Beginning Data Exploration and Analysis with Apache Spark

GETTING STARTED WITH SPARK'S RESILIENT DISTRIBUTED DATASETS

Swetha Kolalapudi
CO-FOUNDER, LOONYCORN
www.loonycorn.com

Overview

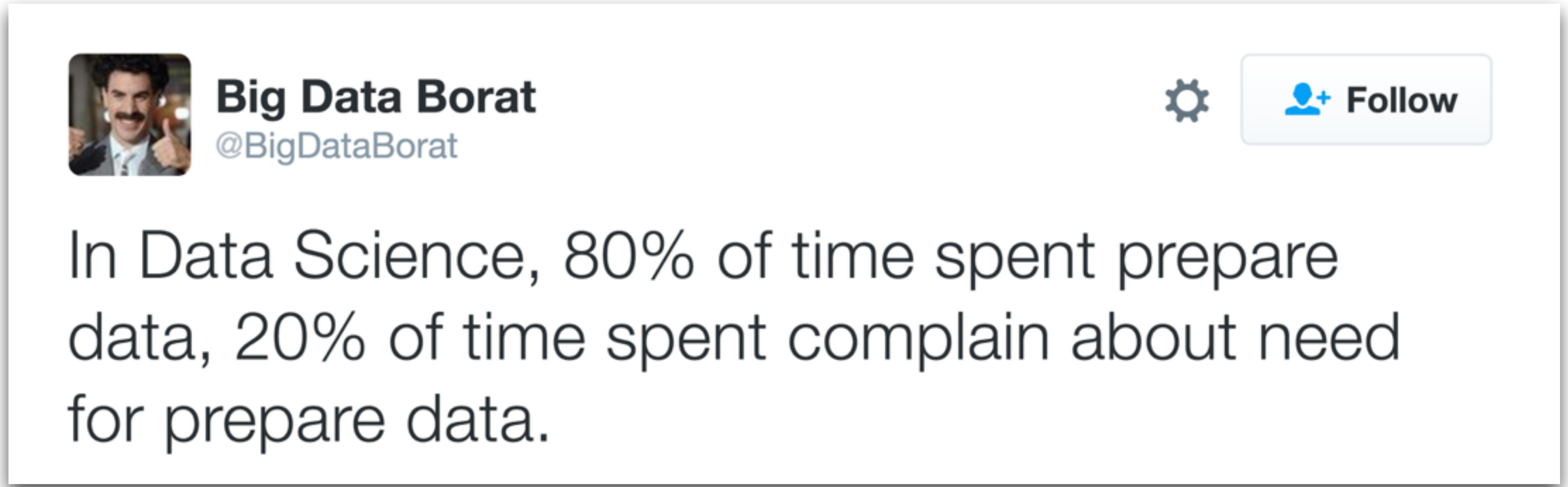
Understanding the role of Spark in data analysis

Understanding RDDs and their characteristics

Installing Spark Standalone in a local environment

Loading data from a file

Reading data from an RDD


Big Data Borat
@BigDataBorat

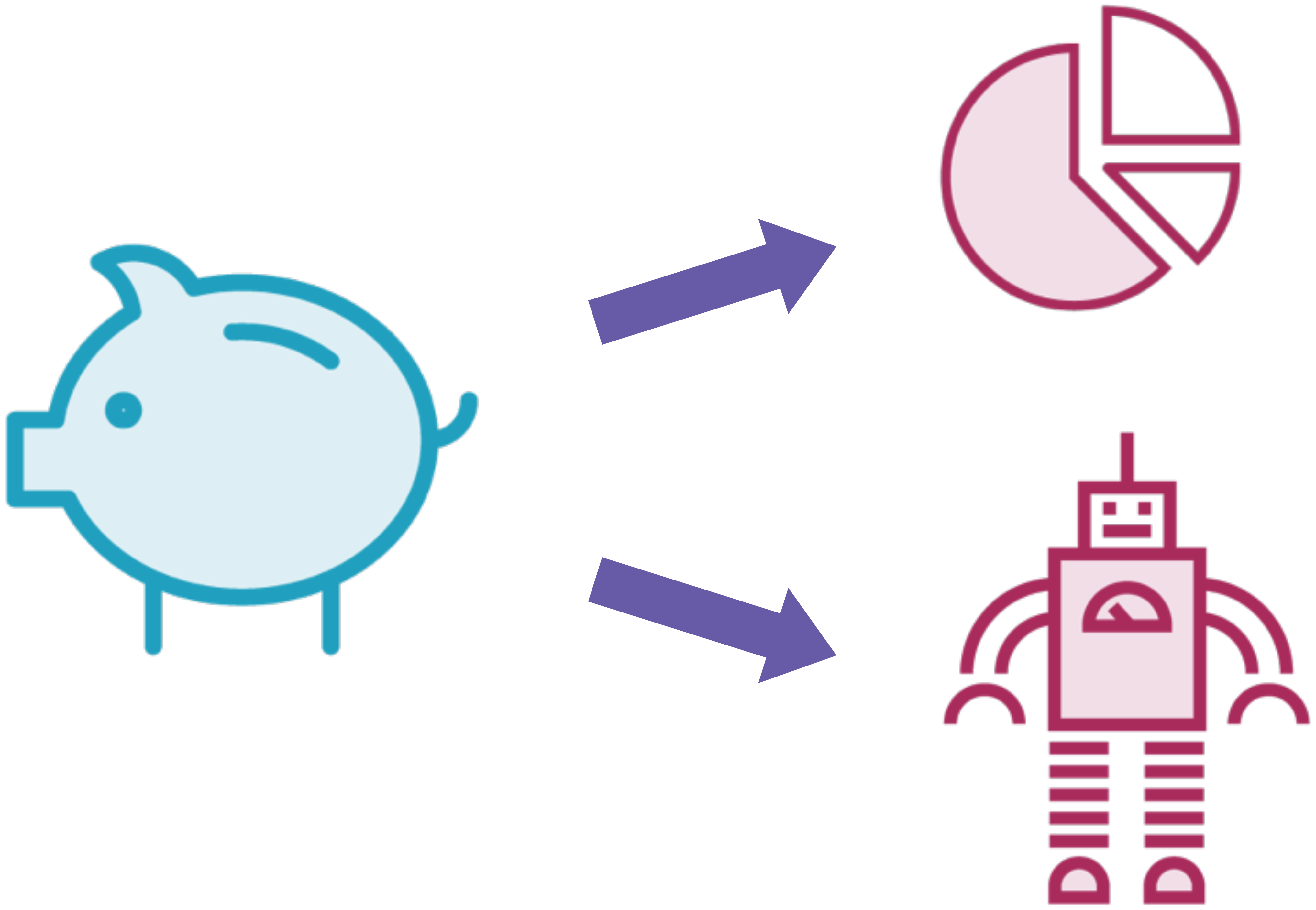
In Data Science, 80% of time spent prepare data, 20% of time spent complain about need for prepare data.

There is no shortage of data today, public or private

Public and Private

Comments	Purchases
Reviews	Pageviews
Tweets	Clicks
Blogs	Messages
Government Reports	Surveys/ Questionnaires

