cloudera

# Spark MLlib, GraphX Hands On and review of other options

Jayant Shekhar

Amina Abdulla

# Agenda

- Setup (15 min)
- MLlib (45 min)
- GraphX (15 min)
- Review of Other Options (10 min)
- Q & A (5 min)

# Setup

Locally With IntelliJ | With Spark Cluster

# Setup Options

1. Run it locally on your laptop with IntelliJ/Git/MVN

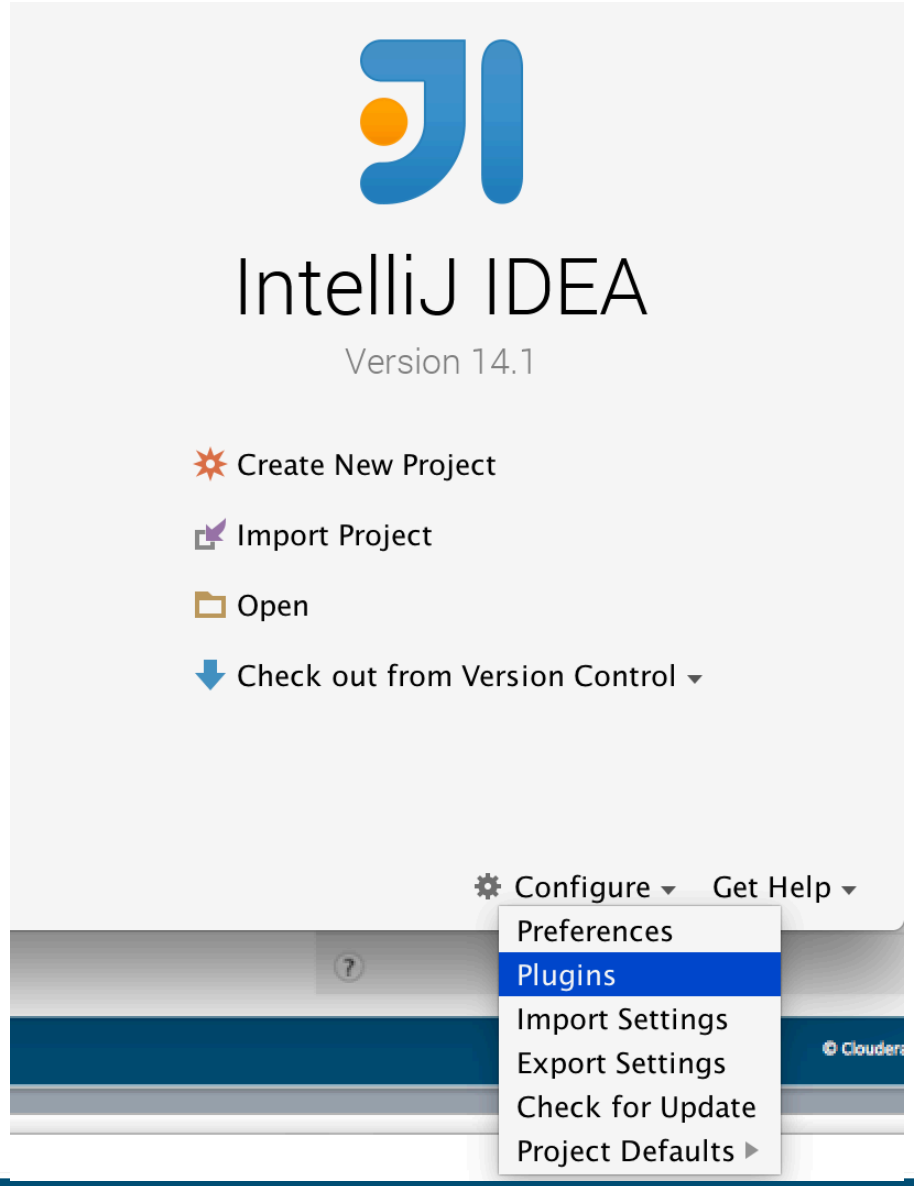
2. Run it on the cluster sateam-{1-4}.vpc.cloudera.com

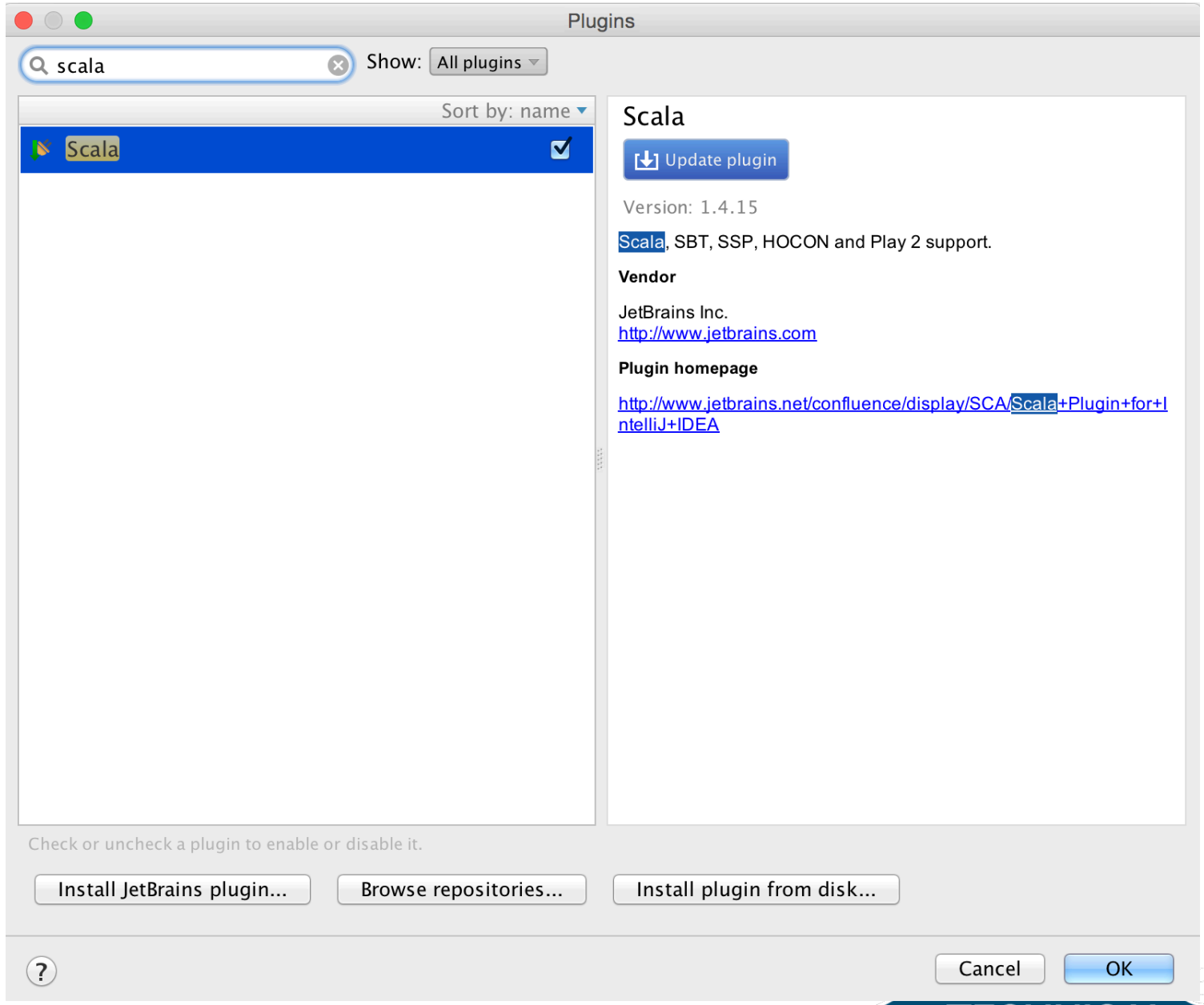
3. Run it on your own cluster. Need Spark/Git/Mvn

