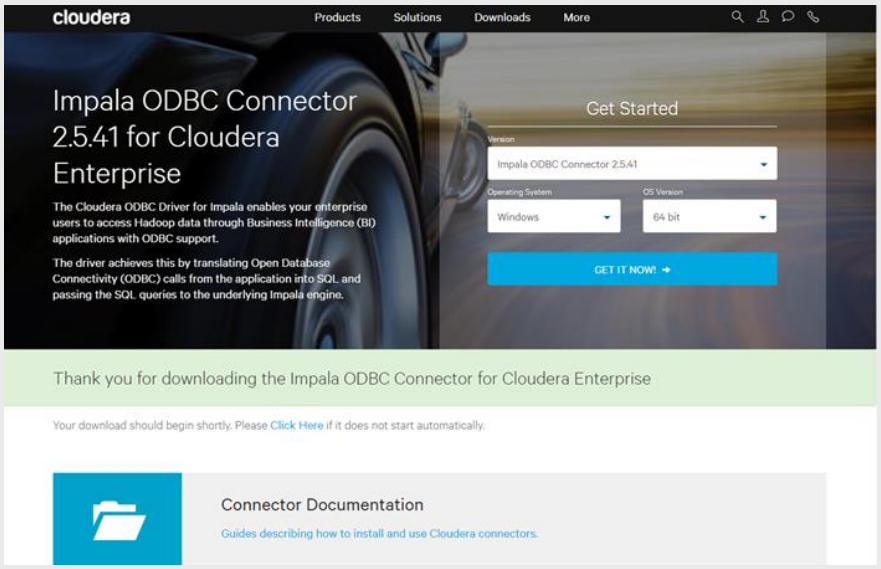
Hadoop Apache Project Commercial Support Tracker December 2017					
	Amazon EMR 5.11	Cloudera CDH 5.13	Google	Hortonworks HDP 2.6.2	MapR MEP 3.0.1
Total supported projects					
Apache HDFS	2.7.3	2.6.0	**2.8.1**	2.7.3	API
Apache Mapreduce	2.7.3	2.6.0	**2.8.1**	2.7.3	2.7.0+
Apache YARN	2.7.3	2.6.0	**2.8.1**	2.7.3	2.7.0+
Apache Hive	**2.3.2**	1.2	2.1.1	2.1.0	2.1.1
Apache Pig	**0.17**	0.12.0	0.16.0	0.16.0	0.16
Apache Spark	**2.2.1**	**2.2**	**2.2.0**	2.1.1	2.1.0
Apache Avro	X	1.7.6	**1.8.2**	1.7.5	1.7.4
Apache Flume	X	1.7.0	1.7.0	1.5.2	1.7
Apache HBase	1.3.1 +S3	1.2	**1.3.1**	1.1.2	1.1.8
Apache Kafka	X	0.11	0.11.0.1	0.10.1.2	0.9 (Streams)
Apache Oozie	4.3.0	4.1.0	4.3.0	4.2.0	4.3.0
Apache Parquet	X	1.51	**1.9.0**	1.8.1	1.8.1
Apache Sqoop	1.4.6	1.4.8	1.99.4	1.4.6	1.4.6
Apache Zookeeper	3.4.10	3.4.5	3.4.6	3.4.6	3.4.5
Hue	4.0.1	4.00	3.11.0	2.6.1	3.12

Hadoop distributions may have a preferred engine (for example Hortonworks -> Hive, Cloudera -> Impala)