Untold Opportunities



Draw insights

Apply Machine Learning

Doing Something Meaningful with Data

- Unstructured
- Messy
- Semantically complex

- Structured
- Clean
- Easy to consume

Data Processing

Data Processing Tasks

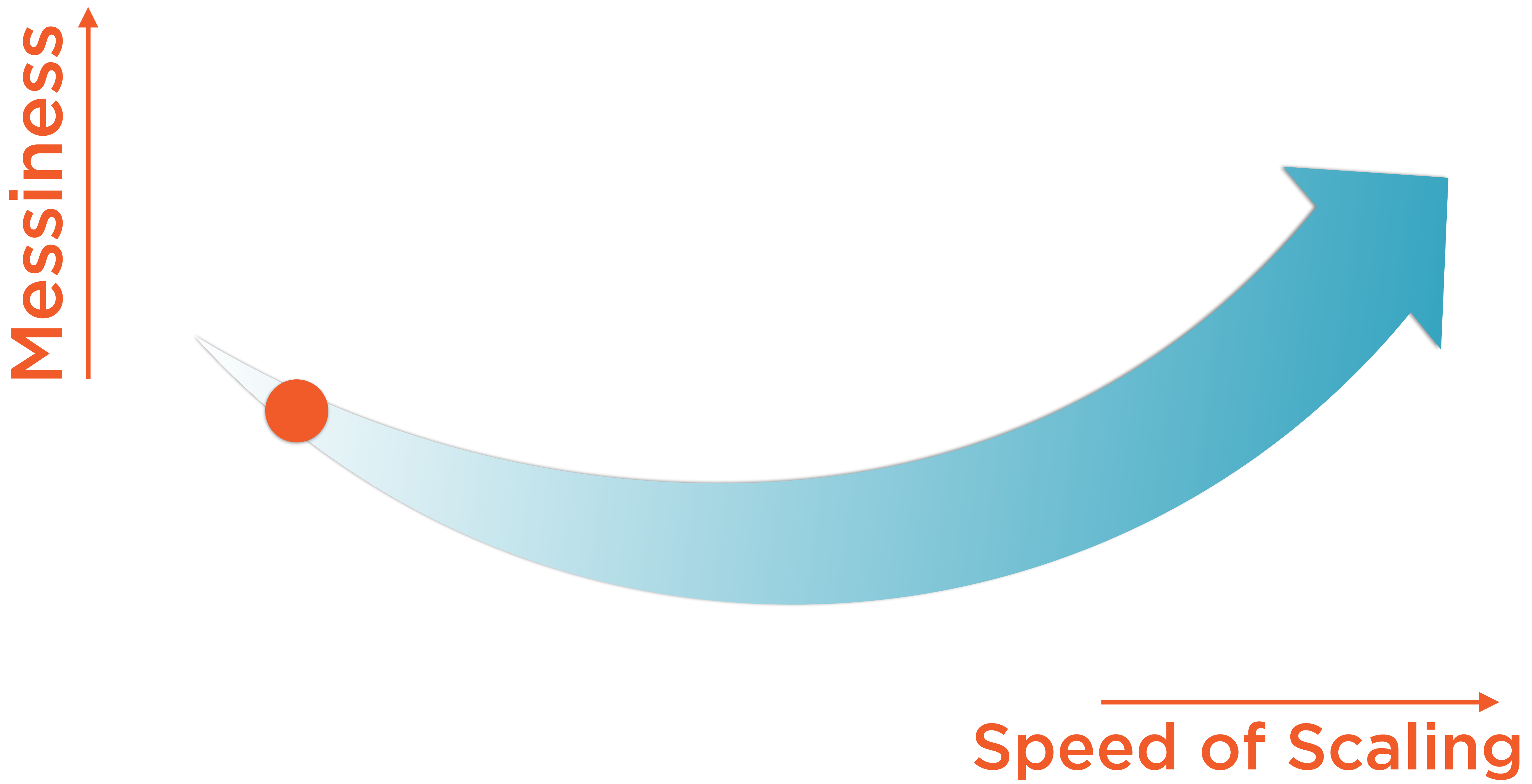
Parsing fields from text

Accounting for missing values

Identifying and investigating anomalies

Summarizing using tables and charts

Data Complexity



Small Data

Somewhat Messy

Spreadsheets

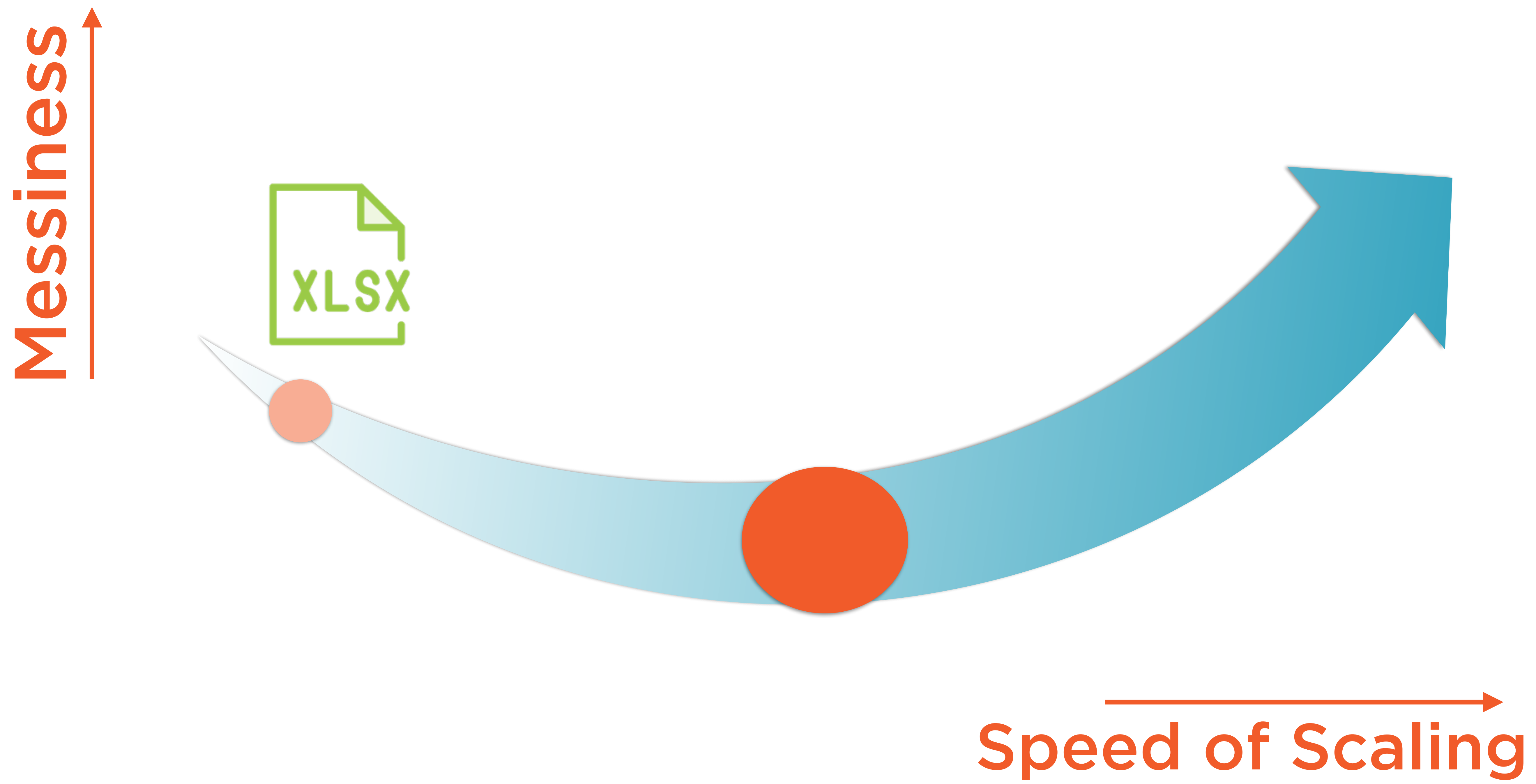
Low data collection frequency

10s - 1000s of rows per day

Sometimes involves manual data collection

Many many files

Data Complexity



Medium Data

High Data Integrity

Databases

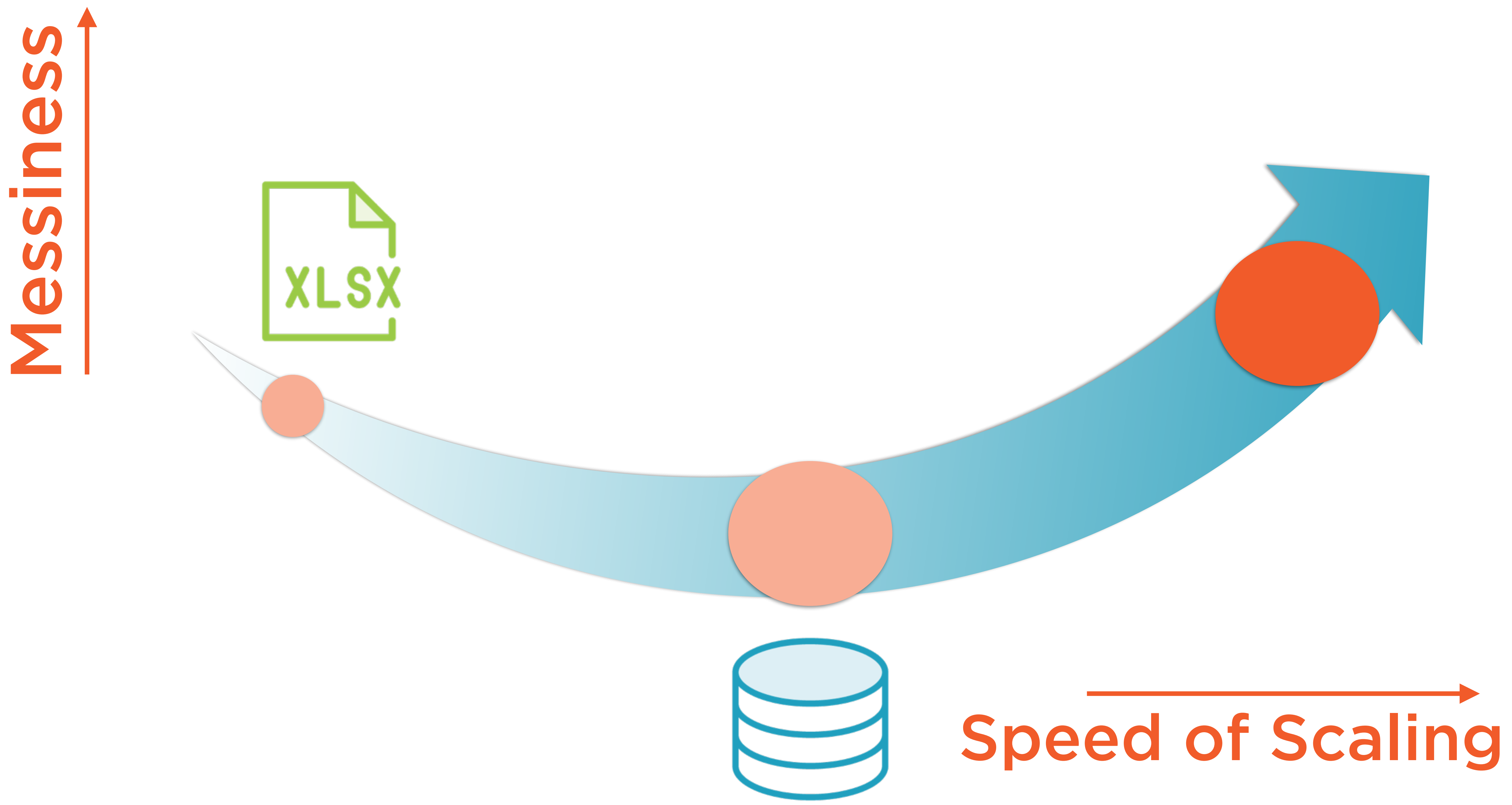
High frequency of collection

~100K rows per day

Programmatically collected

ACID properties

Data Complexity



Big Data

Very messy

Distributed Computing

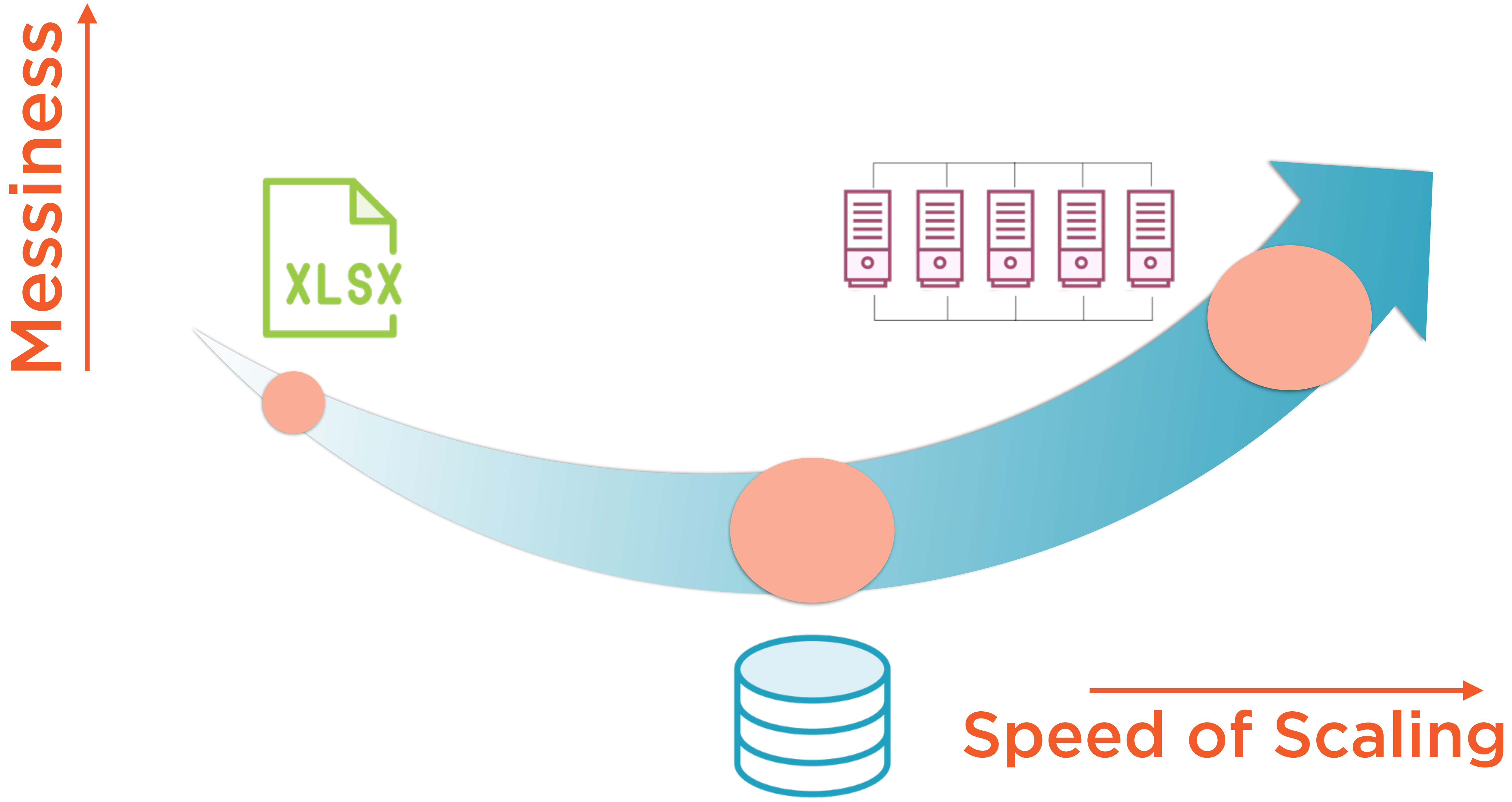
Very high frequency of data collection

Millions/Billions of rows per day

Files stored across a cluster of machines

Many many files (webpages, log files)

Data Complexity



Tools for Data Munging

Excel

SQL
Python
Java

Hadoop MapReduce
Spark

Spark

An engine for data processing and analysis

General Purpose

Interactive

Distributed Computing

General Purpose

Exploring

Cleaning and Preparing

Applying Machine Learning

Building Data Applications

Spark

An engine for data processing and analysis

General Purpose

Interactive

Distributed Computing

Interactive

```
    len([1,2,5])
>>> 3
```

REPL

Interactive

REPL

Read-Evaluate-Print-Loop

Interactive environments

Fast feedback

Spark

An engine for data processing and analysis

General Purpose

Interactive

Distributed Computing

Distributed Computing

Process data across a cluster of machines

Integrate with Hadoop

Read data from HDFS

Spark APIs

Scala

Python

Java

Almost all data is processed using

Resilient Distributed Datasets

Resilient Distributed Datasets

RDDs are the main programming abstraction in Spark

Resilient Distributed Datasets

RDDs are in-memory collections of objects

In-memory,
yet resilient!