# Option 1 – Run it locally on your laptop

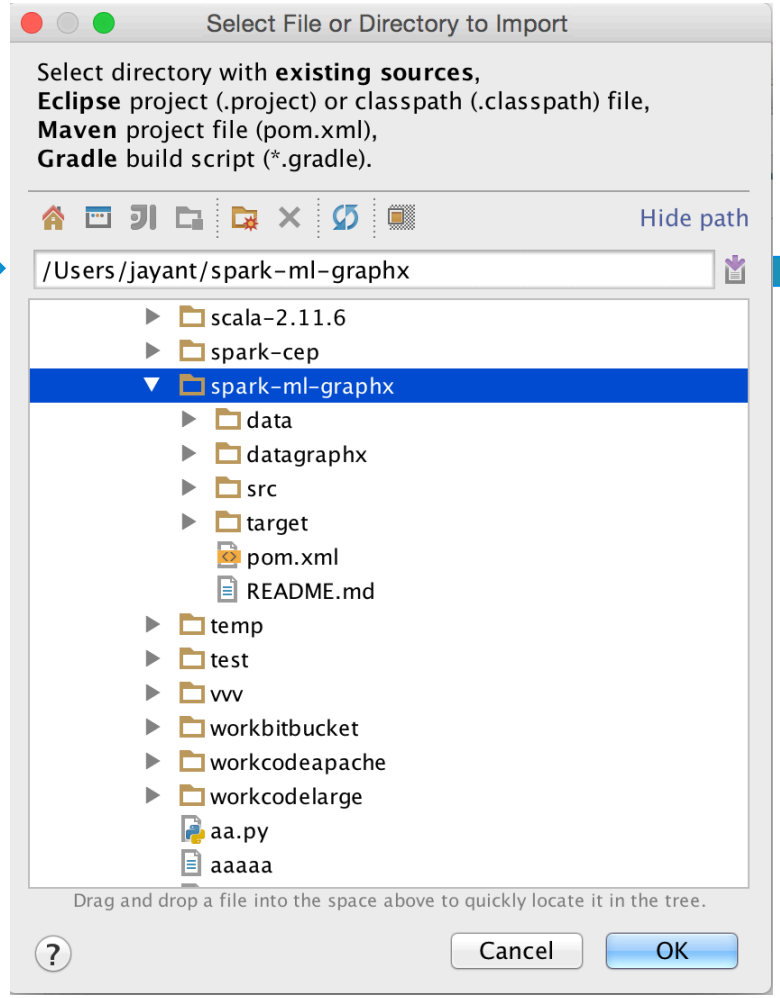
- Download and Install IntelliJ (14.1)
  - https://www.jetbrains.com/idea/download/
- Add the scala/sbt plugin
- Install git if you do not already have it
- Checkout Git Repo
  - git clone https://github.com/jayantshekhar/spark-ml-graphx.git
- Build spark-ml-graphx
  - cd spark-ml-graphx
  - mvn package
- Import as Maven project into IntelliJ

