

# Twitter Social Graph Analysis



5G E

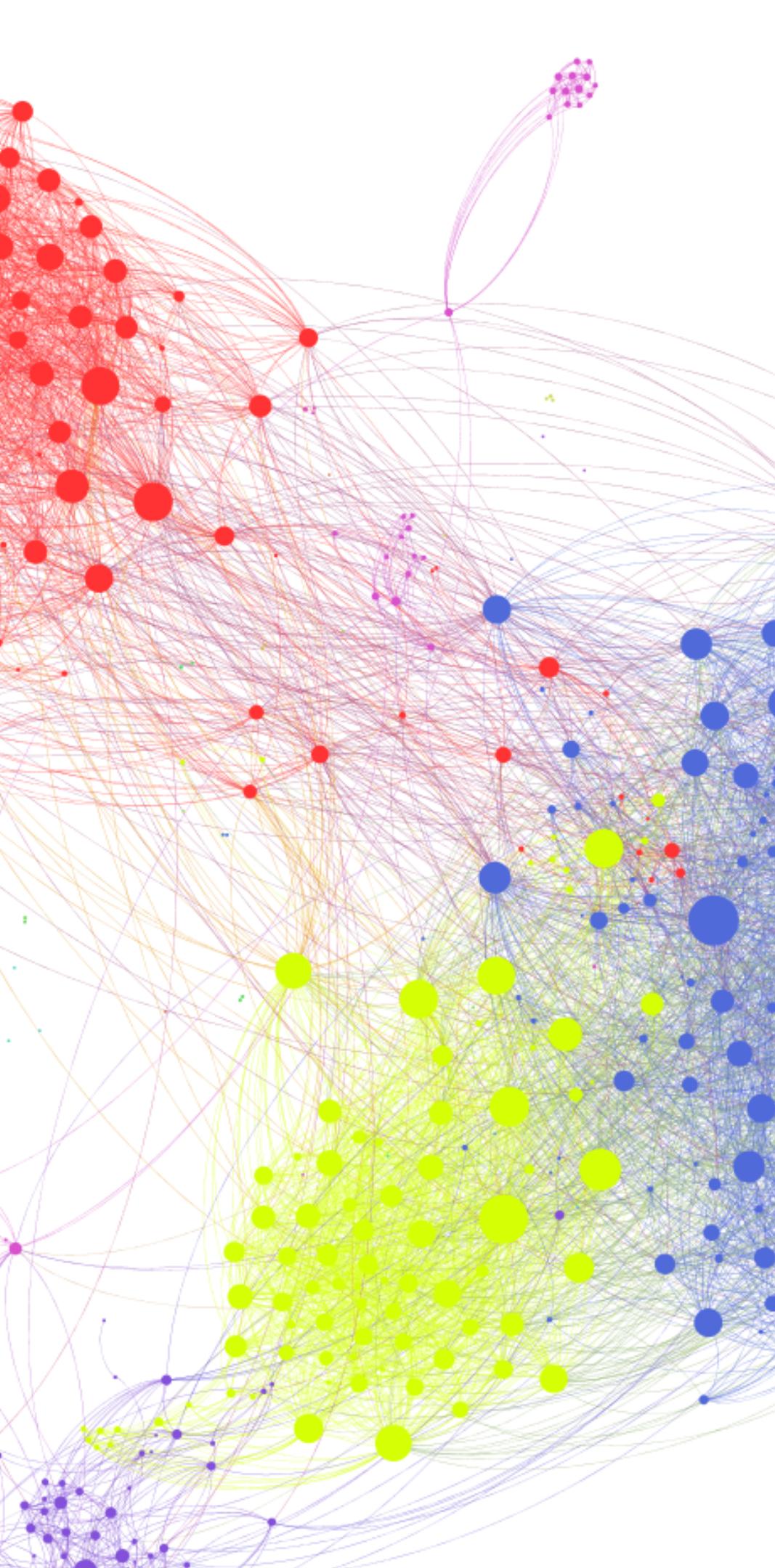


# Agenda

- 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



Top 3 accounts  
based on followers



The  
importance  
is defined by  
the amount  
of followers!

ID

Followers

10316422 — 2583

14206126 — 940

12750862 — 914

These are Twitters hot shots!