Resilient Distributed Datasets

With RDDs, you can interact and play with billions of rows of data

...without caring about any of the complexities

Spark is made up of a few different components

Spark Core

The basic functionality of Spark

RDDs

Spark Core

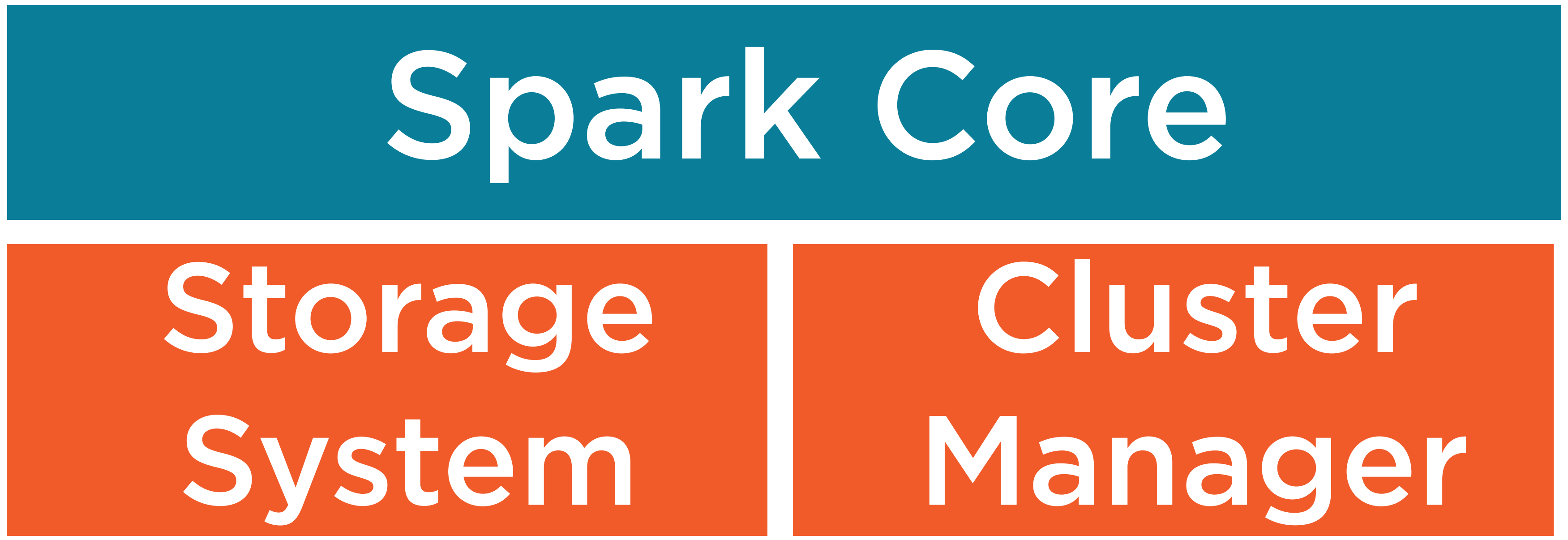
Spark Core is just a computing engine

It needs two additional components

Spark Core

A **Storage System** that stores the data to be processed

A **Cluster Manager** to help Spark run tasks across a cluster of machines

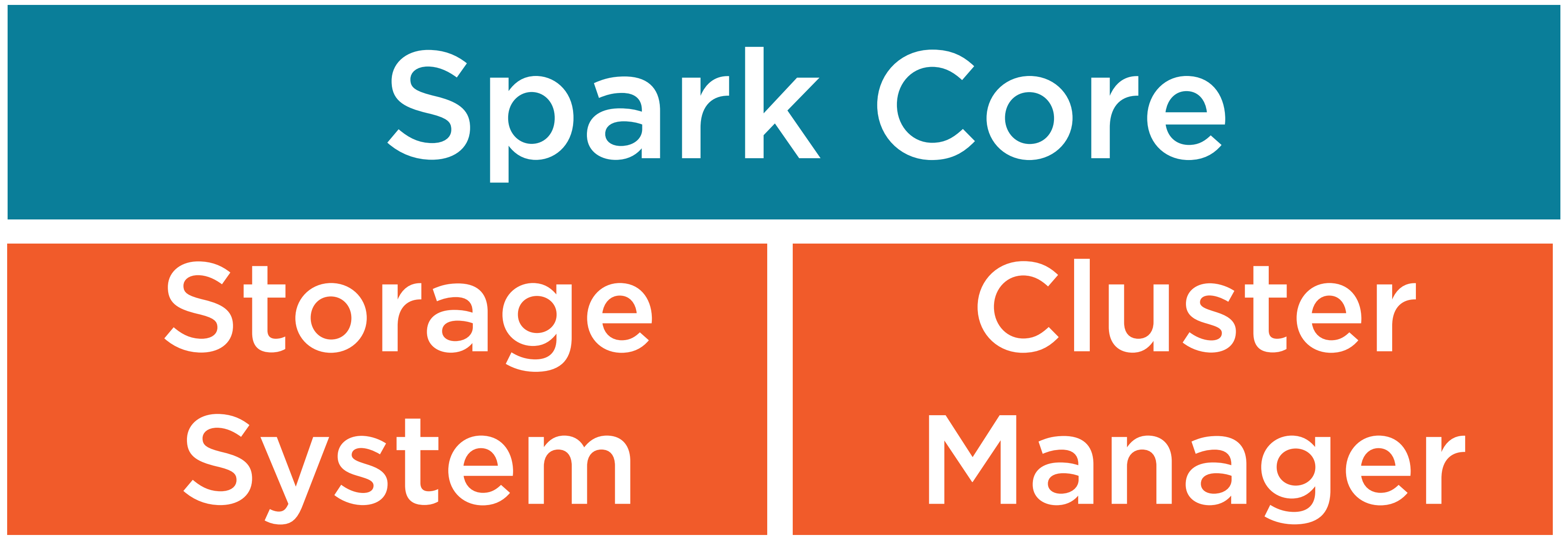


Both of these are plug and play components

Local file system

HDFS

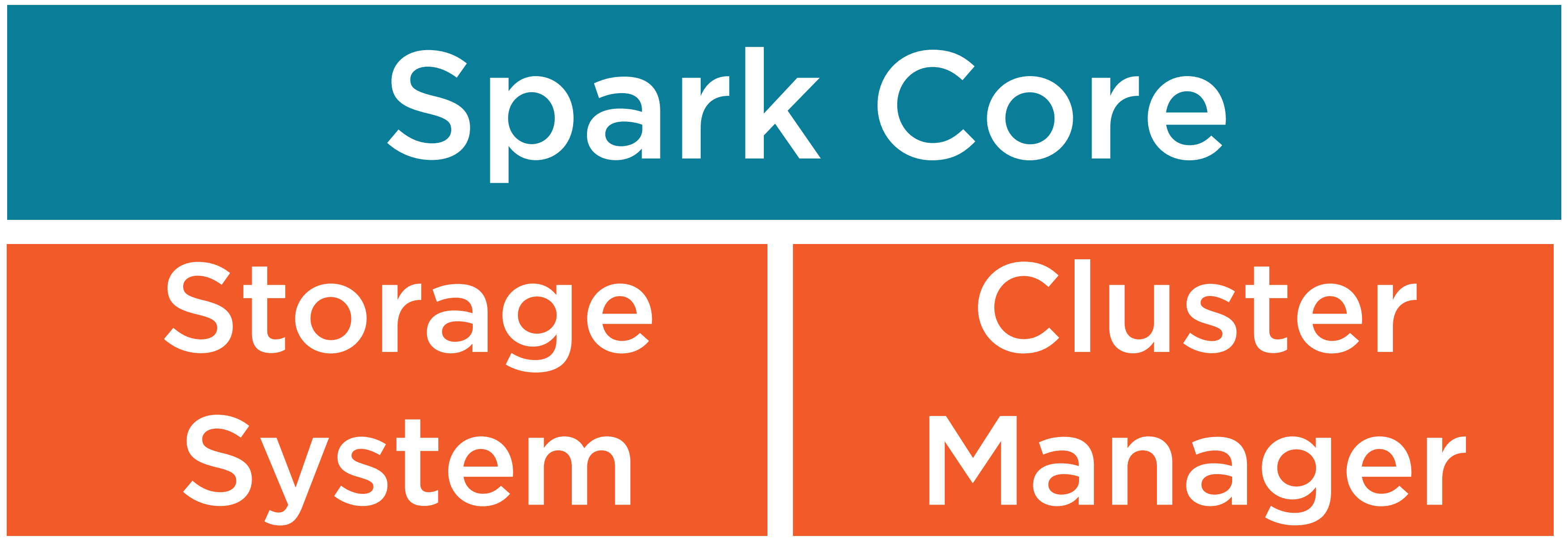
Storage System


Spark Core
Storage System
Cluster Manager

Built-in Cluster Manager

YARN

Cluster Manager

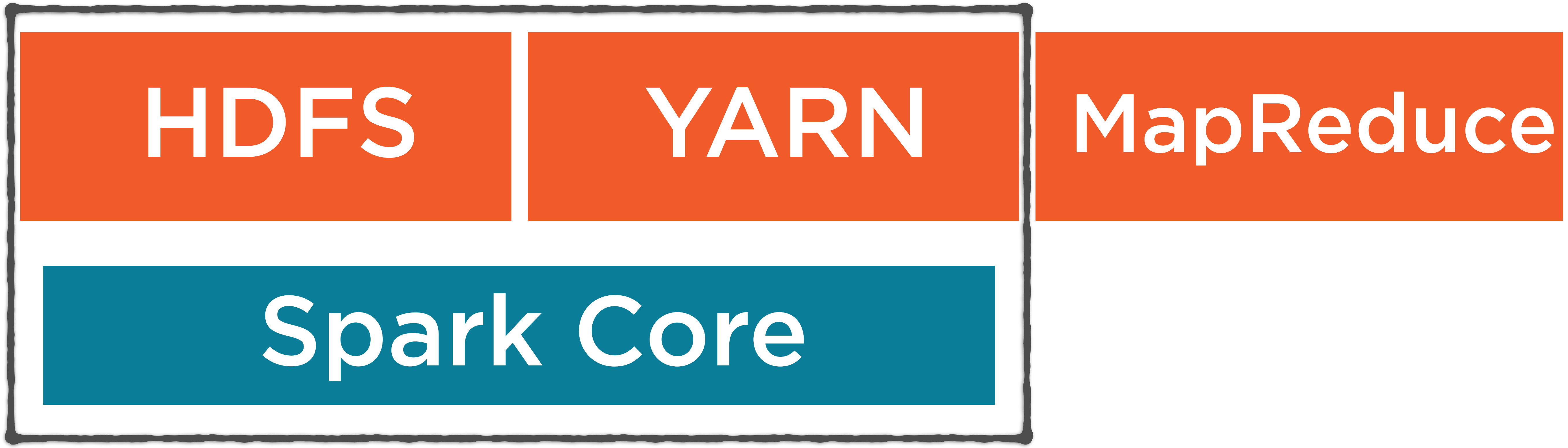


Plug and Play makes it easy to integrate with Hadoop

A Hadoop Cluster

HDFS	YARN	MapReduce
For storage	For managing the cluster	For computing

A Hadoop Cluster



Use Spark as an alternate/additional compute engine

Installing Spark

Prerequisites

Java 7 or above

Scala

IPython (Anaconda)

Installing Spark

Download Spark binaries

Update environment variables

Configure iPython Notebook for Spark

Spark Environment Variables

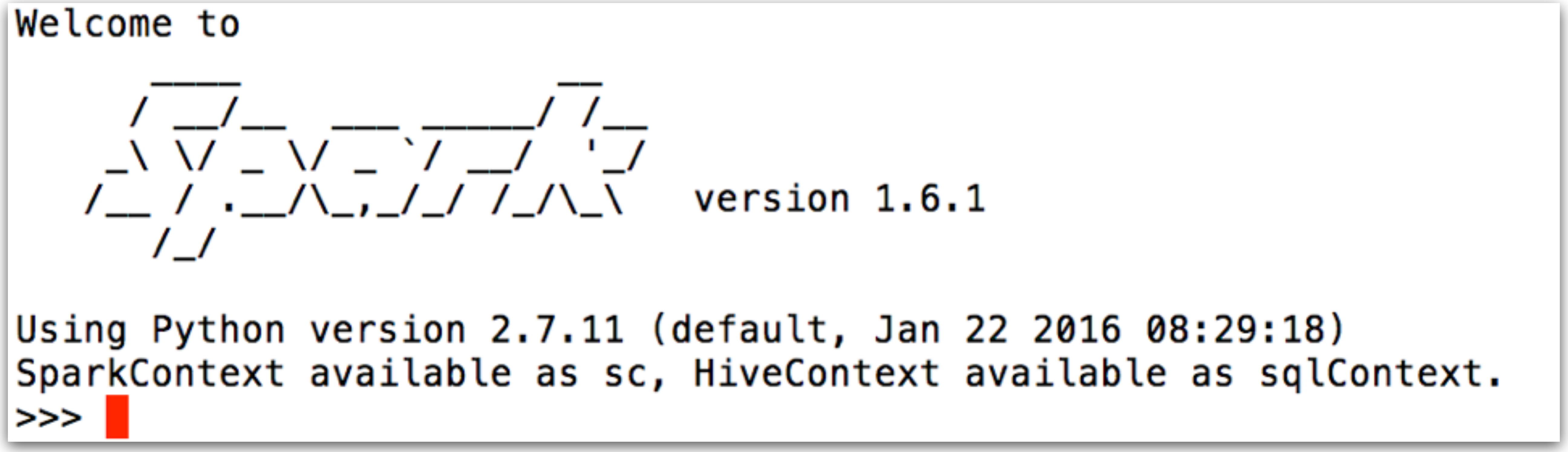
SPARK_HOME	Point to the folder where Spark has been extracted	
PATH	`$SPARK_HOME/bin`	Linux/Mac OS X
	`%SPARK_HOME%/bin`	Windows

Configuring IPython

`PYSPARK_DRIVER_PYTHON`	`ipython`
`PYSPARK_DRIVER_PYTHON_OPTS`	`“notebook”`

Launch PySpark

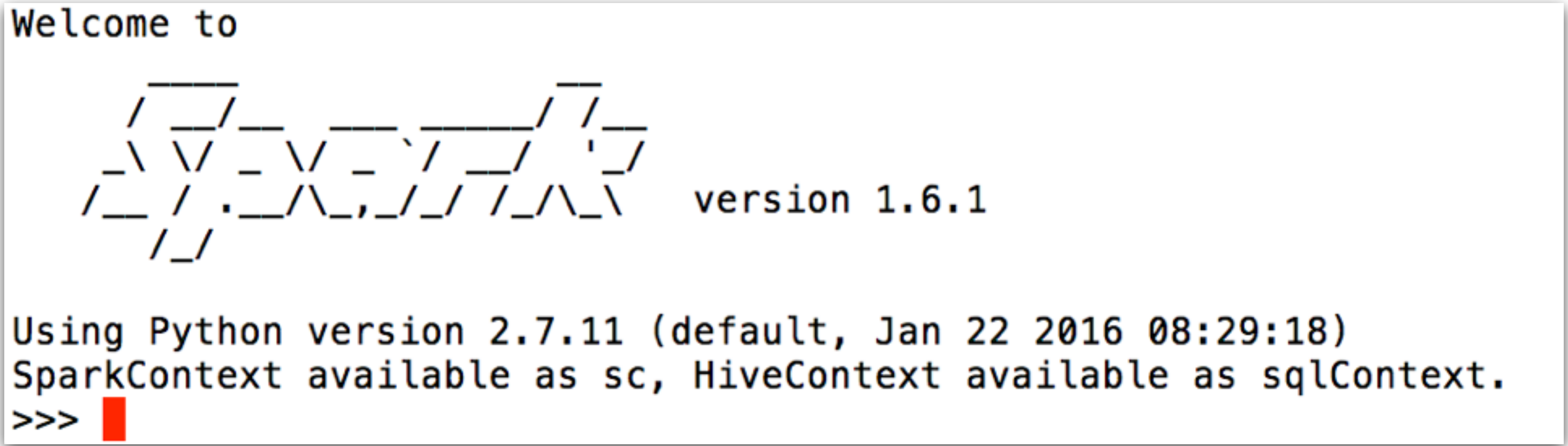
```
> pyspark
```



```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.6.1
      /_/

Using Python version 2.7.11 (default, Jan 22 2016 08:29:18)
SparkContext available as sc, HiveContext available as sqlContext.
>>>
```