blogs.gartner.com/merv-adrian/2017/12/29/december-2017-tracker-wheres-hadoop

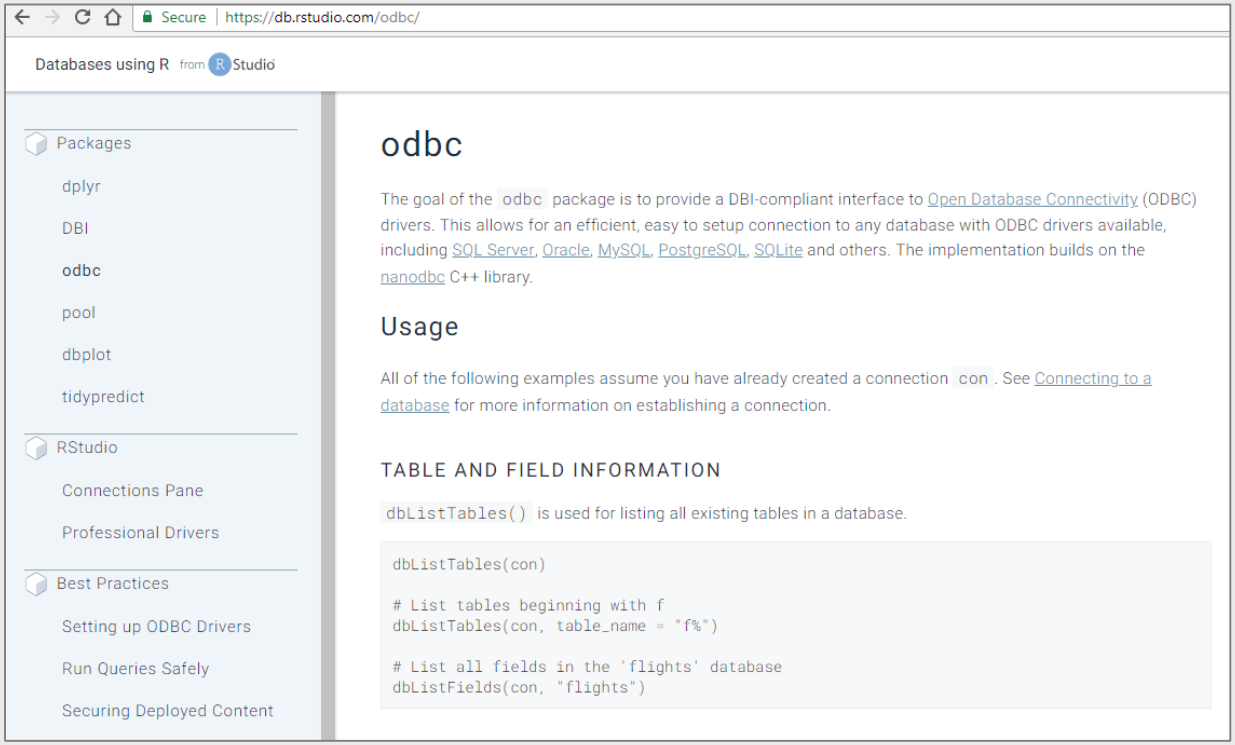
- Steps for trying Impala-based access
  - Download the Cloudera QuickStart VM
  - Download the Impala ODBC driver