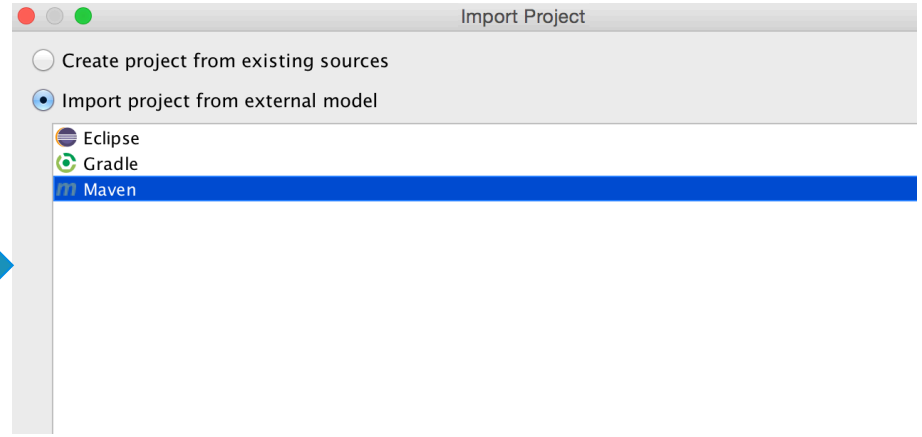
# Install Scala Plugin



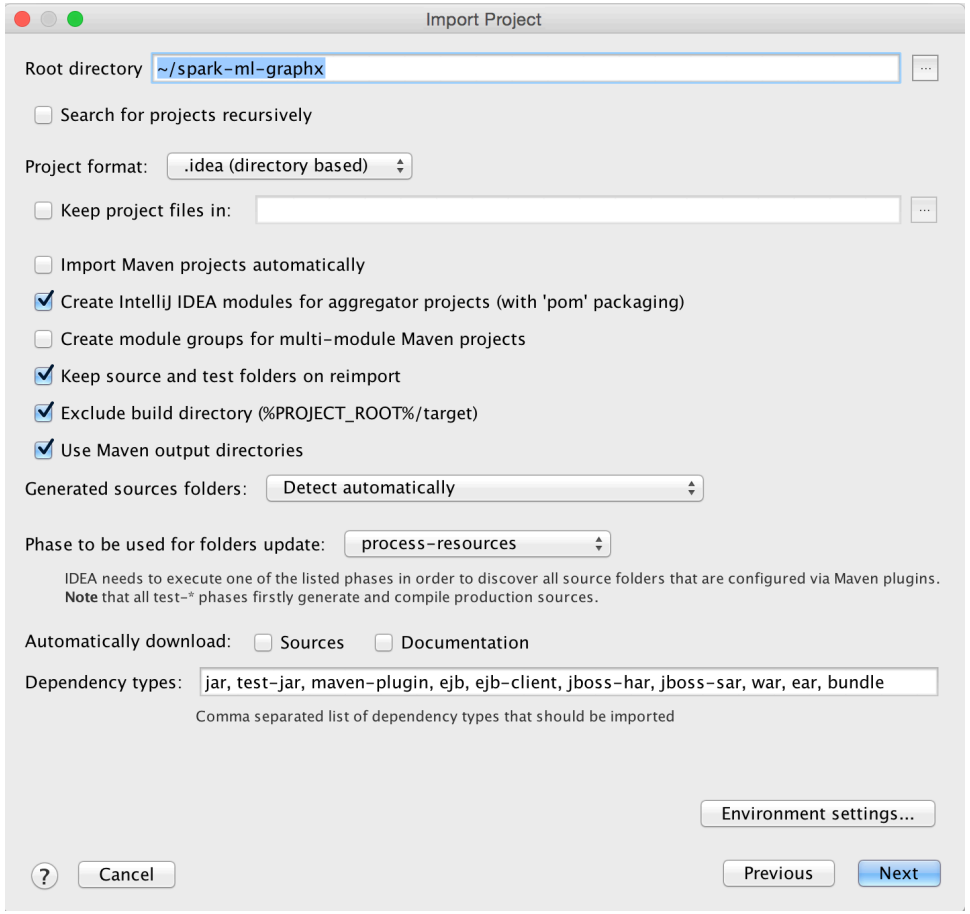



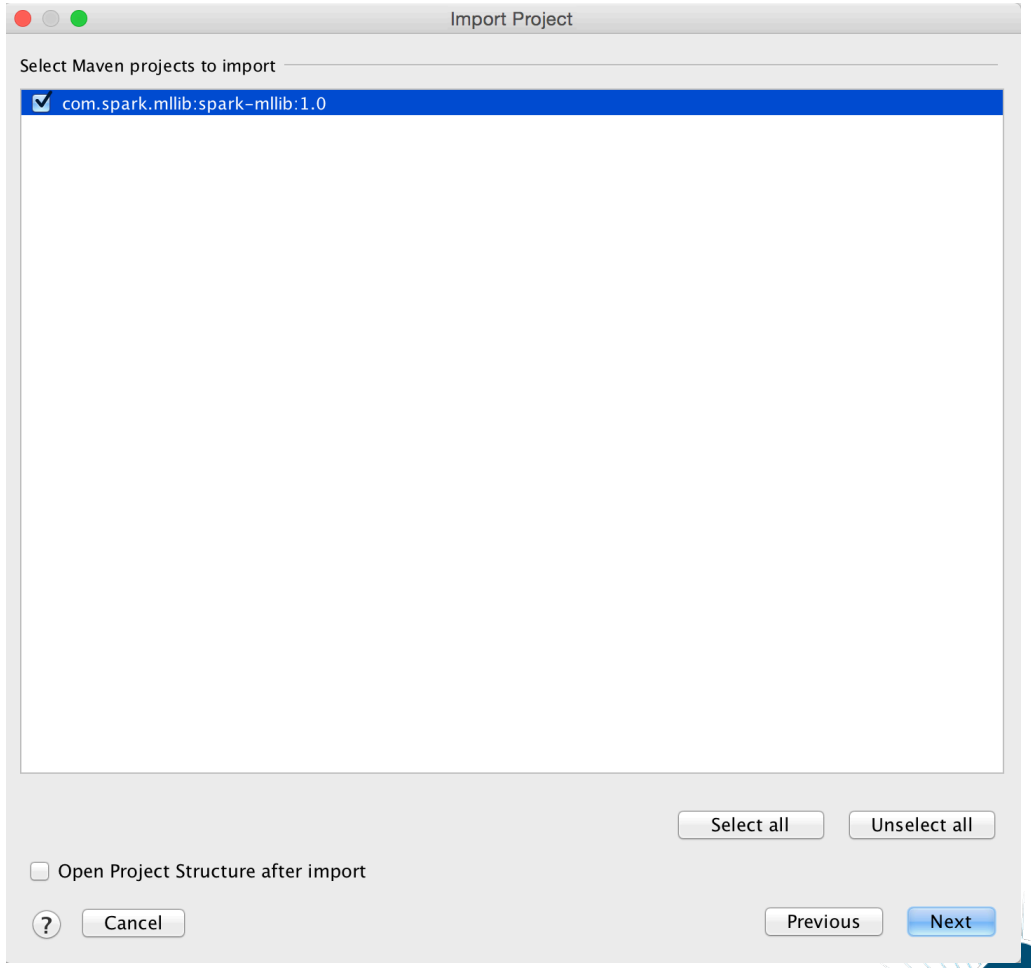
# Import Project into IntelliJ