Launch PySpark

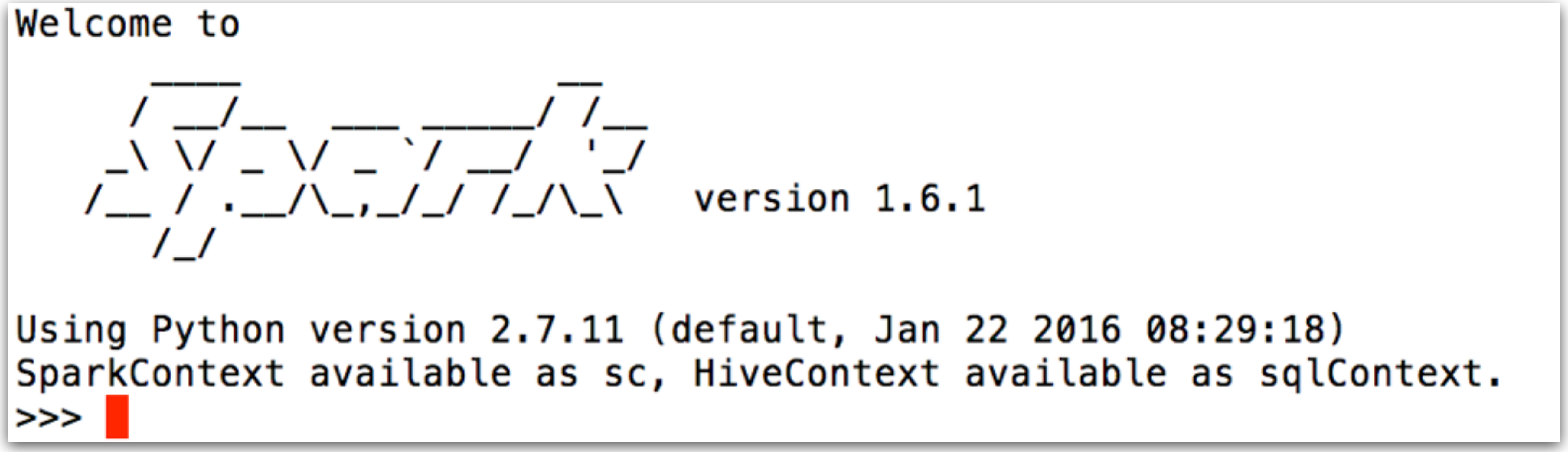


```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.6.1
      /_/

Using Python version 2.7.11 (default, Jan 22 2016 08:29:18)
SparkContext available as sc, HiveContext available as sqlContext.
>>> 
```

This is just like a Python shell

Launch PySpark

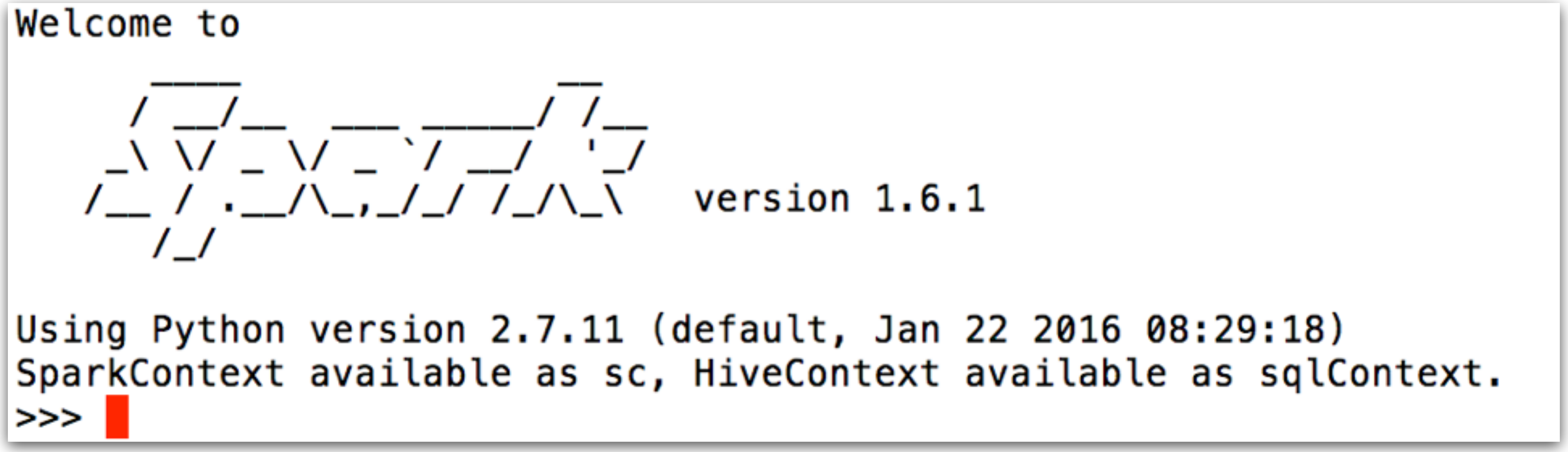


```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.6.1
      /_/

Using Python version 2.7.11 (default, Jan 22 2016 08:29:18)
SparkContext available as sc, HiveContext available as sqlContext.
>>> 
```

Use Python functions, dicts, lists etc

Launch PySpark

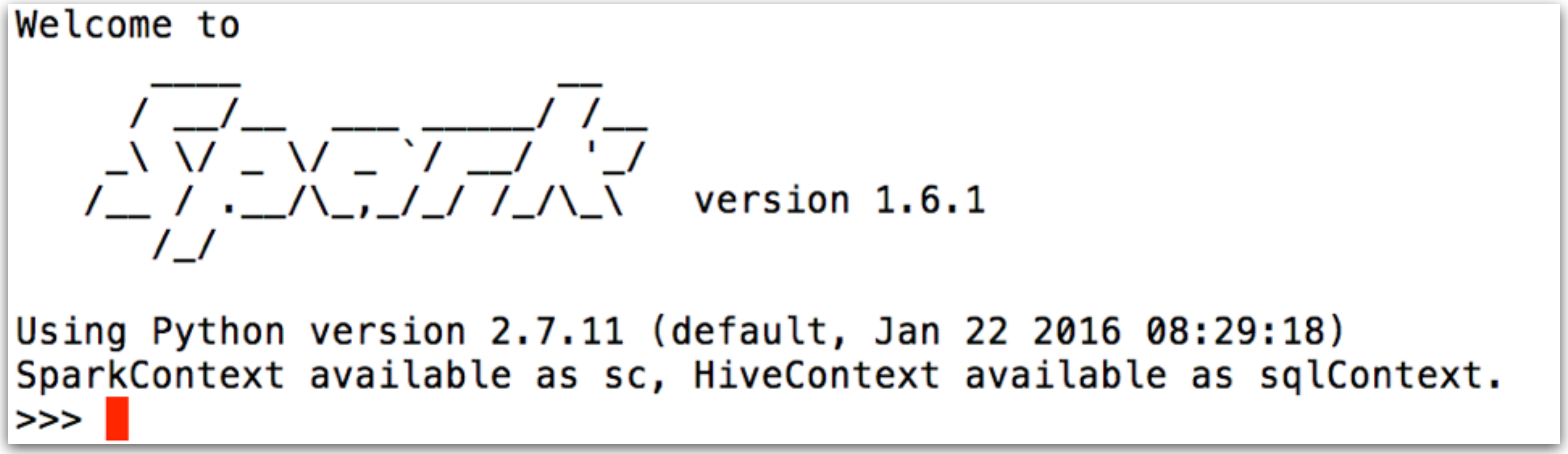


```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.6.1
      /_/

Using Python version 2.7.11 (default, Jan 22 2016 08:29:18)
SparkContext available as sc, HiveContext available as sqlContext.
>>>
```

You can import and use any installed Python modules

Launch PySpark

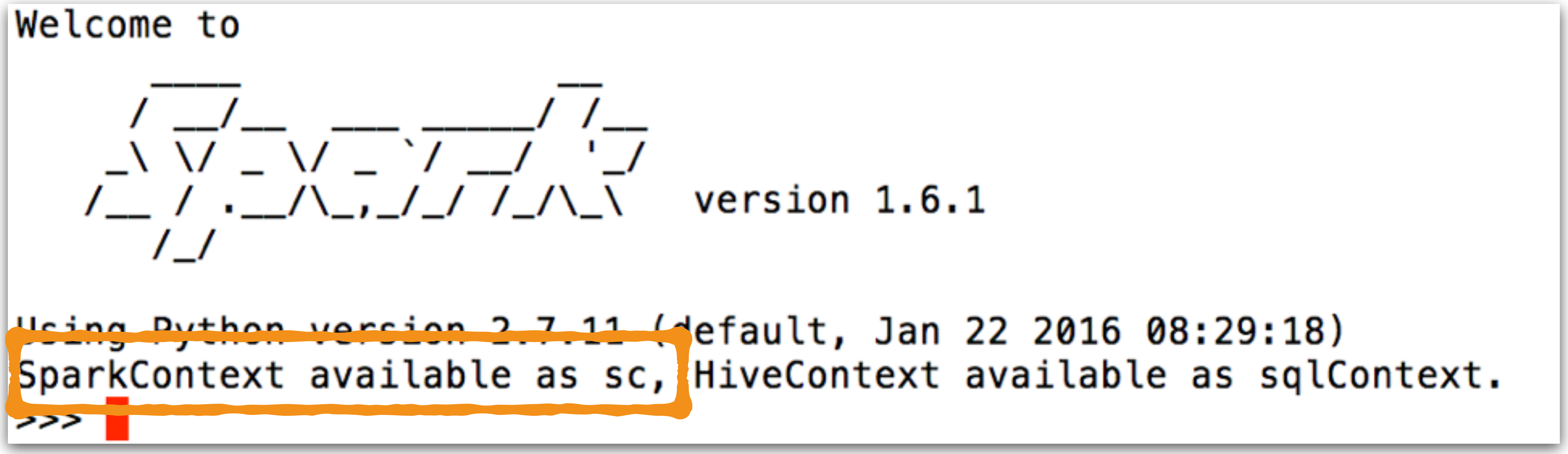


```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.6.1
      /_/

Using Python version 2.7.11 (default, Jan 22 2016 08:29:18)
SparkContext available as sc, HiveContext available as sqlContext.
>>> 
```

Launches by default in a local non-distributed mode

SparkContext

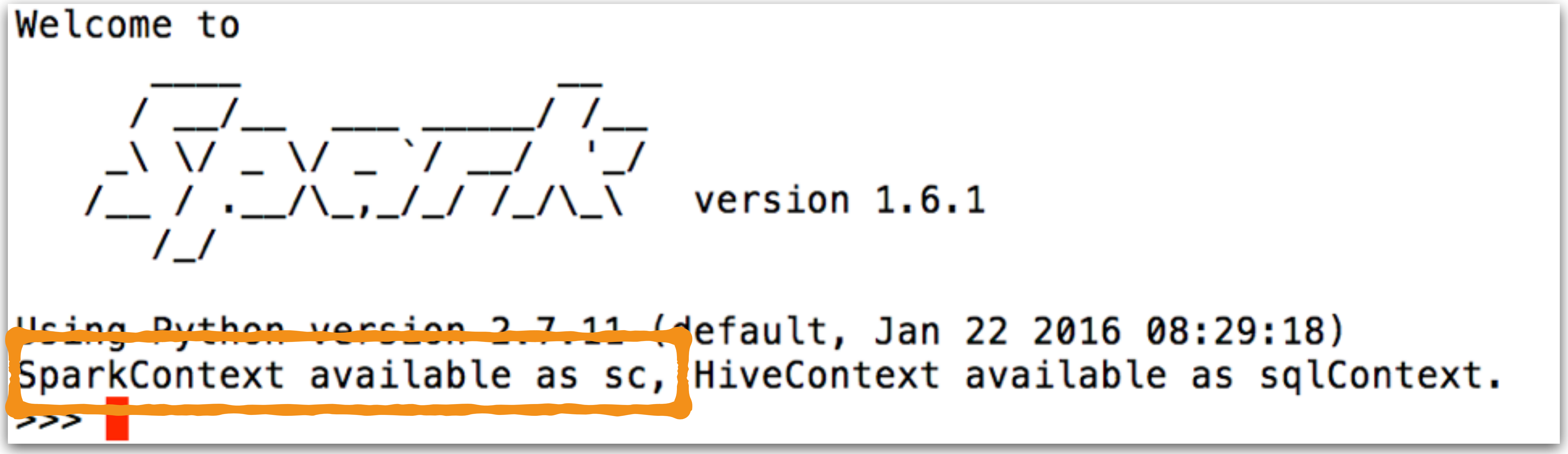


```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.6.1
      /_/

Using Python version 2.7.11 (default, Jan 22 2016 08:29:18)
SparkContext available as sc, HiveContext available as sqlContext.
>>>
```

