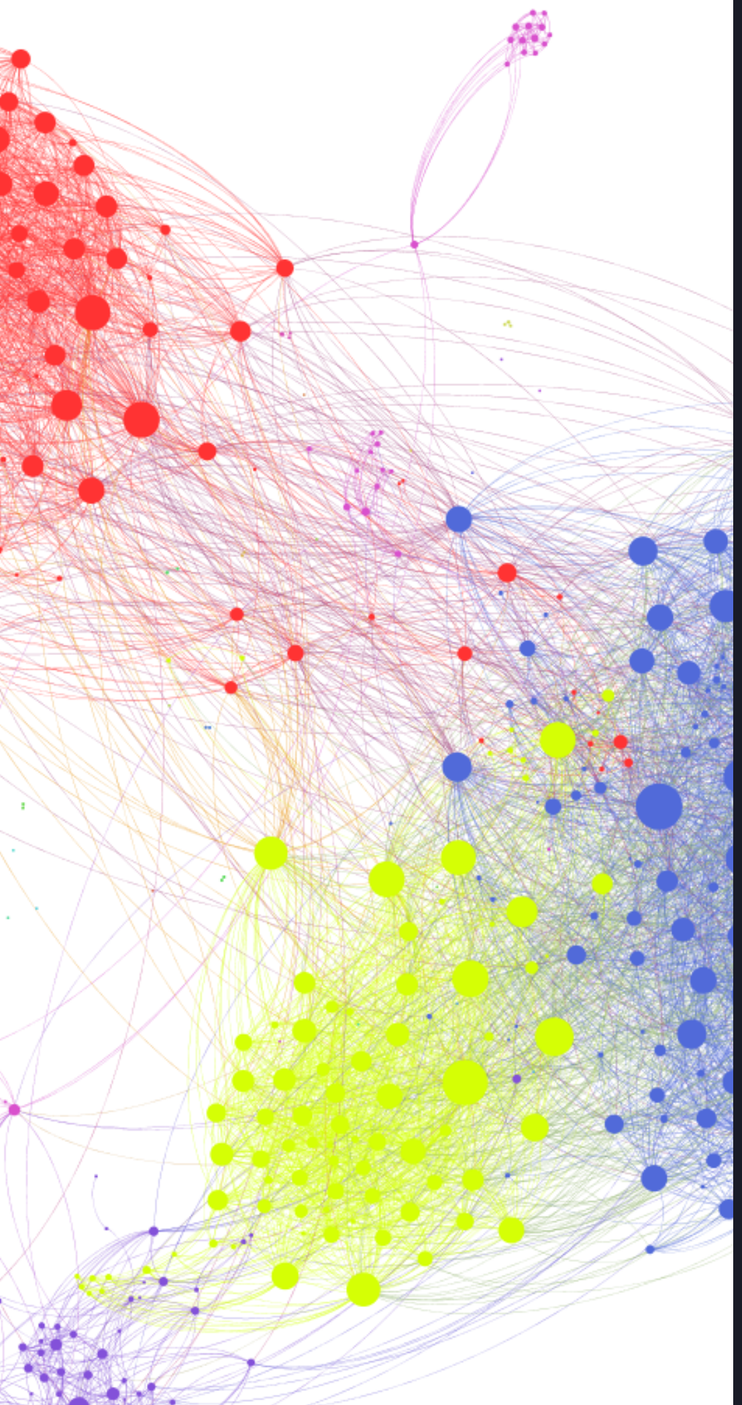

Twitter Social
Graph Analysis

Agenda

- INTRODUCTION TO GRAPHS
- TOP 3 ACCOUNTS BASED ON FOLLOWERS
- USER FOLLOWING MOST ACCOUNTS
- ADDITIONAL TOPICS RELATED TO GRAPHS