ODBC R package

- The ODBC R Package
  - DBI interface for ODBC
  - Maintained by Rstudio



ODBC R package

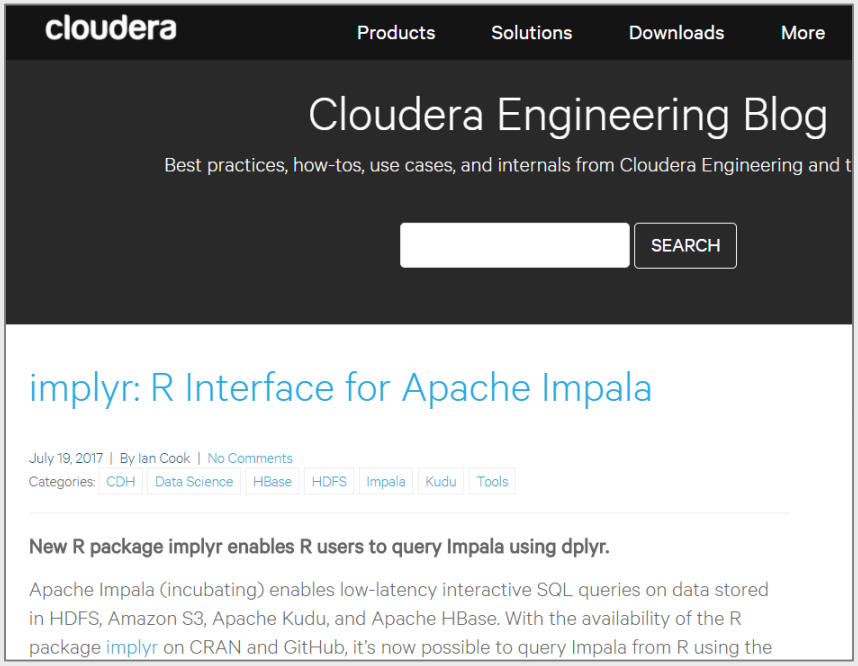
```
library(odbc)
library(DBI)

drv <- odbc::odbc()

con <- dbConnect(drv,
                 driver = "Cloudera ODBC Driver for Impala",
                 host = "localhost",
                 port = 21050,
                 database = "default",
                 uid = "",
                 pwd = ""
)

#list available tables
dbListTables(con)
dbListTables(con, table_name = "%port%")
```

ODBC R package

```
# list fields
dbListFields(con, "airports")

# Load all data from SQL into a local data.frame
df_airports <- dbReadTable(con, "airports")
str(df_airports)

# Read data using an SQL query
query_results= dbGetQuery(con, "select * from airports where faa='SFO'")
head(query_results)

query_results= dbGetQuery(con, "select * from airports where name like '%London%'")
head(query_results)
```