When the shell is launched it initializes a SparkContext

SparkContext

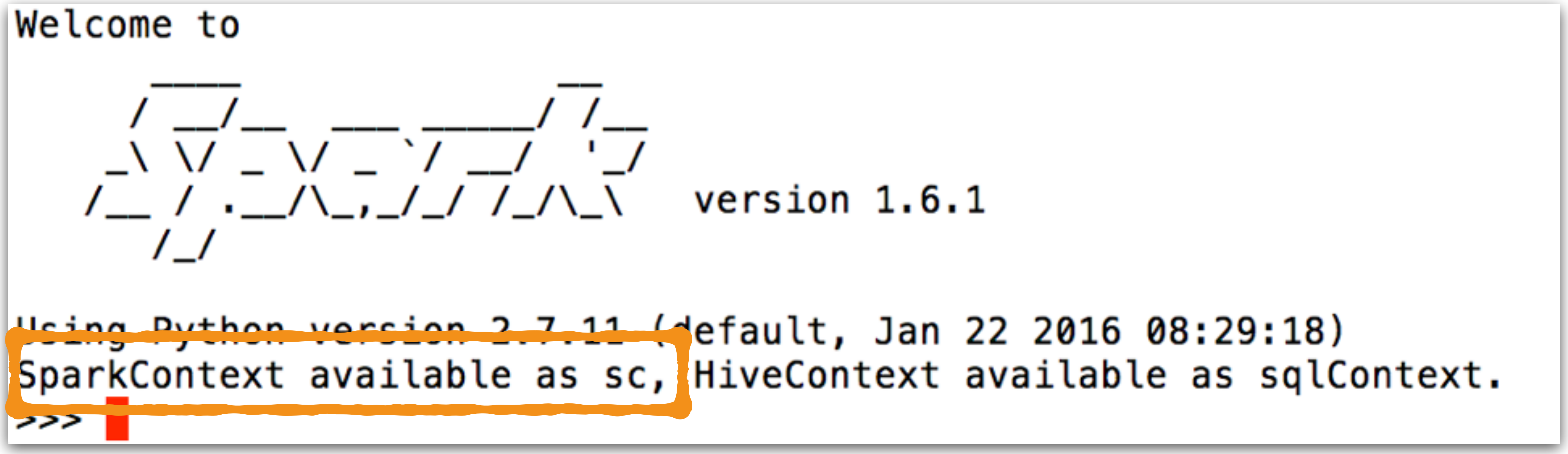


```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.6.1
      /_/

Using Python version 2.7.11 (default, Jan 22 2016 08:29:18)
SparkContext available as sc, HiveContext available as sqlContext.
>>>
```

The SparkContext represents a connection to the Spark Cluster

SparkContext

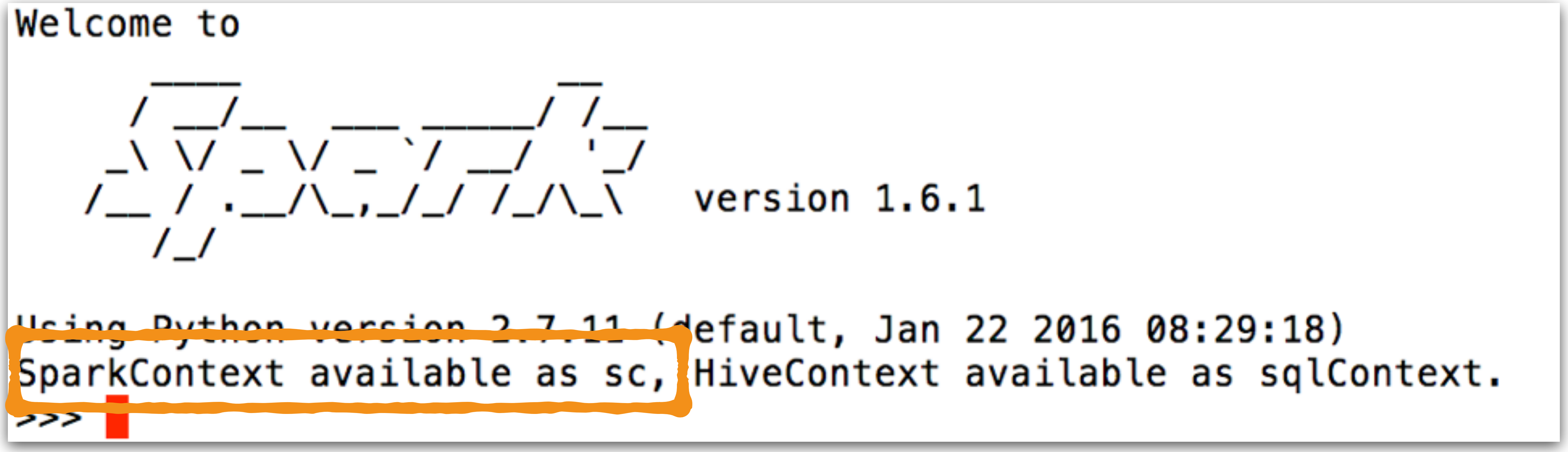


```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.6.1
      /_/

Using Python version 2.7.11 (default, Jan 22 2016 08:29:18)
SparkContext available as sc, HiveContext available as sqlContext.
>>> 
```

Used to load data into memory from a specified source

SparkContext



```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.6.1
      /_/

Using Python version 2.7.11 (default, Jan 22 2016 08:29:18)
SparkContext available as sc, HiveContext available as sqlContext.
>>>
```

The data gets loaded into an RDD

Resilient
Distributed
Datasets

Partitions
Read-only
Lineage

RDDs represent data in-memory

Partitions

1	Swetha	30	Bangalore
2	Vitthal	35	New Delhi
3	Navdeep	25	Mumbai
4	Janani	35	New Delhi
5	Navdeep	25	Mumbai
6	Janani	35	New Delhi

Partitions

Data is divided into partitions

Distributed to multiple machines

1	Swetha	30	Bangalore
2	Vitthal	35	New Delhi
3	Navdeep	25	Mumbai
4	Janani	35	New Delhi
5	Navdeep	25	Mumbai
6	Janani	35	New Delhi

Partitions

Data is divided into partitions

1	Swetha	30	Bangalore
2	Vitthal	35	New Delhi
3	Navdeep	25	Mumbai
4	Janani	35	New Delhi
5	Navdeep	25	Mumbai
6	Janani	35	New Delhi

Partitions

Data is divided into partitions

1	Swetha	30	Bangalore
2	Vitthal	35	New Delhi

3	Navdeep	25	Mumbai
4	Janani	35	New Delhi

5	Navdeep	25	Mumbai
6	Janani	35	New Delhi

Distributed to multiple machines, called nodes

Partitions

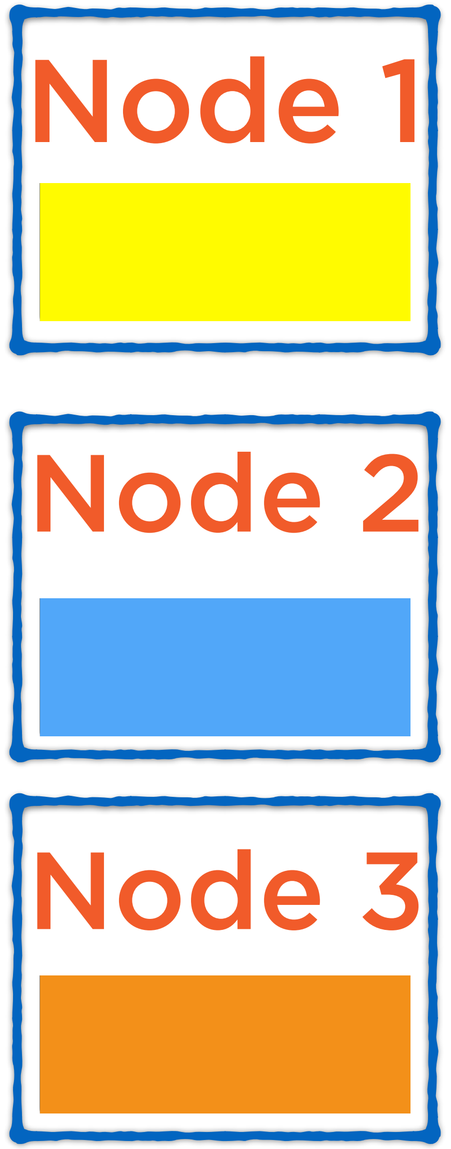
1	Swetha	30	Bangalore
2	Vitthal	35	New Delhi