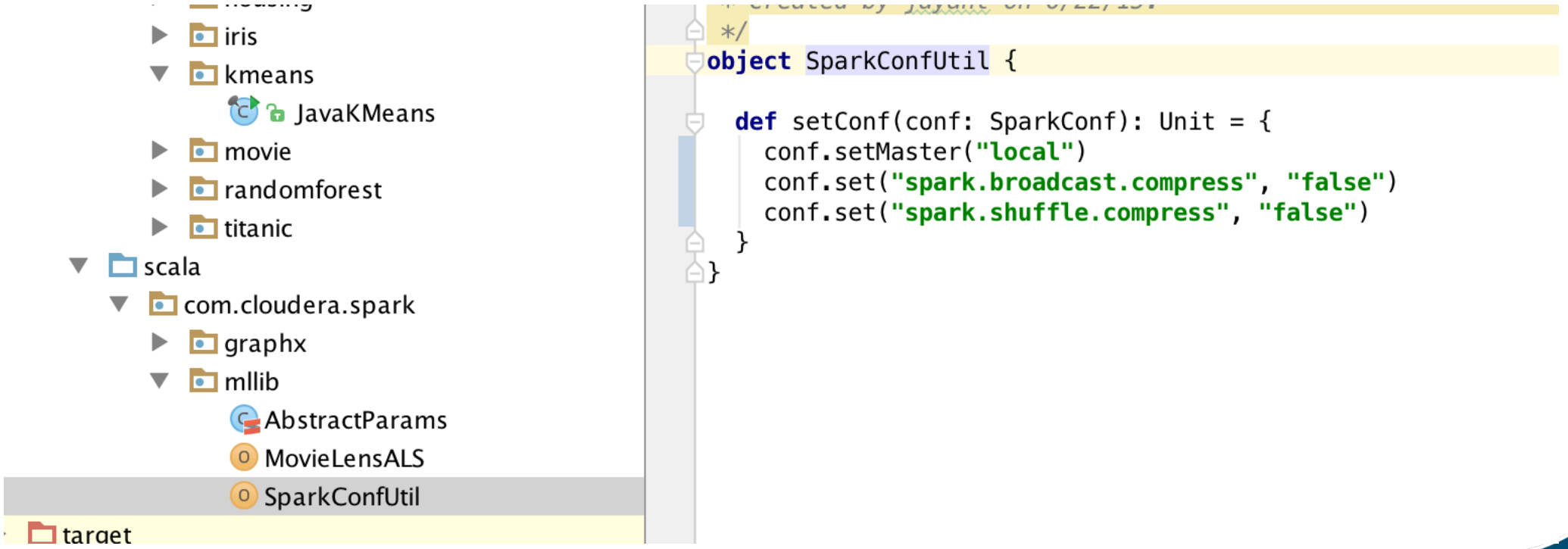
# Cont. - Import Project into IntelliJ



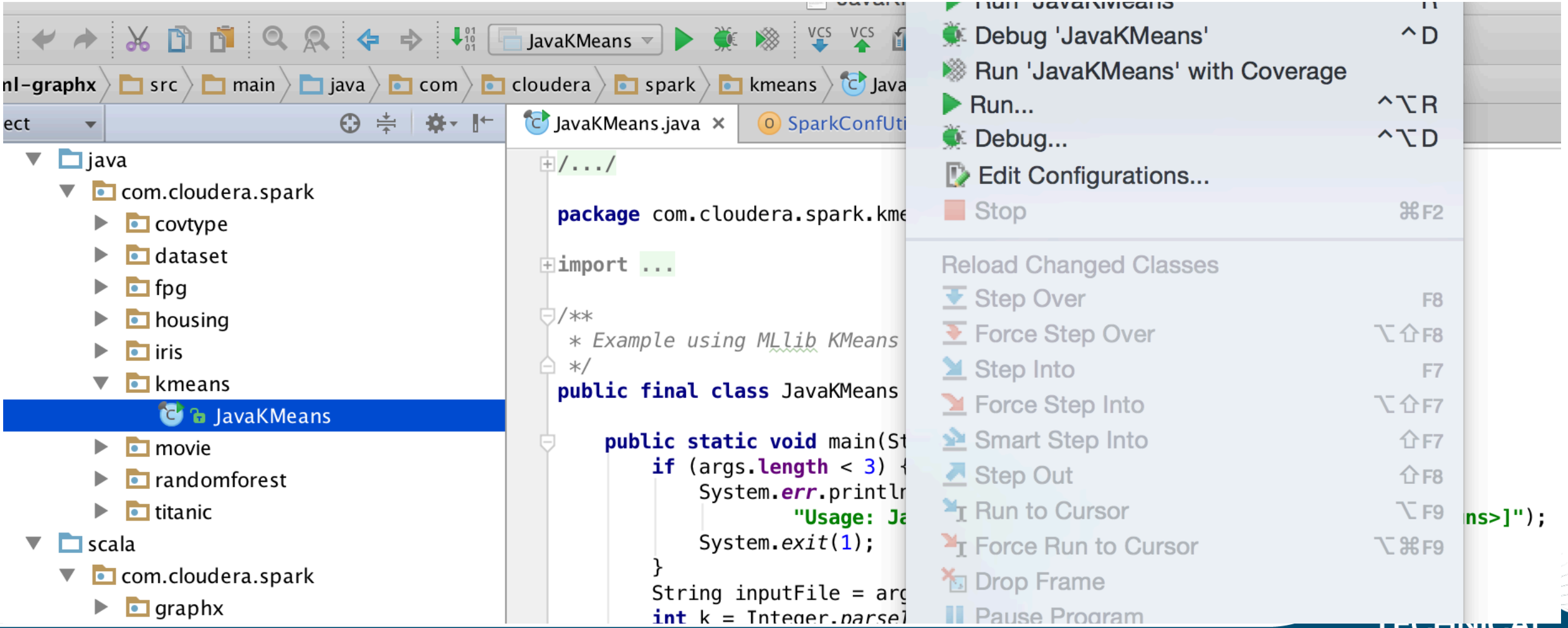
# Uncomment configuration settings for running locally