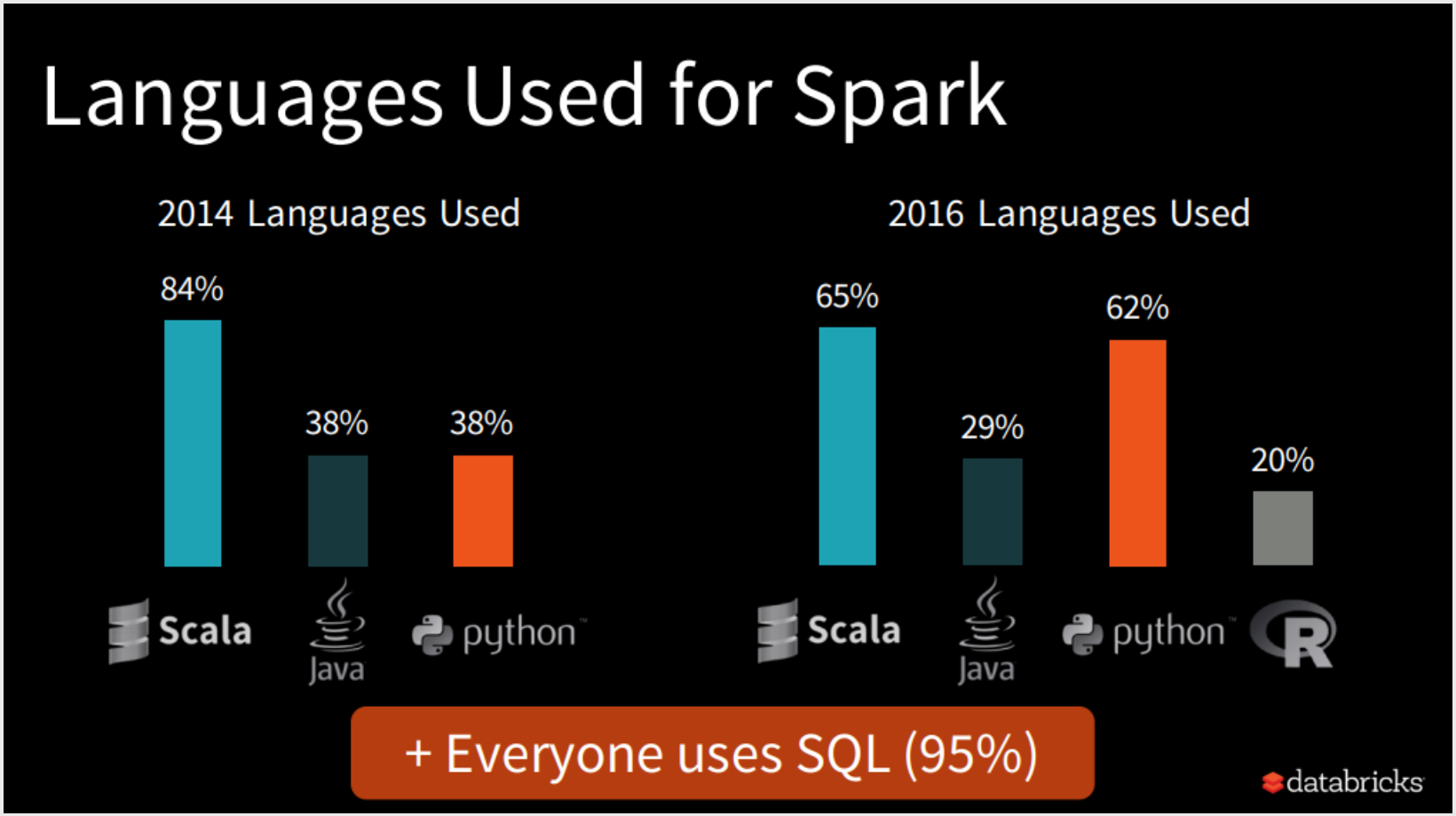
Implyr package

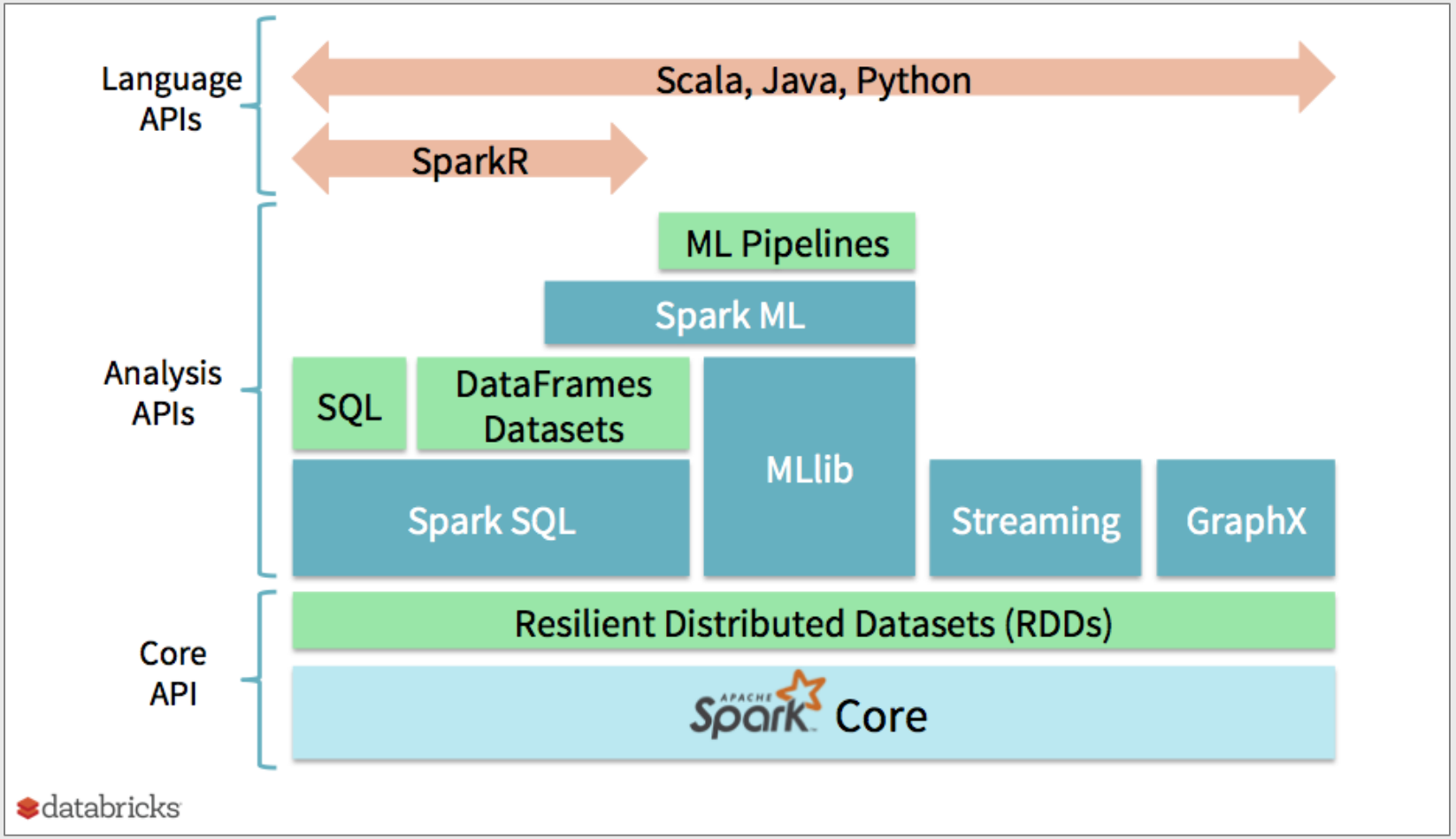
- The implyr package
  - dplyr SQL backend for Impala
  - Developed by Cloudera, Ian Cook
  - Uses the ODBC connector for data access