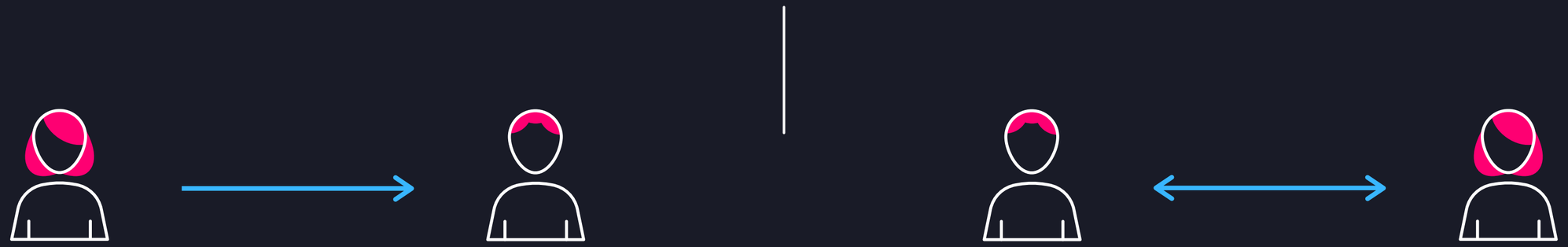
Introduction to graphs



WHAT IS A GRAPH?

A graph is a construct of vertices and edges. Edges can connect vertices in a directed or undirected way.

Difference between directed and undirected



TWITTER

FACEBOOK

Top 3 accounts based on followers

The importance is defined by the amount of followers!

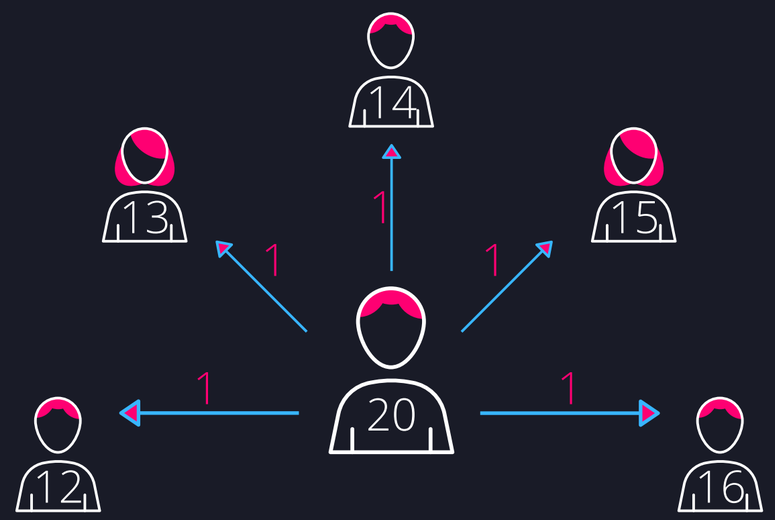
These are Twitters hot shots!