3	Navdeep	25	Mumbai
4	Janani	35	New Delhi

5	Navdeep	25	Mumbai
6	Janani	35	New Delhi

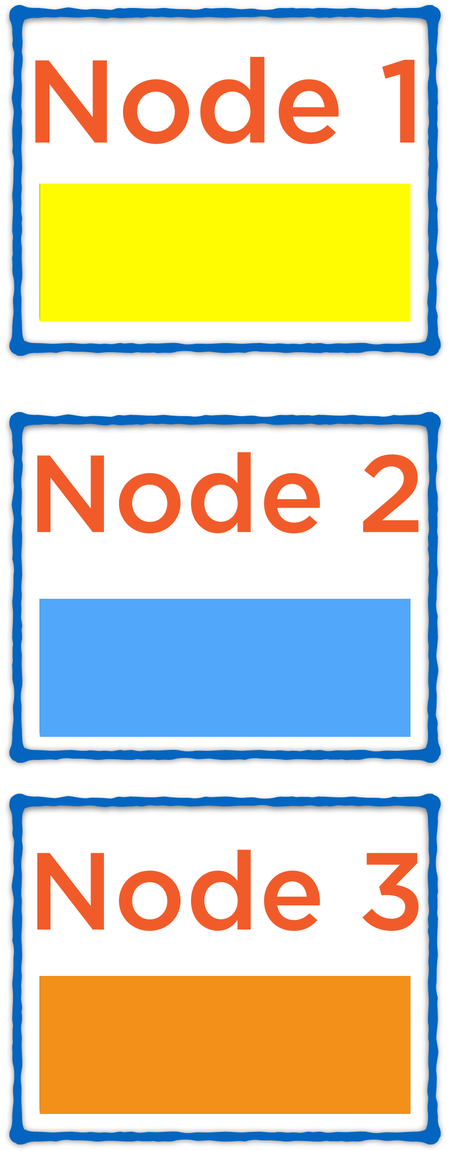
Partitions

Distributed to multiple machines, called nodes



Nodes process data in parallel

Partitions



Resilient
Distributed
Datasets

Partitions
Read-only
Lineage

Read-only

RDDs are immutable

Only Two Types of Operations

Transformation

Transform into another RDD

Action

Request a result

Transformation

1	Swetha	30	Bangalore
2	Vitthal	35	New Delhi
3	Navdeep	25	Mumbai
4	Janani	35	New Delhi
5	Navdeep	25	Mumbai
6	Janani	35	New Delhi

A data set loaded into an RDD

Transformation

1	Swetha	30	Bangalore
2	Vitthal	35	New Delhi
3	Navdeep	25	Mumbai
4	Janani	35	New Delhi
5	Navdeep	25	Mumbai
6	Janani	35	New Delhi

The user may define a chain of transformations on the dataset

Transformation

1	Swetha	30	Bangalore
2	Vitthal	35	New Delhi
3	Navdeep	25	Mumbai
4	Janani	35	New Delhi
5	Navdeep	25	Mumbai
6	Janani	35	New Delhi

1. **Load data**
2. **Pick only the 3rd column**
3. **Sort the values**

Transformation

1. Load data
2. Pick only the 3rd column
3. Sort the values

Transformation

Wait until a result is requested before executing any of these transformations

Only Two Types of Operations

Transformation

Transform into another RDD

Action

Request a result

Action

Request a result using an action

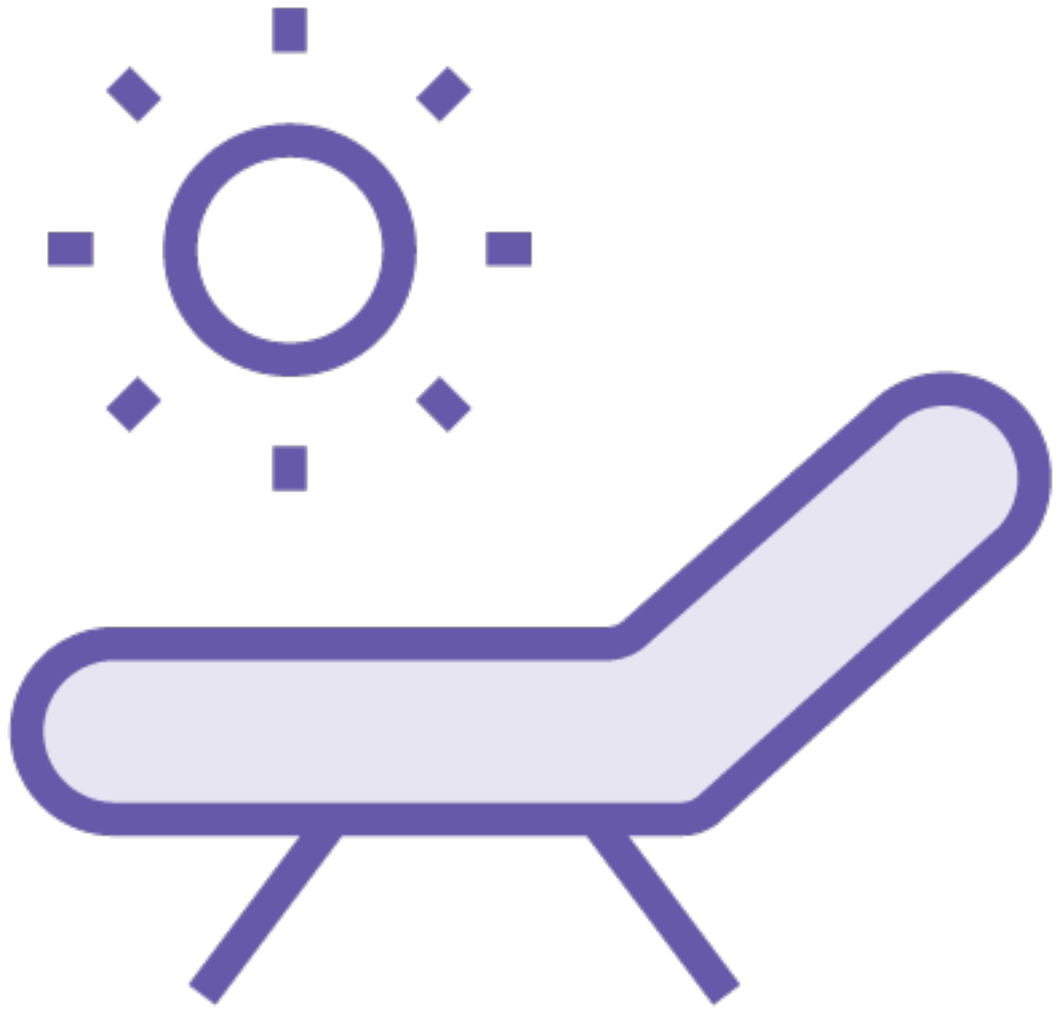
1. The first 10 rows
2. A count
3. A sum

Action

Data is processed only when the user requests a result

The chain of transformations defined earlier is executed

Lazy Evaluation



Spark keeps a record of the series of transformations requested by the user

Lazy Evaluation

It groups the transformations in an efficient way when an Action is requested

Resilient
Distributed
Datasets

Partitions
Read-only
Lineage

Lineage

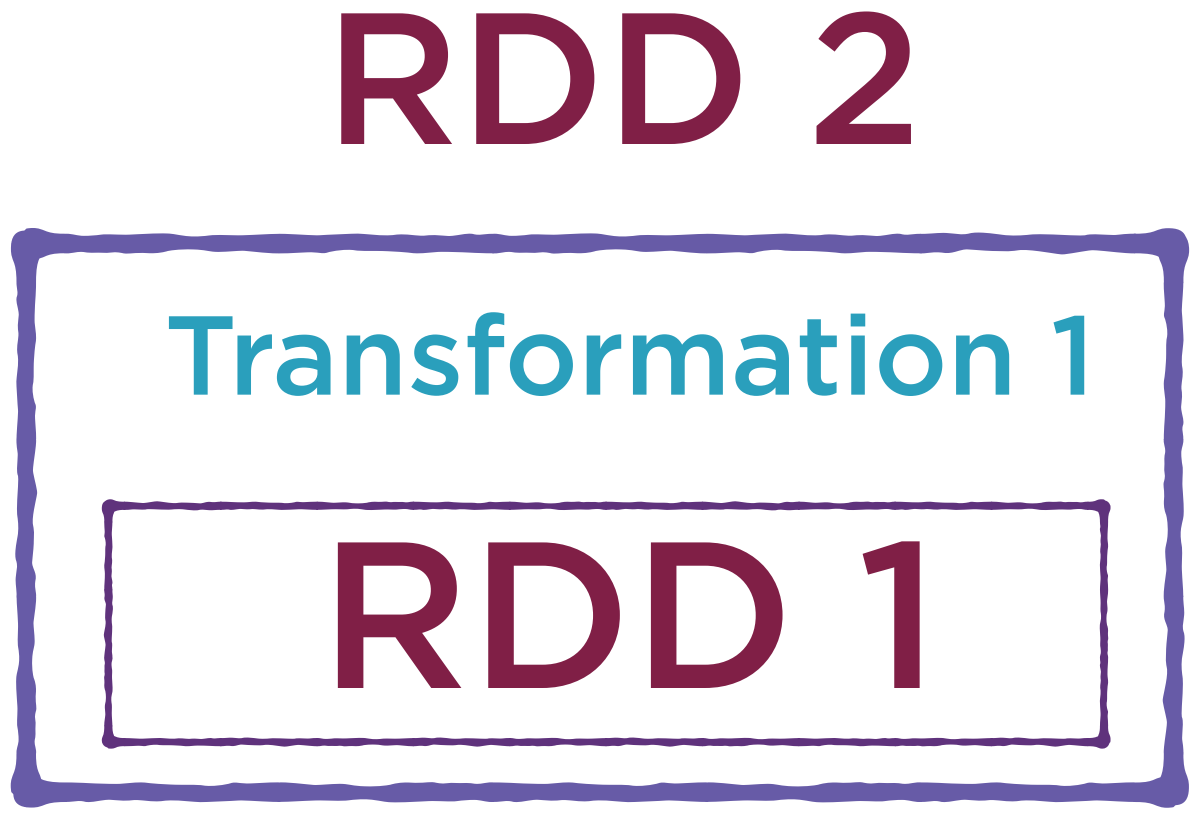
When created, an RDD just holds metadata