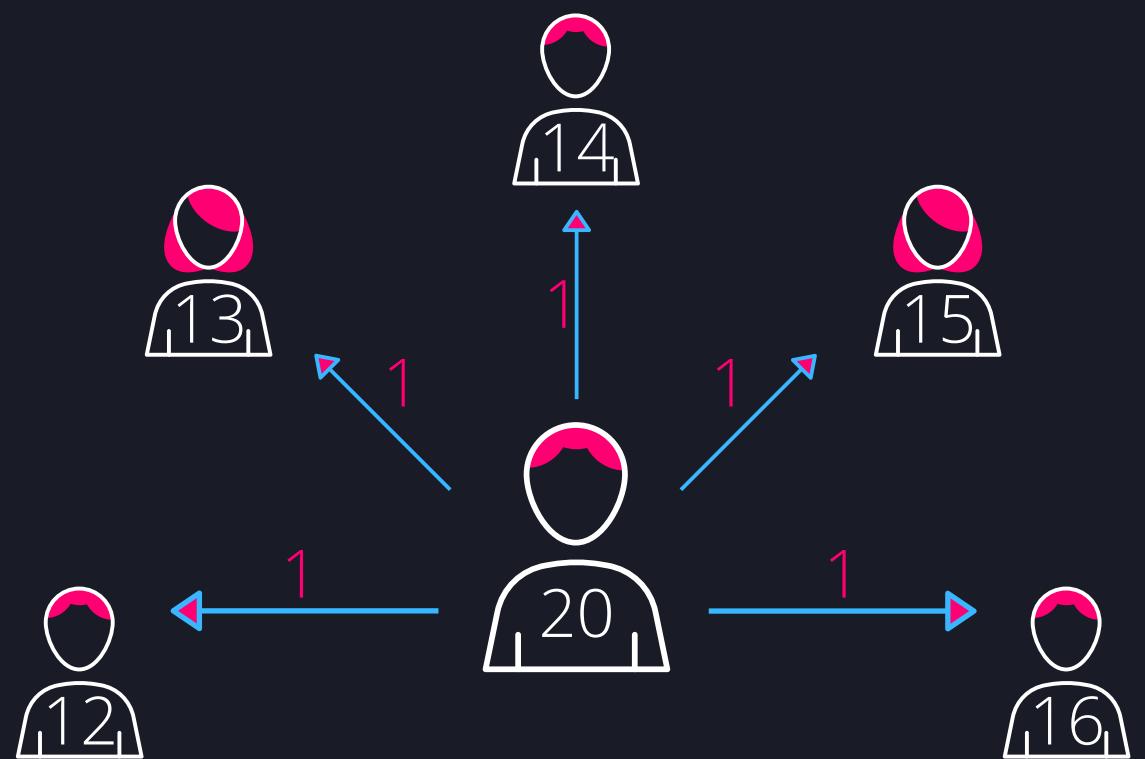
User following  
most accounts



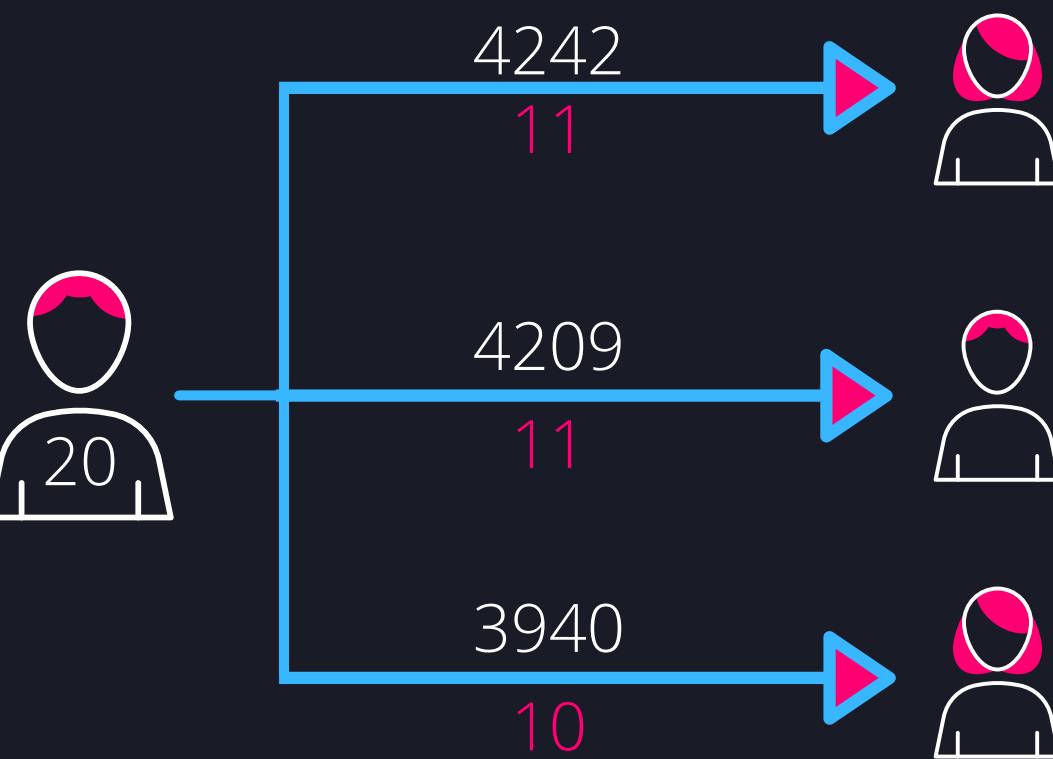
User 20  
follows  
**1'213'787**  
accounts