- Uncomment lines in **src/main/scala/com/cloudera/spark/mllib/SparkConfUtil.scala**

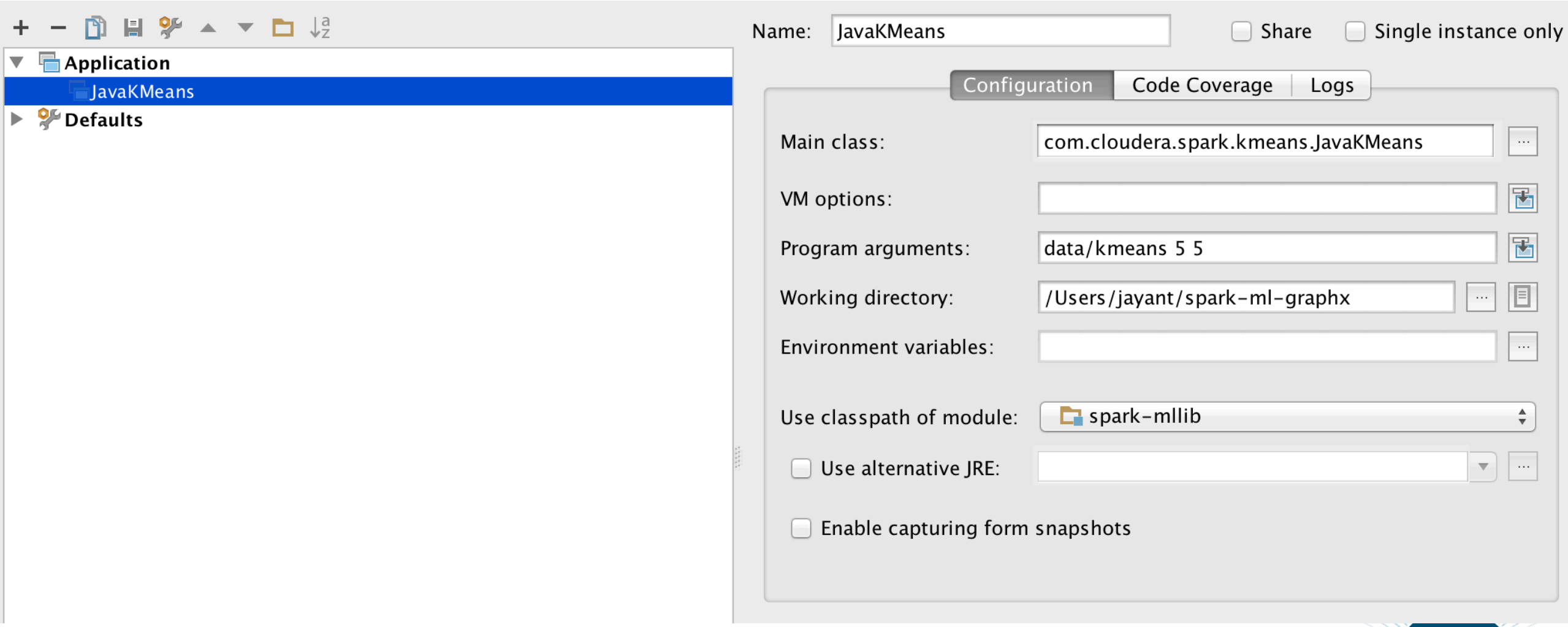


# Run KMeans

- **Run Kmeans it with the arguments : data/kmeans 5 5**

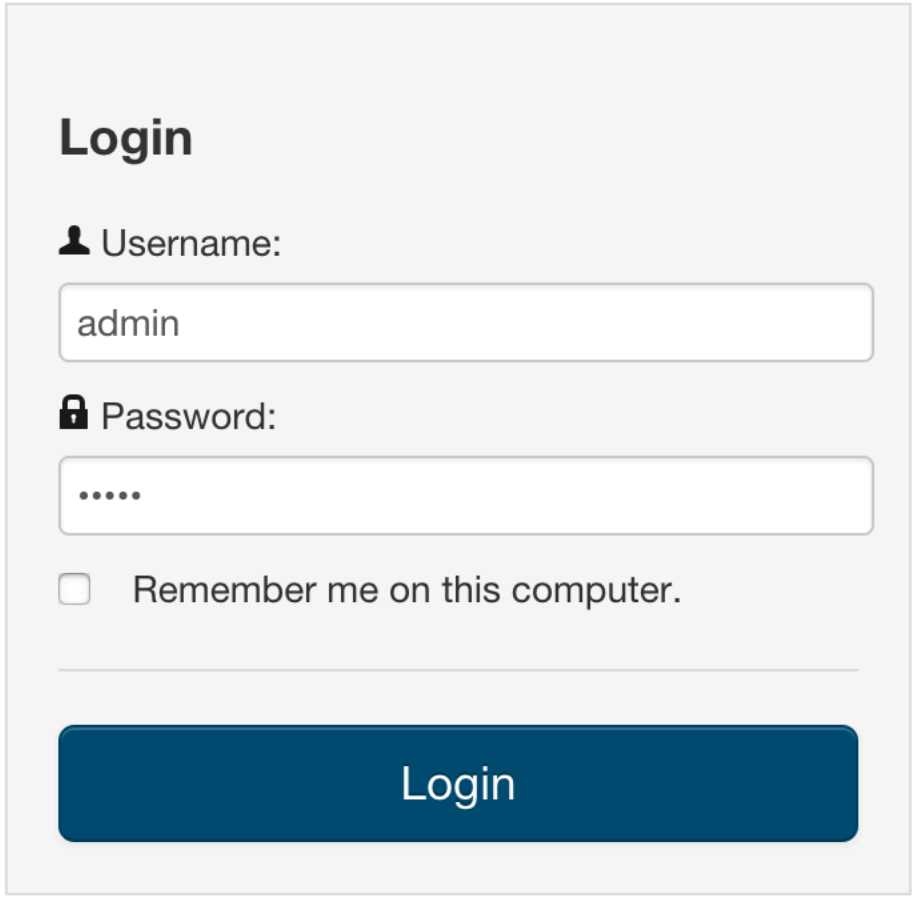


# Run KMeans… cont…



# Option 2 : Run it on the Cluster

# http://sateam-1.vpc.cloudera.com:7180/



# Cloudera Manager : http://sateam-1.vpc.cloudera.com:7180/cmf/home

| Status | Name | IP | Roles | Last Heartbeat | Load Average | Disk Usage | Physical Memory |
|---|---|---|---|---|---|---|---|
| ● | sateam-1.vpc.cloudera.com | 172.28.199.116 | 41 Role(s) | 8s ago | 0.00 0.06 0.15 | 15.7 GiB / 66.8 GiB | 3.8 GiB / 7.3 GiB |
| ● | sateam-2.vpc.cloudera.com | 172.28.195.12 | 13 Role(s) | 927ms ago | 0.05 0.02 0.00 | 9.8 GiB / 66.8 GiB | 813.8 MiB / 7.3 GiB |
| ● | sateam-3.vpc.cloudera.com | 172.28.199.15 | 13 Role(s) | 730ms ago | 0.00 0.00 0.00 | 9.8 GiB / 66.8 GiB | 825.9 MiB / 7.3 GiB |
| ● | sateam-4.vpc.cloudera.com | 172.28.196.119 | 13 Role(s) | 14.49s ago | 0.00 0.01 0.00 | 9.8 GiB / 66.8 GiB | 826.5 MiB / 7.3 GiB |

Display 25

# Option 2 – Run it on sateam-1.vpc.cloudera.com

- VPN into Cloudera
- ssh summit2015@sateam-1.vpc.cloudera.com
  - Password : summit2015
- mkdir <my name>
- cd <my name>
- Checkout Git Repo
  - git clone https://github.com/jayantshekhar/spark-ml-graphx.git
- cd spark-ml-graphx
- mvn package
- more README.md
- spark-submit --class com.cloudera.spark.kmeans.JavaKMeans --master yarn target/spark-recipes-1.0.jar data/kmeans 5 5

Directory Structure

- /home/summit2015
  - jayant
    - spark-ml-graphx
  - <.....>

# Spark History Server : http://sateam-1.vpc.cloudera.com:18088/

Spark 1.3.0 **History Server**

**Event log directory:** hdfs://sateam-1.vpc.cloudera.com:8020/user/spark/applicationHistory

**Showing 1-11 of 11**

| App ID | App Name | Started | Completed | Duration | Spark User | Last Updated |
|---|---|---|---|---|---|---|
| application_1438162344988_0015 | JavaHousing | 2015/07/29 06:31:39 | 2015/07/29 06:32:13 | 34 s | summit2015 | 2015/07/29 06:32:13 |
| application_1438162344988_0014 | JavaIris | 2015/07/29 06:22:05 | 2015/07/29 06:22:42 | 38 s | summit2015 | 2015/07/29 06:22:42 |
| application_1438162344988_0013 | JavaMovieLensALS | 2015/07/29 05:49:15 | 2015/07/29 05:50:34 | 1.3 min | summit2015 | 2015/07/29 05:50:34 |
| application_1438162344988_0011 | JavaMovieLensALS | 2015/07/29 05:40:16 | 2015/07/29 05:40:42 | 27 s | summit2015 | 2015/07/29 05:40:43 |
| application_1438162344988_0010 | JavaMovieLensALS | 2015/07/29 05:37:25 | 2015/07/29 05:37:52 | 27 s | summit2015 | 2015/07/29 05:37:52 |
| application_1438162344988_0009 | JavaMovieLensALS | 2015/07/29 05:32:15 | 2015/07/29 05:32:42 | 26 s | summit2015 | 2015/07/29 05:32:42 |
| application_1438162344988_0006 | JavaKMeans | 2015/07/29 04:39:35 | 2015/07/29 04:39:59 | 24 s | summit2015 | 2015/07/29 04:39:59 |
| application_1438162344988_0005 | JavaKMeans | 2015/07/29 04:08:30 | 2015/07/29 04:08:55 | 24 s | summit2015 | 2015/07/29 04:08:55 |
| application_1438162344988_0004 | JavaKMeans | 2015/07/29 03:25:12 | 2015/07/29 03:25:36 | 24 s | summit2015 | 2015/07/29 03:25:36 |
| application_1438162344988_0003 | JavaKMeans | 2015/07/29 03:21:50 | 2015/07/29 03:22:15 | 25 s | summit2015 | 2015/07/29 03:22:15 |
| application_1438162344988_0001 | JavaKMeans | 2015/07/29 02:33:44 | 2015/07/29 02:34:18 | 34 s | offsite | 2015/07/29 02:34:18 |

# MLlib

## Data Types, Basic Statistics, Feature Selection & Algorithms

https://spark.apache.org/docs/1.3.0/mllib-guide.html

- Data Types
- Basic Statistics
- Feature Extraction & Transformation
- Dimensionality Reduction
  - SVD
  - PCA
- Optimization
  - SGD
  - L-BFGS

- Local vector
- Labeled point
- Local matrix
- Distributed matrix
  - BlockMatrix
  - RowMatrix
  - IndexedRowMatrix
  - CoordinateMatrix

- Summary statistics
- Correlations
- Stratified sampling
- Hypothesis testing
- Random data generation

- TF-IDF
- Word2Vec
- StandardScaler
- Normalizer
- Feature Selection : ChiSqSelector

- Classification & Regression
  - Linear Models (SVMs, Linear Regression, Logistic Regression)
  - Naïve Bayes
  - Decision Tree
  - Ensembles
    - Random Forests
    - Gradient Boosted Trees
- Clustering
  - K-Means
  - Gaussian Mixture
  - Power Iteration Clustering
- Collaborative Filtering
  - ALS
- Frequent Pattern Mining

# MLlib

## Exercises

# MLlib Exercises

- KMeans : KMeans Clustering
- Movie Lens
- Frequent Parallel Growth
- Random Forest
- Iris
- Housing
- Random Forest
- Titanic

# Datasets – spark-ml-graphx/data

| Dataset | Details |
|---|---|
| covtype | |
| fpg | |
| housing | |
| iris | |
| kmeans | |

| Dataset | Details |
|---|---|
| movielens | |
| movielens_small | |
| mtcars | |
| svm | |
| titanic | |

# KMeans

0.0 0.0 0.0
0.1 0.1 0.1
0.2 0.2 0.2
9.0 9.0 9.0
9.1 9.1 9.1
9.2 9.2 9.2

Create RDD of Vector

Train the Kmeans Model

Cluster centers:
[0.1,0.1,0.1]
[9.1,9.1,9.1]