ID	Followers
10316422	2583
14206126	940
12750862	914

User following most accounts

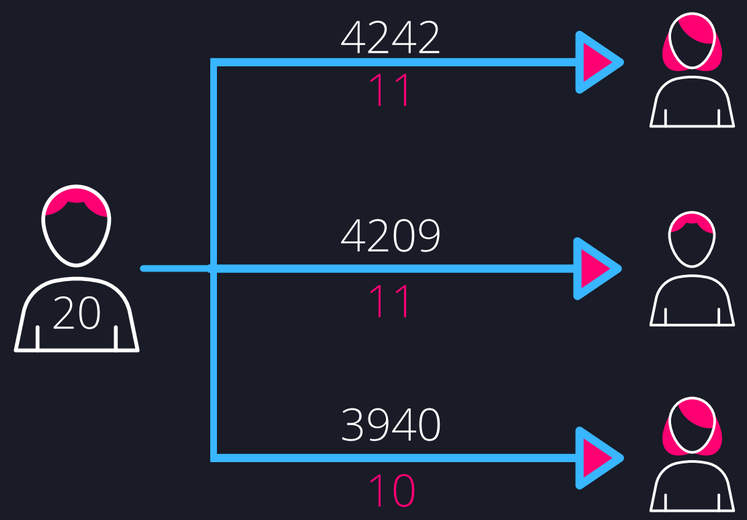

User 20
follows
1'213'787
accounts

Shortest path

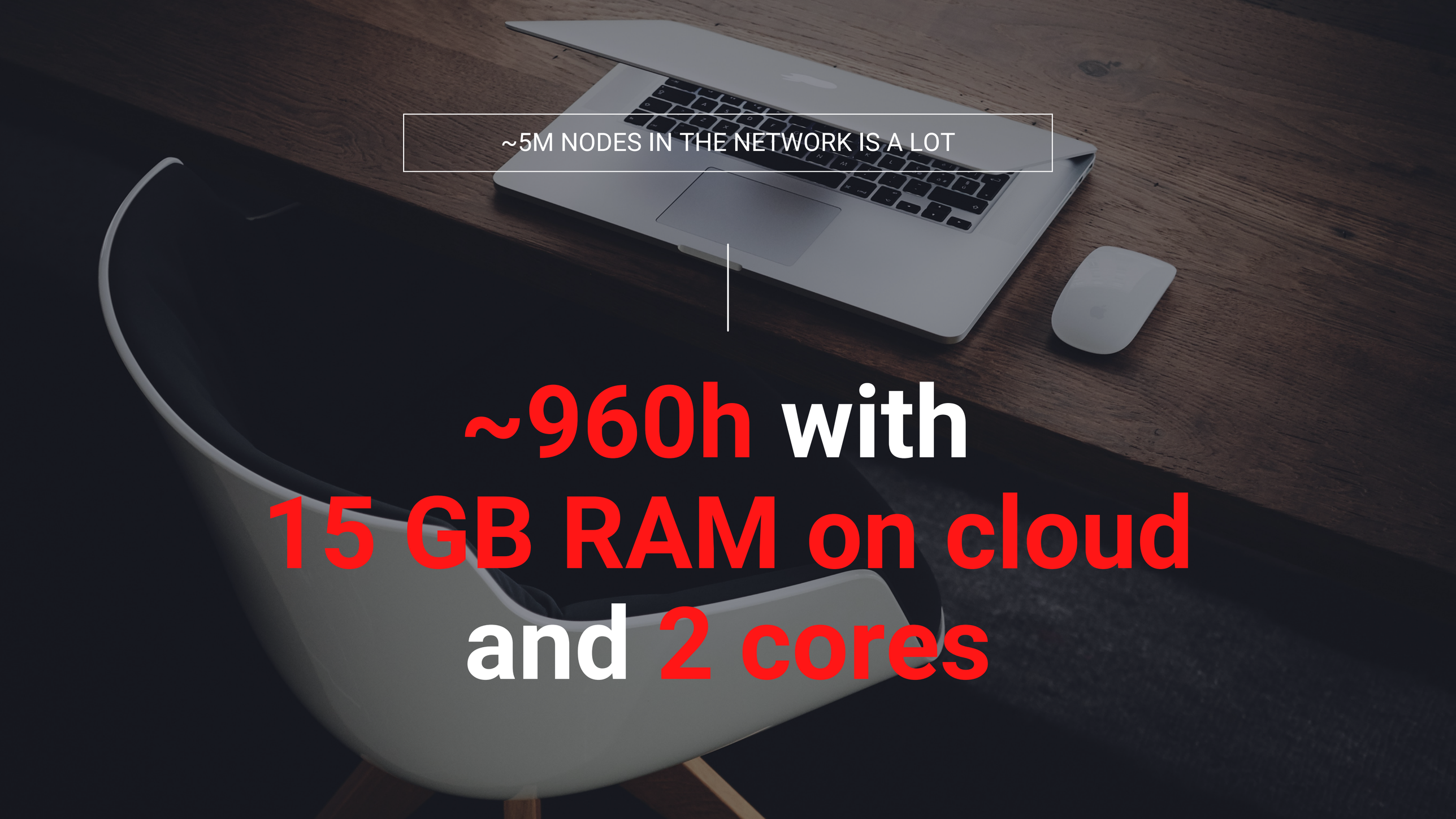


There is no one closer
than your neighbors

Longest path

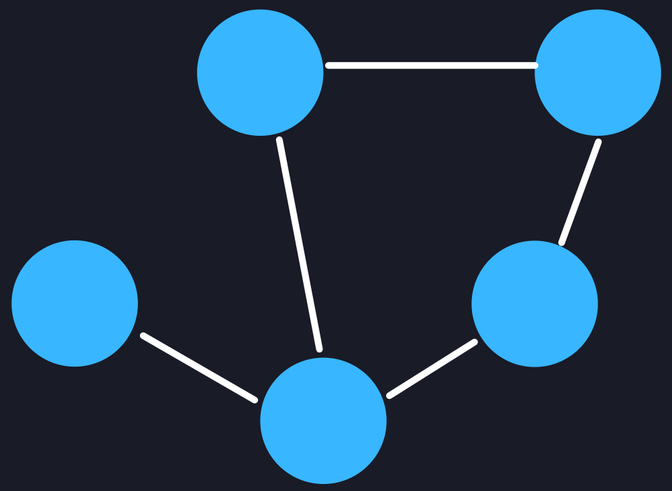


These results are only
representative for the first 4k nodes

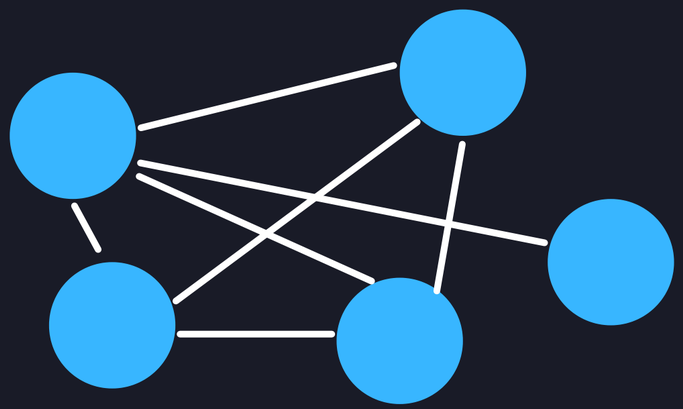

~5M NODES IN THE NETWORK IS A LOT
~960h with
15 GB RAM on cloud
and 2 cores

Additional topics related to graphs

TWITTER THE SPARSE GRAPH



Sparse



Dense

Politically Exposed Persons

Checking the Country, Legal identity and co-owned companies

Unusual transactions and timeframe

Amounts in form of weights and from where the money travelled

Synthetic Identities

Similar information that identifies several idividuals as one identity

Cycles

Money runs over several companies and returns to starting point

NEXT

Most followed accounts

- 10316422 / 14206126 / 12750862

Account following most accounts

- Account 20 follows most accounts
- Closest paths of lenght 1 are neighbors
- Longest path only possible with corresponding computation power

Additional Info

- People in a club as dense graph
- Relationship, Legal Identity, weights

Thank you!