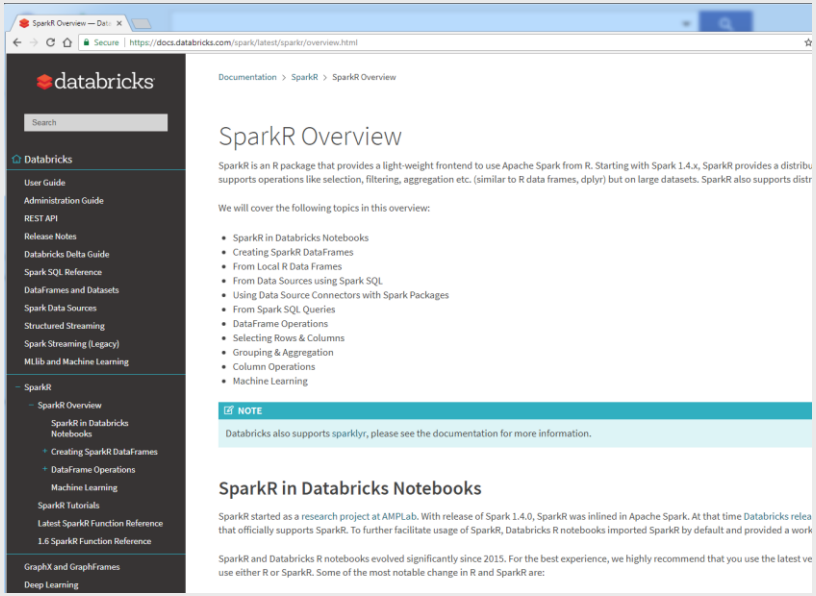
```
# packages
install.packages("implyr")
library(odbc)
library(implyr)
library(dplyr)

drv <- odbc::odbc()

impala <- src_impala(
 drv = drv,
 driver = "Cloudera ODBC Driver for Impala",
 host = "host",
 port = 21050,
 database = "default",
 uid = "username",
 pwd = "password"
)

# list available tables
src_tbls(impala)
```