Print Cluster Centers

Compute the Cost

| Data | data/kmeans |
|---|---|
| Program Arguments | data/kmeans 2 5 |

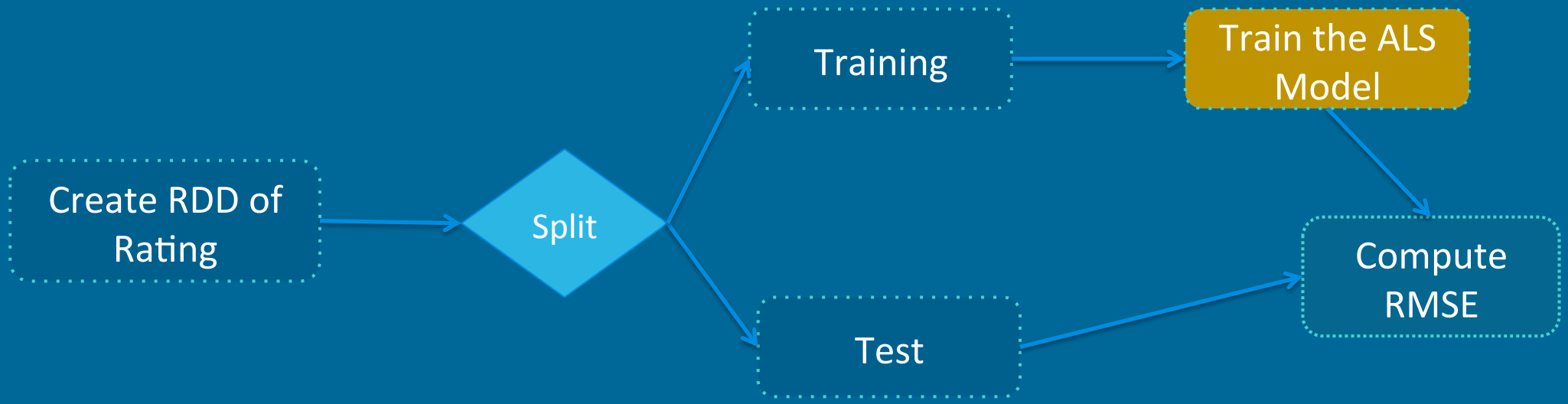
Input File

k

Max iterations

# MovieLens

```
0::2::3
0::3::1
0::5::2
```




| Data | data/movielens_small |
|---|---|
| Program Arguments | data/movielens_small 5 5 |

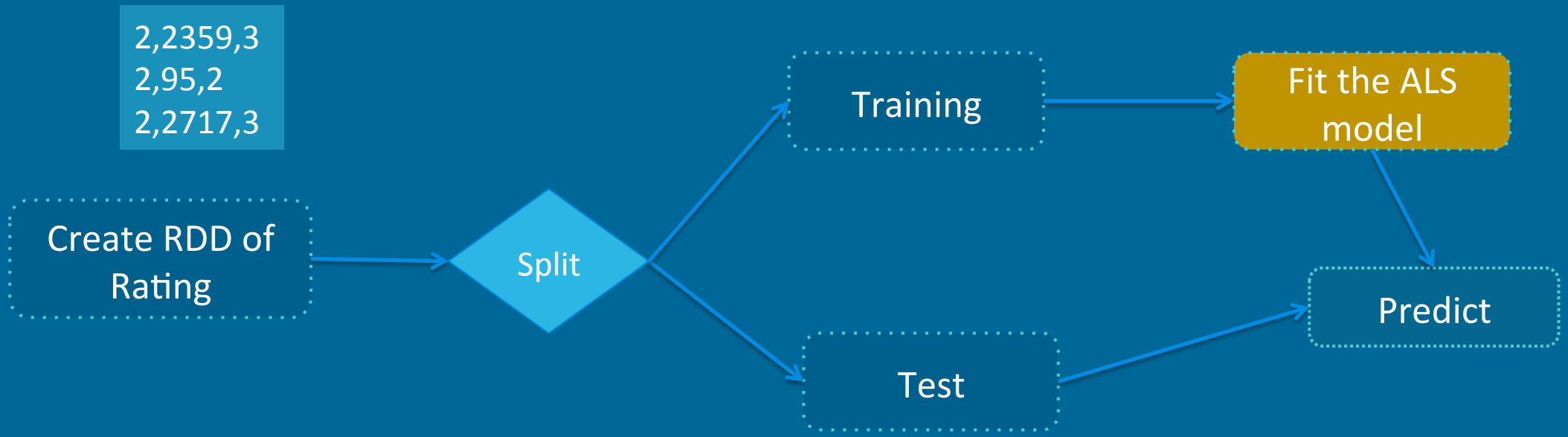
Input File

rank

Max iterations

```
Cluster centers:
[0.1,0.1,0.1]
 [9.1,9.1,9.1]
```

# MovieLens with DataFrames and ml




| Data | data/movielens/ratings |
|---|---|
| Program Arguments | data/movielens/ratings 5 5 |