## Shortest path

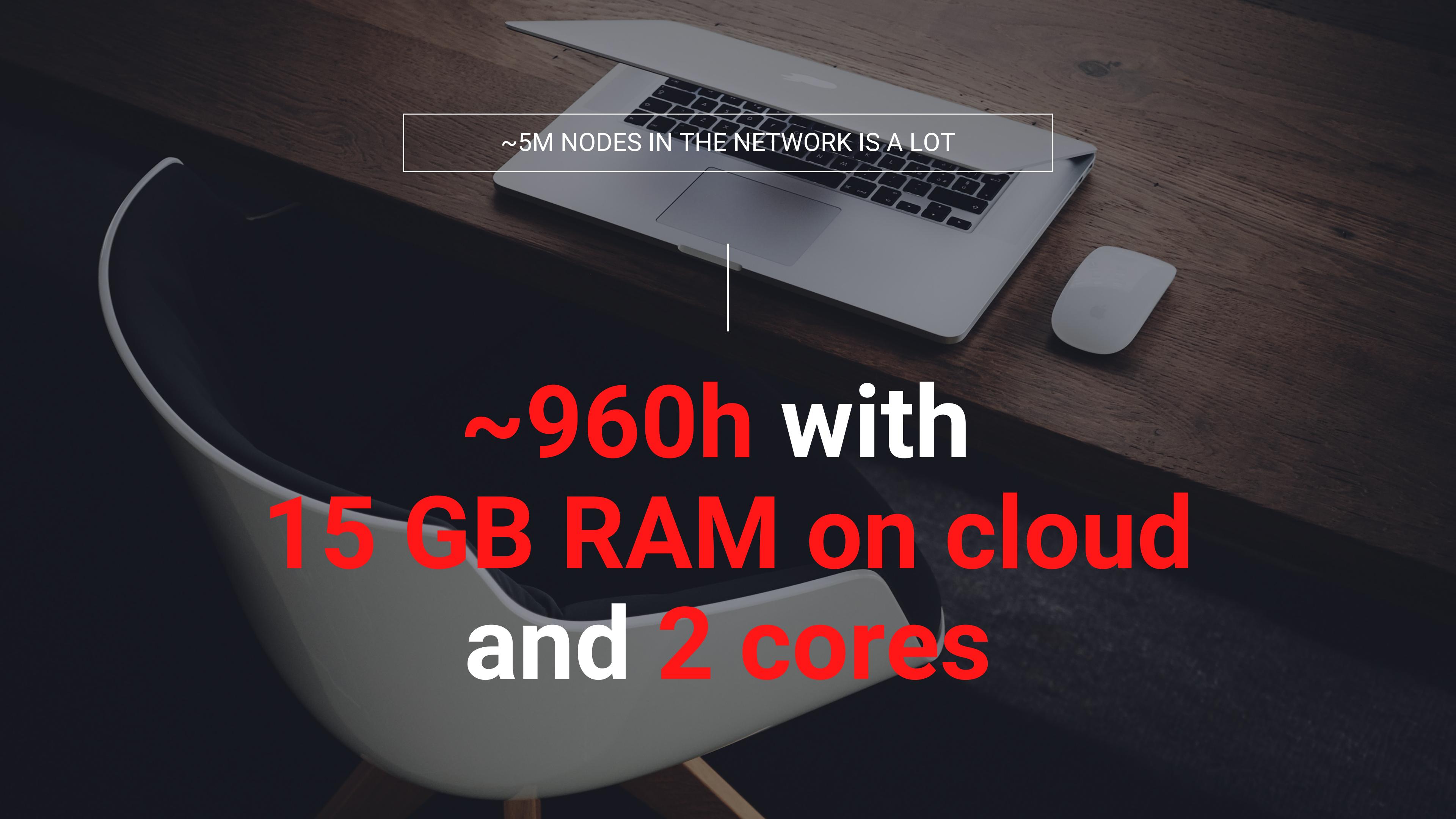


## Longest path



There is no one closer  
than your neighbors

These results are only  
representative for the first 4k nodes

A dark, moody photograph of a laptop and a mouse on a wooden desk. The laptop is open, showing its keyboard and trackpad. A white mouse sits to its right. The background is dark, with dramatic lighting highlighting the textures of the wood and the metallic surfaces of the laptop.