1. A transformation

2. It’s parent RDD

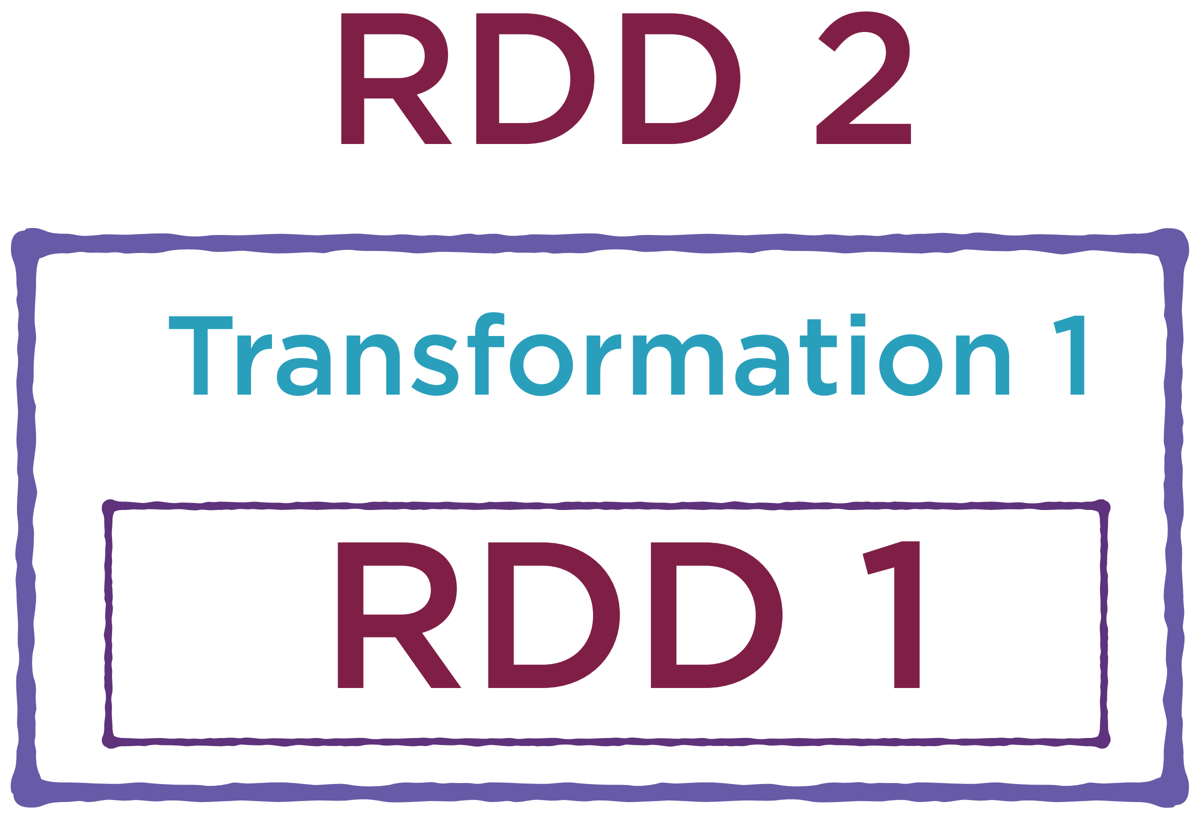
Lineage

Every RDD knows where it came from



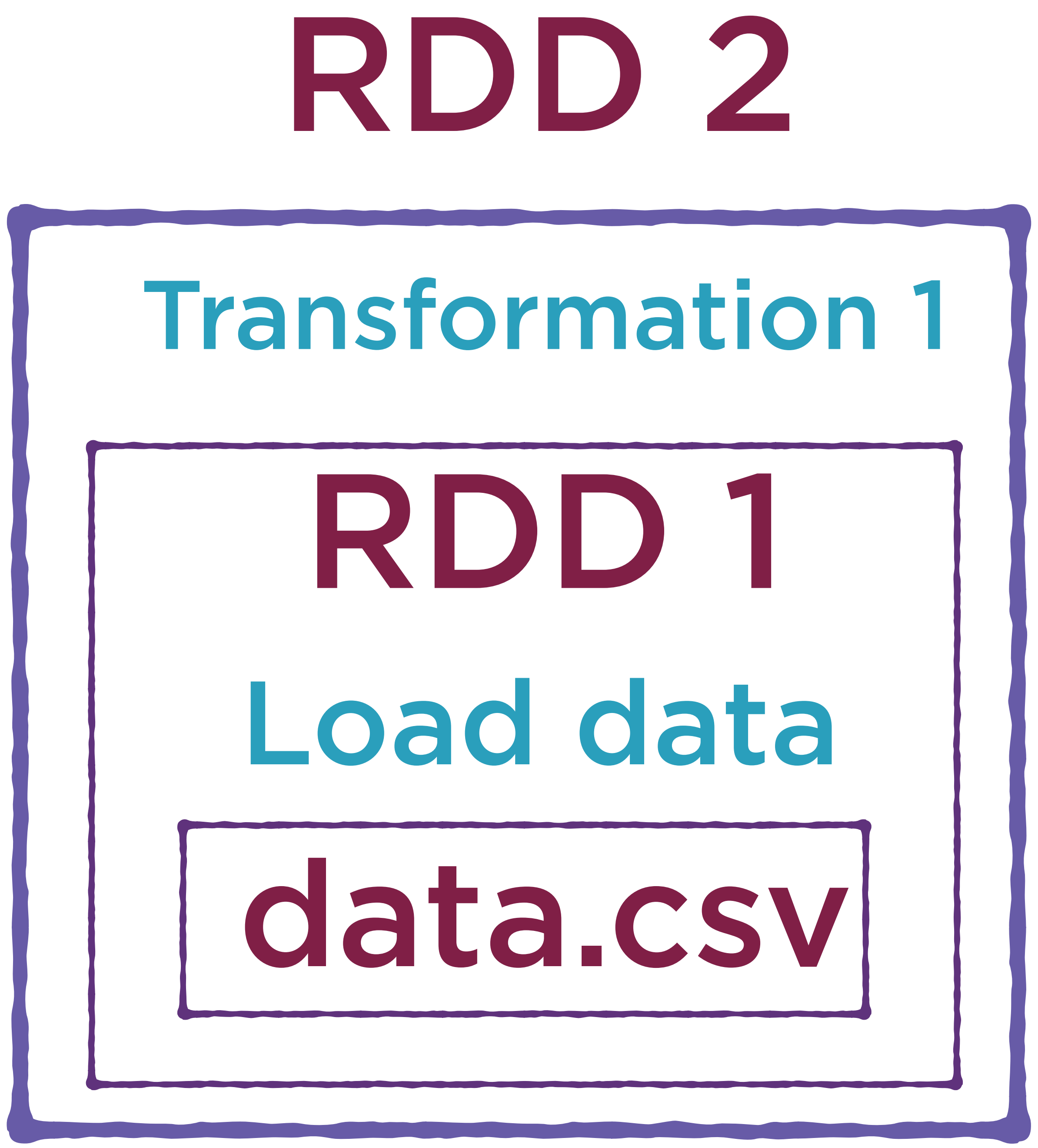
Lineage

Lineage can be traced back all the way to the source



Lineage

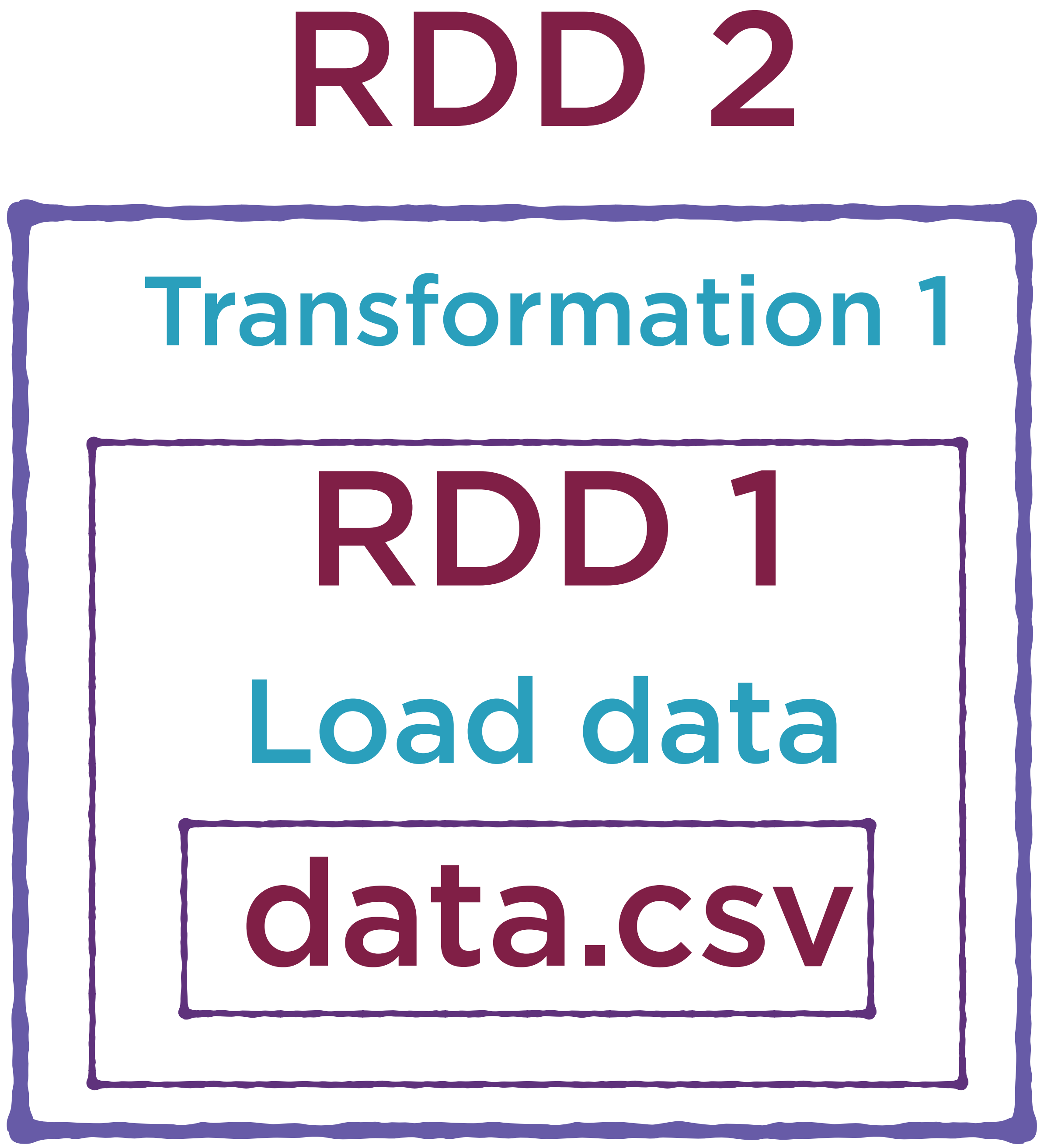
Lineage can be traced back all the way to the source



Lineage

When an action is requested on an RDD

All its parent RDDs are materialized



Lineage

Resilience

In-built fault tolerance

If something goes wrong, reconstruct from source

Lazy Evaluation

Materialize only when necessary

Summary

Understood the role of Spark in data analysis

Loading data from a file is a Spark transformation

Reading data from an RDD is a Spark Action

RDDs are partitioned, read-only collections which know their lineage

Spark transformations are lazily evaluated