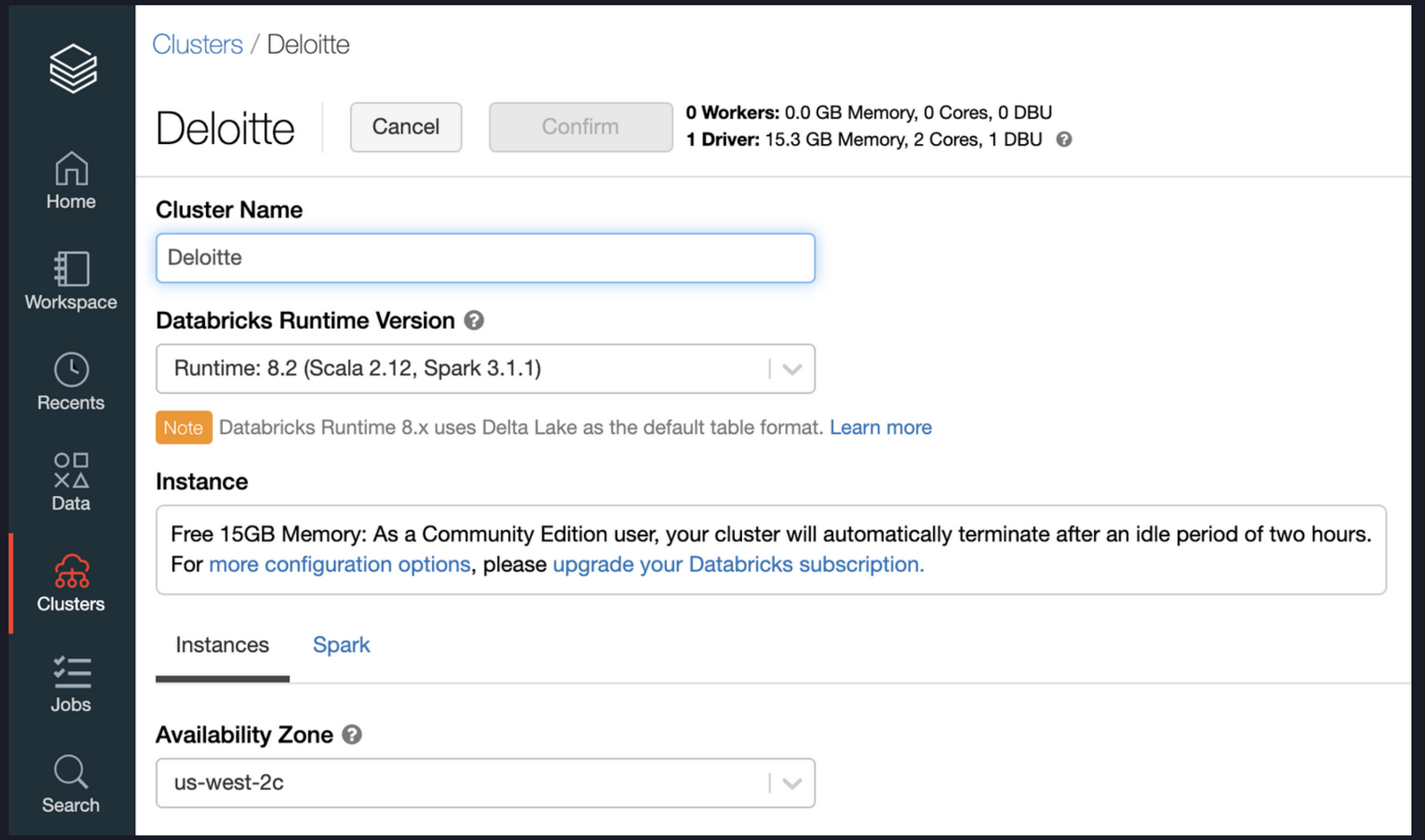
APPENDIX

DEFINITION OF DENSITY

$$0 \leq D \leq 1$$

$$D = \frac{|E|}{|V|(|V|-1)} = \begin{cases} \text{Sparse}, & 0<D<\frac{1}{2} \\ \text{Dense}, & \frac{1}{2}<D\leq 1 \end{cases}$$

WORKING IN THE CLOUD



LONGEST PATH WITH ALL

```scala
// Definition of the variables used in the loop
val a = vertices.rdd.map(r => r(0)).collect()
var r = Seq((0,0)).toDF("Vertex","Length")
var q = Seq((1,0)).toDF("Vertex","Length")
var value = 2


// This function loops over each node and checks the distance it has to the node 20.
// Since there might be nodes in clusters that are separated there will not exist a path between this two nodes and therefore the distance will be 0
for (i <- a) {
  // Try clause to see if there is a connection between the nodes
    try
    {
      value = shortestPath(intConverter(i).getOrElse(0)).get
      q = Seq((i,value)).toDF("Vertex","Length")
      r = r.union(q)
    }
  // Catch clause if the above fails since it gets an empty value it will add the node with a 0
    catch
    {
      case x @ (_ : RuntimeException | _ : NoSuchElementException) =>
            {
              q = Seq((intConverter(i).getOrElse(0),0)).toDF("Vertex","Length")
              r = r.union(q)
            }
    }
}
```

Cancel Running command...