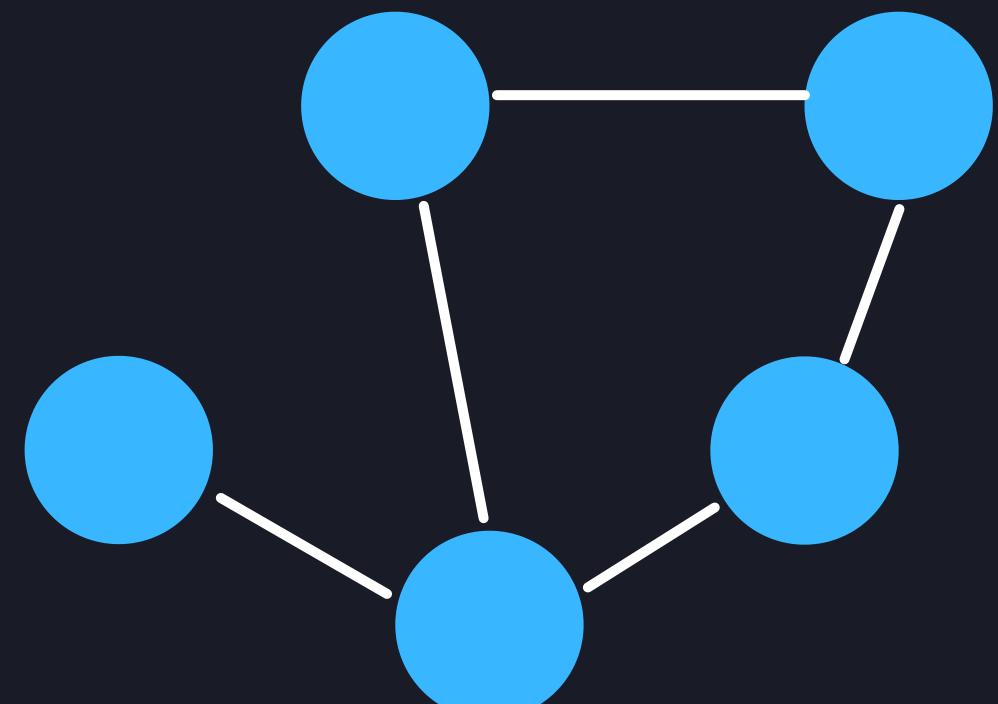
~5M NODES IN THE NETWORK IS A LOT

~960h with  
15 GB RAM on cloud  
and 2 cores

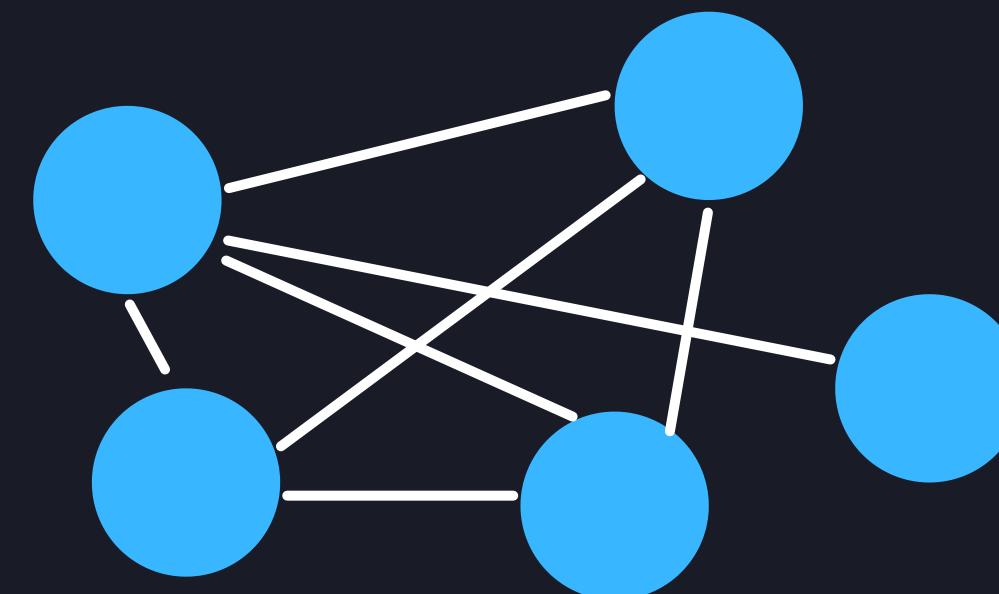
# Additional topics related to graphs



## 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 individuals 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

Clusters / Deloitte

**Deloitte** | [Cancel](#) [Confirm](#)

**0 Workers:** 0.0 GB Memory, 0 Cores, 0 DBU  
**1 Driver:** 15.3 GB Memory, 2 Cores, 1 DBU [?](#)

**Cluster Name**

Deloitte

**Databricks Runtime Version** [?](#)

Runtime: 8.2 (Scala 2.12, Spark 3.1.1) | [▼](#)