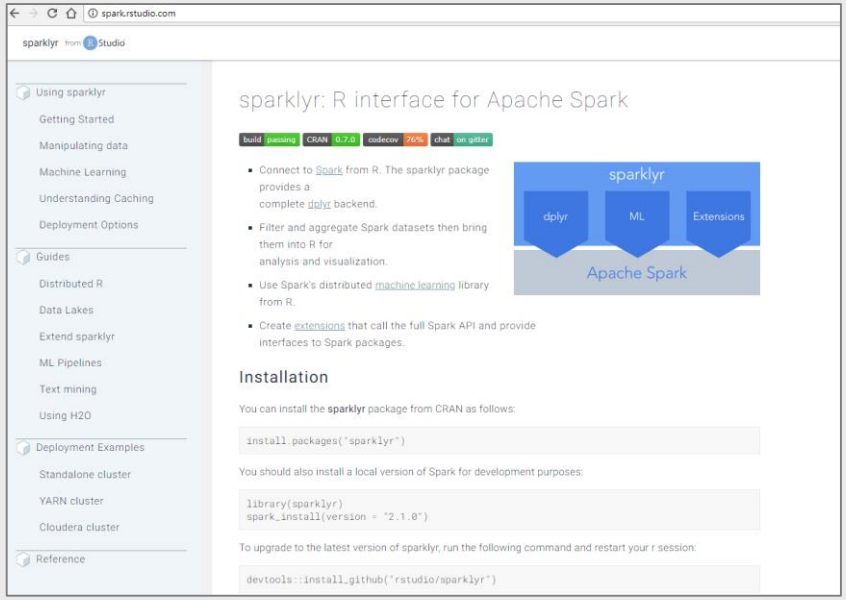
```
# create  airports reference
airports_tbl <- tbl(impala, "airports")
airports_tbl

# Running SQL - refresh Impala metadata
dbExecute(impala, "refresh airports")

# Query data using SQL
airport_cnt <- dbGetQuery( impala, "select count(*) from airports")
airport_cnt

airport_sfo <- dbGetQuery( impala, "select * from airports where faa='SFO'")
head(airport_sfo)

# same using dplyr
airports_tbl %>% filter( FAA =="SFO")
```


Languages Used for Spark
2014 Languages Used
84%
38%
38%
Scala
Java
python
2016 Languages Used
65%
29%
62%
20%
Scala
Java
python
R
+ Everyone uses SQL (95%)
databricks


Language APIs
Scala, Java, Python
SparkR
Analysis APIs
ML Pipelines
Spark ML
SQL
DataFrames Datasets
Spark SQL
MLlib
Streaming
GraphX
Core API
Resilient Distributed Datasets (RDDs)
Spark Core
databricks

- R access for Spark (examples)
  - SparkR (Databricks)
  - RevoScaler (Microsoft)
  - sparklyr (Rstudio)





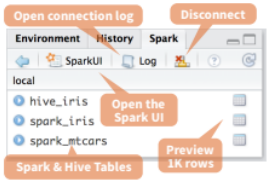
Data Science in Spark with Sparklyr : : CHEAT SHEET

Intro

sparklyr is an R interface for **Apache Spark™**, it provides a complete **dplyr** backend and the option to query directly using **Spark SQL** statement. With sparklyr, you can orchestrate distributed machine learning using either **Spark's MLlib** or **H2O** Sparkling Water.

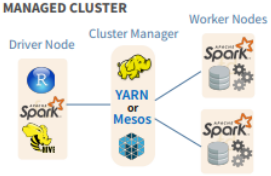
Starting with **version 1.044, RStudio Desktop, Server and Pro include integrated support for the sparklyr package**. You can create and manage connections to Spark clusters and local Spark instances from inside the IDE.