(5) Spark Jobs

- Job 30005 View (1 stages)
- Job 30006 View (1 stages)
- Job 30007 View (18 stages)
- Job 30008 View (40 stages)
- Job 30009 View (40 stages)

Cmd 20

```scala
display(r.orderBy("Length", ascending = False))
```

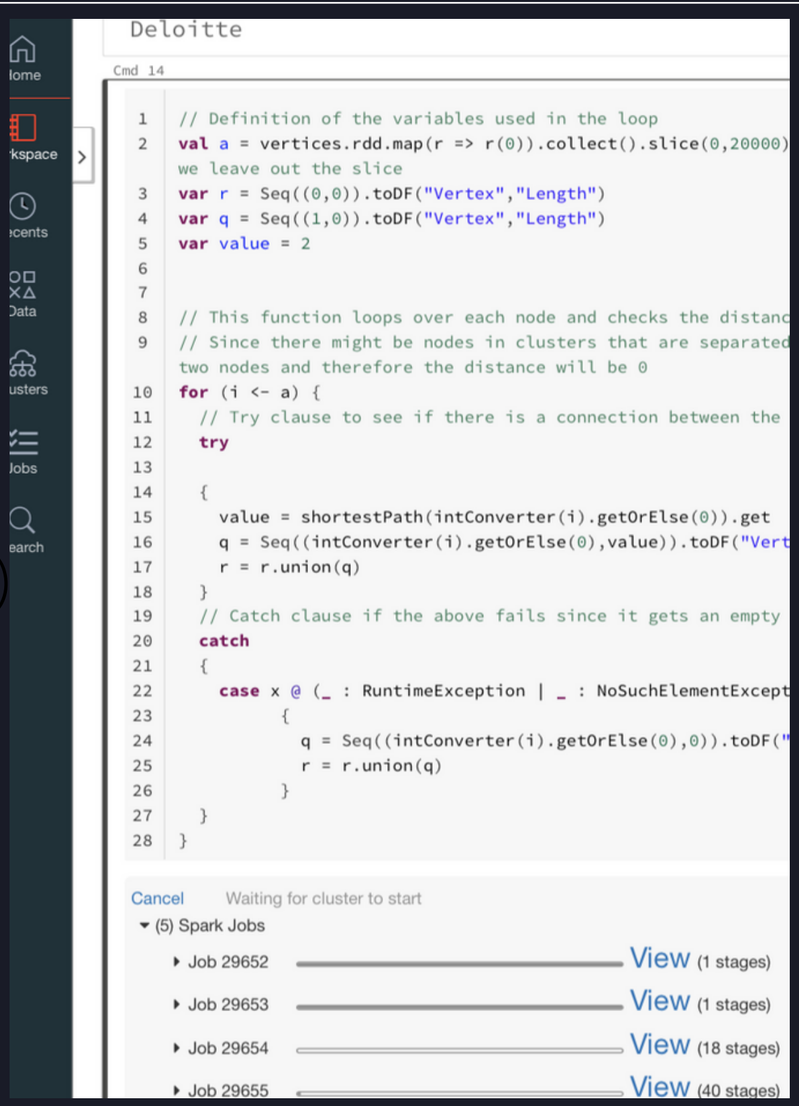
Cancel Waiting to run...

Cmd 21

6h+ running
with breakdown
of server

LONGEST PATH WITH 20K

7h+ running
with no end
(observing progress on phone)



CIRCLES IN THE GRAPH

```
def dfs(graph, start, end):
    fringe = [(start, [])]
    while fringe:
        state, path = fringe.pop()
        if path and state == end:
            yield path
            continue
        for next_state in graph[state]:
            if next_state in path:
                continue
            fringe.append((next_state, path+[next_state]))

cycles = [[node]+path  for node in graph for path in dfs(graph, node, node)]
```

The Idea of finding circles in the graph

READING THE DATA DIRECTLY

Instead of downloading the data locally, this process could be used
in a notebook that is in a cloud
(as bash command this would be piped with |)

- !wget http://an.kaist.ac.kr/~haewoon/release/twitter_social_graph/twitter_rv.zip
- !unzip temp.zip
- !rm temp.zip