Input File

rank

Max iterations

# *Iris*

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
```

Create RDD of Vector

Summary Statistics

Calculate Correlation

Cluster

| **Data** | **data/iris** |
|---|---|
| Program Arguments | data/iris 3 5 |

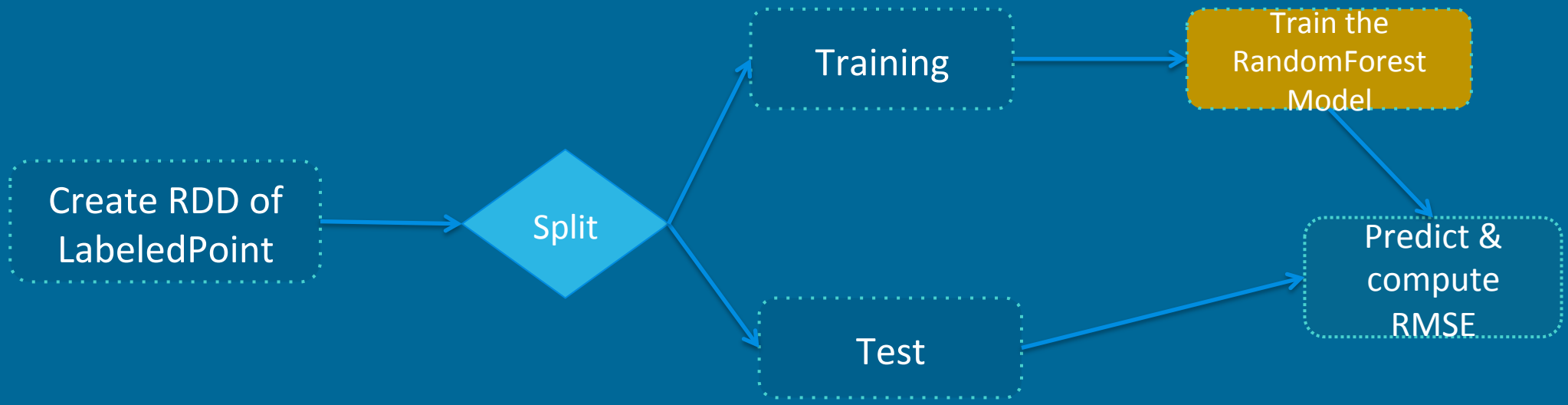
Input File

k

Max iterations

# Covtype

2596,51,3,258,0,510,221,232,148,6279,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,5
2590,56,2,212,-6,390,220,235,151,6225,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,5
2804,139,9,268,65,3180,234,238,135,6121,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,2




| Data | data/covtype |
|---|---|
| Program Arguments | data/covtype |

Input File

rank

Max iterations

# FPGrowth

r z h k p
z y x w v u t s
s x o n r
x z y m t s q e

Create RDD of ArrayList<String>

Run FPGrowth

[s], 3
[s,x], 3
[s,x,z], 2
[s,z], 2
[r], 3
[r,x], 2
[r,z], 2
[y], 3
[y,s], 2
[y,s,x], 2

Print Results

| Data | data/fpg |
| --- | --- |
| Program Arguments | data/fpg |

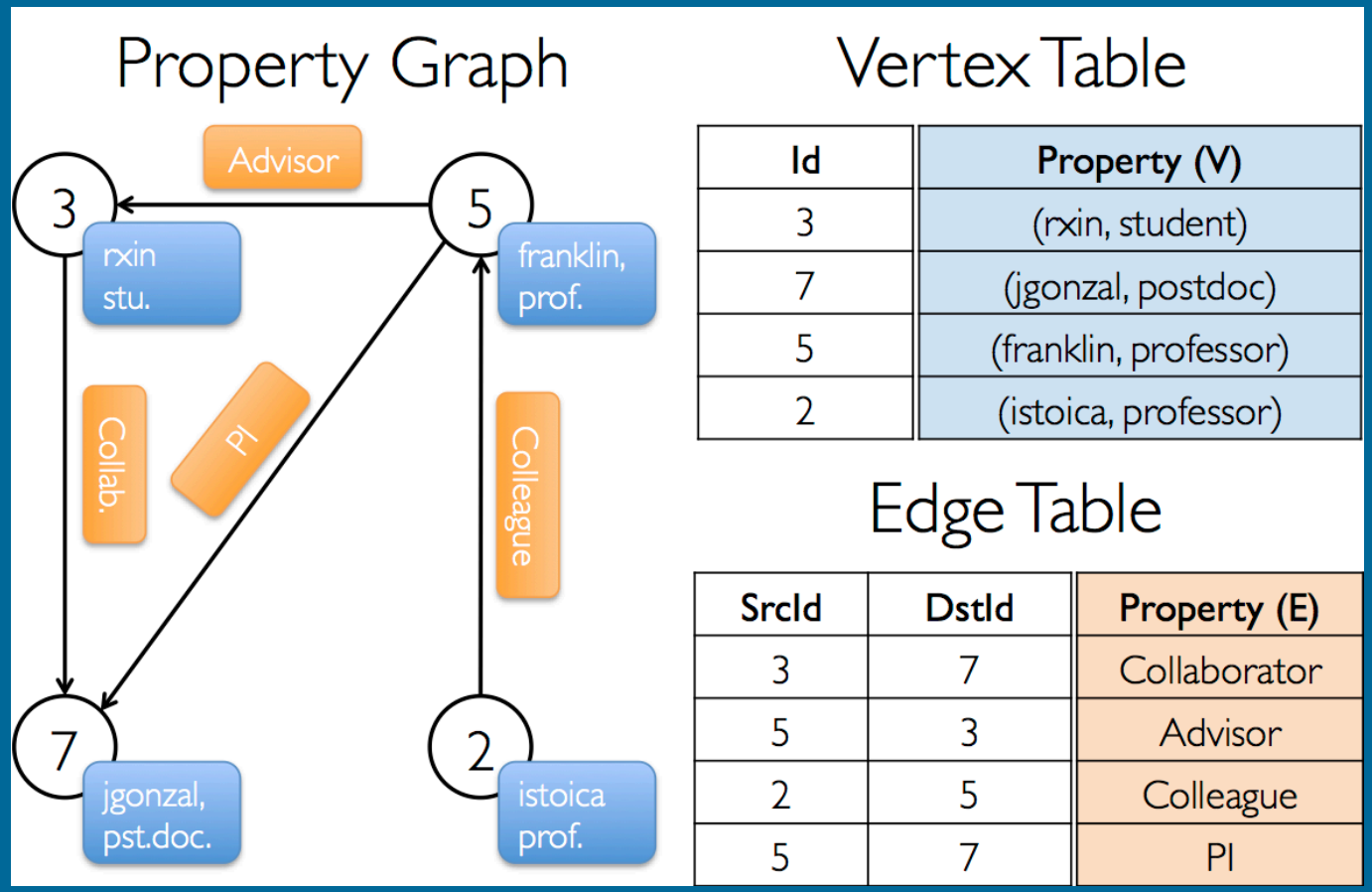
Input File

# GraphX

Property Graph, Graph Operators, Pregel API, Graph Builders, Vertex and Edge RDDs, Graph Algorithms

# Property Graph

Directed multigraph with user defined objects attached to each vertex and edge




Vertex Table

| Id | Property (V) |
|---|---|
| 3 | (rxin, student) |
| 7 | (jgonzal, postdoc) |
| 5 | (franklin, professor) |
| 2 | (istoica, professor) |

Edge Table

| SrcId | DstId | Property (E) |
|---|---|---|
| 3 | 7 | Collaborator |
| 5 | 3 | Advisor |
| 2 | 5 | Colleague |
| 5 | 7 | PI |

```
class Graph[VD, ED] {
  val vertices: VertexRDD[VD]
  val edges: EdgeRDD[ED]
}
```

VertexRDD[VD] and EdgeRDD[ED] extend and are optimized versions of RDD[(VertexID, VD)] and RDD[Edge[ED]]

Type Signature:

val userGraph: Graph[(String, String), String]

# Graph Operators : property graphs have a collection of basic operators that take user defined functions and produce new graphs with transformed properties and structure

```
// Information about the Graph ==================================================
val numEdges: Long
val numVertices: Long
val inDegrees: VertexRDD[Int]
val outDegrees: VertexRDD[Int]
val degrees: VertexRDD[Int]
// Views of the graph as collections ============================================
val vertices: VertexRDD[VD]
val edges: EdgeRDD[ED]
val triplets: RDD[EdgeTriplet[VD, ED]]
// Functions for caching graphs =================================================
def persist(newLevel: StorageLevel = StorageLevel.MEMORY_ONLY): Graph[VD, ED]
def cache(): Graph[VD, ED]
def unpersistVertices(blocking: Boolean = true): Graph[VD, ED]
// Change the partitioning heuristic  ===========================================
def partitionBy(partitionStrategy: PartitionStrategy): Graph[VD, ED]
// Transform vertex and edge attributes =========================================
def mapVertices[VD2](map: (VertexID, VD) => VD2): Graph[VD2, ED]
def mapEdges[ED2](map: Edge[ED] => ED2): Graph[VD, ED2]
def mapEdges[ED2](map: (PartitionID, Iterator[Edge[ED]]) => Iterator[ED2]): Graph[VD, ED2]
def mapTriplets[ED2](map: EdgeTriplet[VD, ED] => ED2): Graph[VD, ED2]
def mapTriplets[ED2](map: (PartitionID, Iterator[EdgeTriplet[VD, ED]]) => Iterator[ED2])
  : Graph[VD, ED2]
```

```scala
// Modify the graph structure ================================================================
def reverse: Graph[VD, ED]
def subgraph(
    epred: EdgeTriplet[VD,ED] => Boolean = (x => true),
    vpred: (VertexID, VD) => Boolean = ((v, d) => true))
  : Graph[VD, ED]
def mask[VD2, ED2](other: Graph[VD2, ED2]): Graph[VD, ED]
def groupEdges(merge: (ED, ED) => ED): Graph[VD, ED]
// Join RDDs with the graph ==================================================================
def joinVertices[U](table: RDD[(VertexID, U)])(mapFunc: (VertexID, VD, U) => VD): Graph[VD, ED]
def outerJoinVertices[U, VD2](other: RDD[(VertexID, U)])
    (mapFunc: (VertexID, VD, Option[U]) => VD2)
  : Graph[VD2, ED]
// Aggregate information about adjacent triplets =============================================
def collectNeighborIds(edgeDirection: EdgeDirection): VertexRDD[Array[VertexID]]
def collectNeighbors(edgeDirection: EdgeDirection): VertexRDD[Array[(VertexID, VD)]]
def aggregateMessages[Msg: ClassTag](
    sendMsg: EdgeContext[VD, ED, Msg] => Unit,
    mergeMsg: (Msg, Msg) => Msg,
    tripletFields: TripletFields = TripletFields.All)
  : VertexRDD[A]
// Iterative graph-parallel computation ======================================================
def pregel[A](initialMsg: A, maxIterations: Int, activeDirection: EdgeDirection)(
    vprog: (VertexID, VD, A) => VD,
    sendMsg: EdgeTriplet[VD, ED] => Iterator[(VertexID,A)],
    mergeMsg: (A, A) => A)
  : Graph[VD, ED]
```