RStudio Integrates with sparklyr



Cluster Deployment

MANAGED CLUSTER



STAND ALONE CLUSTER

RStudio

Data Science Toolchain with Spark + sparklyr

Understand

Import
- Export an R DataFrame
- Read a file
- Read existing Hive table

Tidy
- dplyr verb
- Direct Spark SQL (DBI)
- SDF function (Scala API)

Transform
Transformer function

Visualize
Collect data into R for plotting

Wrangle

Model
- Spark MLlib
- H2O Extension

Communicate
- Collect data into R
- Share plots, documents, and apps

R for Data Science, Grolemund & Wickham

Getting Started

LOCAL MODE (No cluster required)

1. Install a local version of Spark: **spark_install ("2.0.1")**
2. Open a connection **sc <- spark_connect (master = "local")**

ON A MESOS MANAGED CLUSTER

1. Install RStudio Server or Pro on one of the existing nodes
2. Locate path to the cluster's Spark directory
3. Open a connection **spark_connect(master="[mesos URL]", version = "1.6.2", spark_home = [Cluster's Spark path])**

USING LIVY (Experimental)

1. The Livy REST application should be running on the cluster
2. Connect to the cluster **sc <- spark_connect(method = "livy", master = "http://host:port")**

ON A YARN MANAGED CLUSTER

1. Install RStudio Server or RStudio Pro on one of the existing nodes, preferably an edge node
2. Locate path to the cluster's Spark Home Directory, it normally is "/usr/lib/spark"
3. Open a connection **spark_connect(master="yarn-client", version = "1.6.2", spark_home = [Cluster's Spark path])**

ON A SPARK STANDALONE CLUSTER

1. Install RStudio Server or RStudio Pro on one of the existing nodes or a server in the same LAN
2. Install a local version of Spark: **spark_install (version = "2.0.1")**
3. Open a connection **spark_connect(master="spark://host:port", version = "2.0.1", spark_home = spark_home_dir())**

Tuning Spark

EXAMPLE CONFIGURATION

```
config <- spark_config()
config$spark.executor.cores <- 2
config$spark.executor.memory <- "4G"
sc <- spark_connect (master="yarn-client",
config = config, version = "2.0.1")
```

IMPORTANT TUNING PARAMETERS with defaults

- spark.yarn.am.cores
- spark.yarn.am.memory *512m*
- spark.network.timeout *120s*
- spark.executor.memory *1g*
- spark.executor.cores *1*
- spark.executor.instances
- spark.executor.extraJavaOptions
- spark.executor.heartbeatInterval *10s*
- sparklyr.shell.executor-memory
- sparklyr.shell.driver-memory

Using sparklyr

A brief example of a data analysis using Apache Spark, R and sparklyr in local mode

```
library(sparklyr); library(dplyr); library(ggplot2);
library(tidyr);
set.seed(100)
```
Install Spark locally

```
spark_install("2.0.1")
```
Connect to local version

```
sc <- spark_connect(master = "local")

import_iris <- copy_to(sc, iris, "spark_iris",
  overwrite = TRUE)
```
Copy data to Spark memory

```
partition_iris <- sdf_partition(
  import_iris,training=0.5, testing=0.5)
```
Partition data

```
sdf_register(partition_iris,
c("spark_iris_training","spark_iris_test"))
```
Create a hive metadata for each partition

```
tidy_iris <- tbl(sc,"spark_iris_training") %>%
  select(Species, Petal_Length, Petal_Width)

model_iris <- tidy_iris %>%
  ml_decision_tree(response="Species",
  features=c("Petal_Length","Petal_Width"))
```
Spark ML Decision Tree Model

```
test_iris <- tbl(sc,"spark_iris_test")
```
Create reference to Spark table

```
pred_iris <- sdf_predict(
  model_iris, test_iris) %>%
  collect
```
Bring data back into R memory for plotting

```
pred_iris %>%
  inner_join(data.frame(prediction=0:2,
  lab=model_iris$model.parameters$labels)) %>%
  ggplot(aes(Petal_Length, Petal_Width, col=lab)) +
  geom_point()

spark_disconnect(sc)
```
Disconnect

www.rstudio.com/resources/cheatsheets