**Note** Databricks Runtime 8.x uses Delta Lake as the default table format. [Learn more](#)

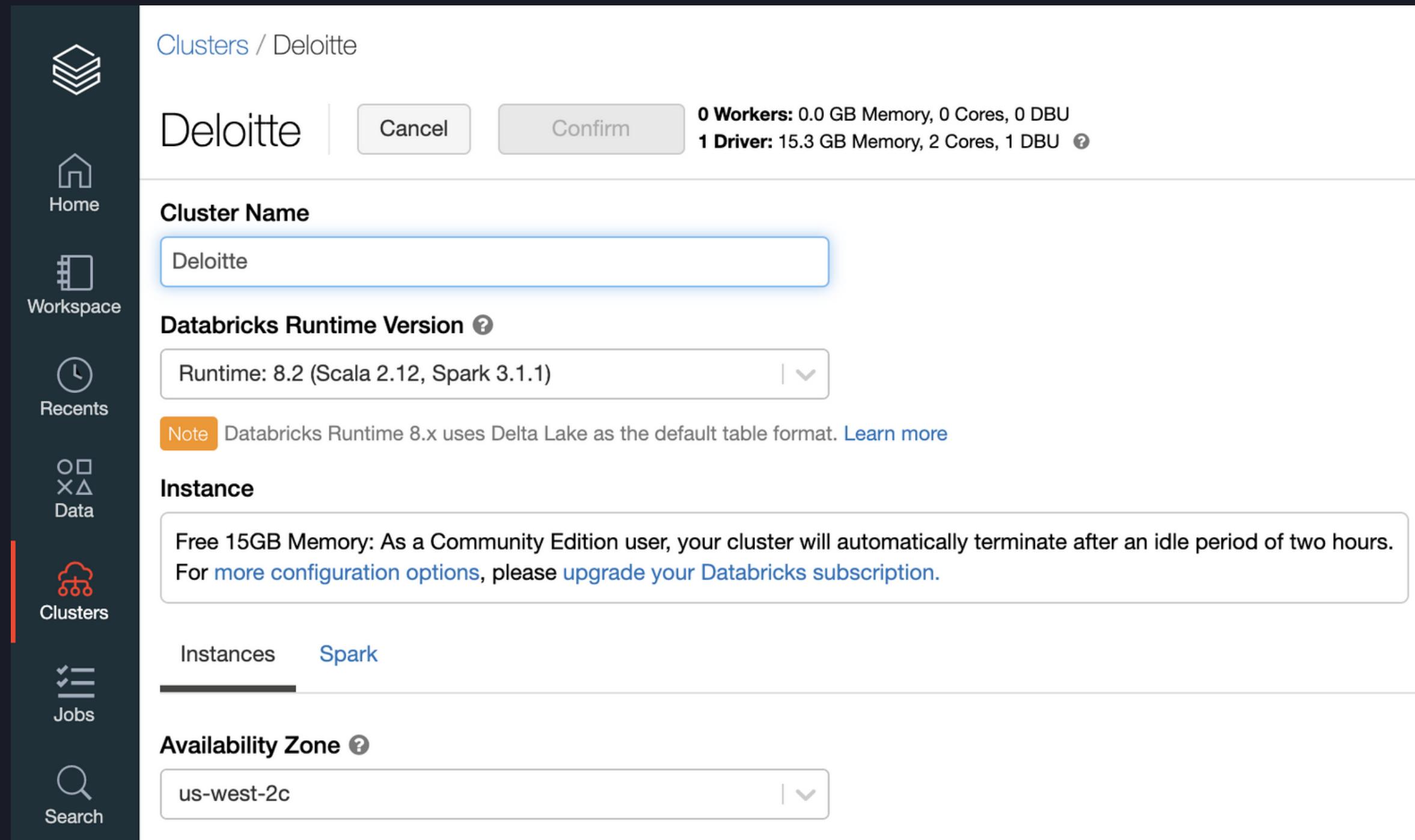
**Instance**

Free 15GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours.  
For [more configuration options](#), please [upgrade your Databricks subscription](#).

Instances [Spark](#)

**Availability Zone** [?](#)

us-west-2c | [▼](#)



## LONGEST PATH WITH ALL

```
1 // Definition of the variables used in the loop
2 val a = vertices.rdd.map(r => r(0)).collect()
3 var r = Seq((0,0)).toDF("Vertex","Length")
4 var q = Seq((1,0)).toDF("Vertex","Length")
5 var value = 2
6
7
8 // This function loops over each node and checks the distance it has to the node 20.
9 // 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
10 for (i <- a) {
11   // Try clause to see if there is a connection between the nodes
12   try
13   {
14     value = shortestPath(intConverter(i).getOrElse(0)).get
15     q = Seq((i,value)).toDF("Vertex","Length")
16     r = r.union(q)
17   }
18   // Catch clause if the above fails since it gets an empty value it will add the node with a 0
19   catch
20   {
21     case x @ (_ : RuntimeException | _ : NoSuchElementException) =>
22     {
23       q = Seq((intConverter(i).getOrElse(0),0)).toDF("Vertex","Length")
24       r = r.union(q)
25     }
26   }
27 }
```

6h+ running  
with breakdown  
of server

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

Cancel \*\*\* Waiting to run...

Cmd 21

## LONGEST PATH WITH 20K

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