```
// Basic graph algorithms
================================================================================
def pageRank(tol: Double, resetProb: Double = 0.15): Graph[Double, Double]
def connectedComponents(): Graph[VertexID, ED]
def triangleCount(): Graph[Int, ED]
def stronglyConnectedComponents(numIter: Int): Graph[VertexID, ED]
```

# Graph Builders

GraphLoader.edgeListFile provides a way to load a graph from a list of edges on disk. It parses an adjacency list of (source vertex ID, destination vertex ID) pairs of the following form, skipping comment lines that begin with #:

```
# This is a comment
2 1
4 1
1 2
```

It creates a Graph from the specified edges, automatically creating any vertices mentioned by edges.

# Pregel API

- Many important graph algorithms iteratively recompute the properties of each vertex until a fixed-point condition is reached.
- At a high level the Pregel operator in GraphX is a bulk-synchronous parallel messaging abstraction
- The Pregel operator executes in a series of super steps in which vertices receive the sum of their inbound messages from the previous super step, compute a new value for the vertex property, and then send messages to neighboring vertices in the next super step

# Single Source Shortest Path

```scala
import org.apache.spark.graphx._
// Import random graph generation library
import org.apache.spark.graphx.util.GraphGenerators
// A graph with edge attributes containing distances
val graph: Graph[Int, Double] =
  GraphGenerators.logNormalGraph(sc, numVertices = 100).mapEdges(e => e.attr.toDouble)
val sourceId: VertexId = 42 // The ultimate source
// Initialize the graph such that all vertices except the root have distance infinity.
val initialGraph = graph.mapVertices((id, _) => if (id == sourceId) 0.0 else Double.PositiveInfin
ity)
val sssp = initialGraph.pregel(Double.PositiveInfinity)(
  (id, dist, newDist) => math.min(dist, newDist), // Vertex Program
  triplet => {  // Send Message
    if (triplet.srcAttr + triplet.attr < triplet.dstAttr) {
      Iterator((triplet.dstId, triplet.srcAttr + triplet.attr))
    } else {
      Iterator.empty
    }
  },
  (a,b) => math.min(a,b) // Merge Message
  )
println(sssp.vertices.collect.mkString("\n"))
```

# Graph Algorithms

- Algorithms are contained in the org.apache.spark.graphx.lib package
- They can be accessed directly as methods on Graph via GraphOps
- Algorithms
  - PageRank
  - Connected Components
    - labels each connected component of the graph with the ID of its lowest-numbered vertex
  - Triangle Counting

# GraphX

Exercises

Page Rank

Wikipedia

V1

V2

Triangle

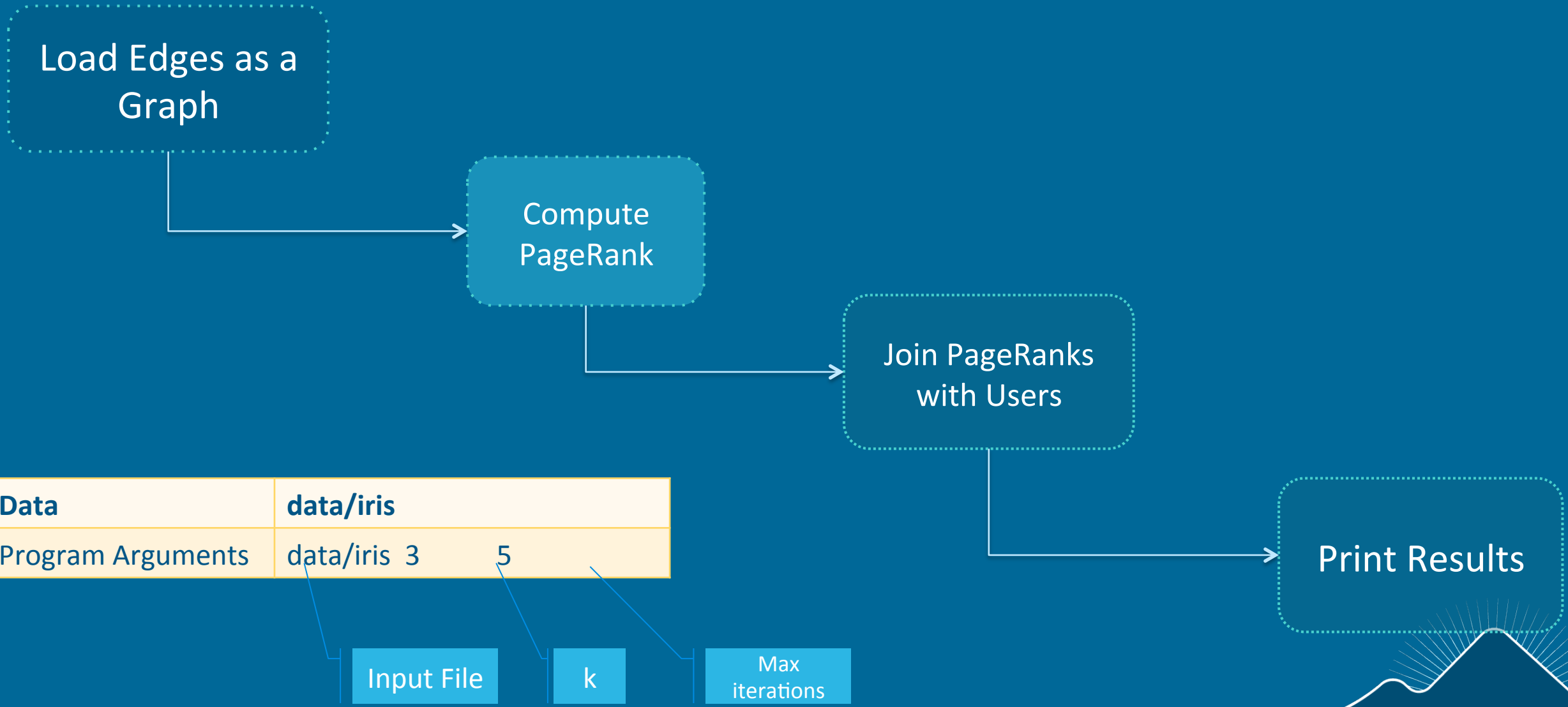
Connected Components

# Datasets – spark-ml-graphx/datagraphx

| Dataset | Details |
| --- | --- |
| followers | |
| sample_graph | |
| users | |
| wikipedia | |
| wikipediaV2 | |

# PageRank

```
2 1
4 1
1 2
```




| Data | data/iris |
|---|---|
| Program Arguments | data/iris 3 5 |

Input File

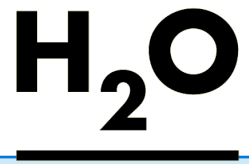
k

Max iterations

# Review of other options

## For Predictive and Graph Algorithms

# MLlib

| | |
|---|---|
|  | • Data Exploration & Visualization<br>• Model Building & Deployment on Hadoop. Does not use M/R |
|  | • M/R with R |
|  | • Runs on Spark & Hadoop |
|  | • Distributed Predictive Algorithms on M/R |

# GraphX

| | |
|---|---|
| Dato | • Cleaning Data, Developing Features, Training a Model, Creating and Maintaining a Predictive Service<br>• Analyzing Graphs : Finding Connected Components, PageRank, Define own computation on Graphs<br>• Visualizing Graphs |
| APACHE GIRAPH | • Implements Bulk Synchronous Processing (BSP) over Map Reduce<br>• Also adds Master Computation, Sharded Aggregators, Edge-Oriented Input, Out-of-Core Computation |
| | |

# Q & A


# Thank You!!!

Jayant Shekhar, Amina Abdulla