The screenshot shows the Deloitte Data Cloud interface. On the left is a sidebar with icons for Home, kspace, Metrics, Data, Clusters, Jobs, and Search. The main area has a title "Deloitte" and "Cmd 14". It contains the following Scala code:

```
1 // Definition of the variables used in the loop
2 val a = vertices.rdd.map(r => r(0)).collect().slice(0,20000)
we leave out the slice
3 var r = Seq((0,0)).toDF("Vertex","Length")
4 var q = Seq((1,0)).toDF("Vertex","Length")
5 var value = 2
6
7
8 // This function loops over each node and checks the distance
9 // Since there might be nodes in clusters that are separated
10 // by two nodes and therefore the distance will be 0
11 for (i <- a) {
12     // Try clause to see if there is a connection between the
13     // two nodes
14     try {
15         value = shortestPath(intConverter(i).getOrElse(0)).get
16         q = Seq((intConverter(i).getOrElse(0),value)).toDF("Vertex",
17                 "Length")
18         r = r.union(q)
19     }
20     // Catch clause if the above fails since it gets an empty
21     catch {
22         case x @ (_ : RuntimeException | _ : NoSuchElementException)
23         {
24             q = Seq((intConverter(i).getOrElse(0),0)).toDF("Vertex",
25                     "Length")
26             r = r.union(q)
27         }
28     }
}
```

At the bottom, there is a "Waiting for cluster to start" message and a list of "Spark Jobs":

- Job 29652 View (1 stages)
- Job 29653 View (1 stages)
- Job 29654 View (18 stages)
- Job 29655 View (40 stages)

## 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