



# Apache Spark Using Structured Streaming API

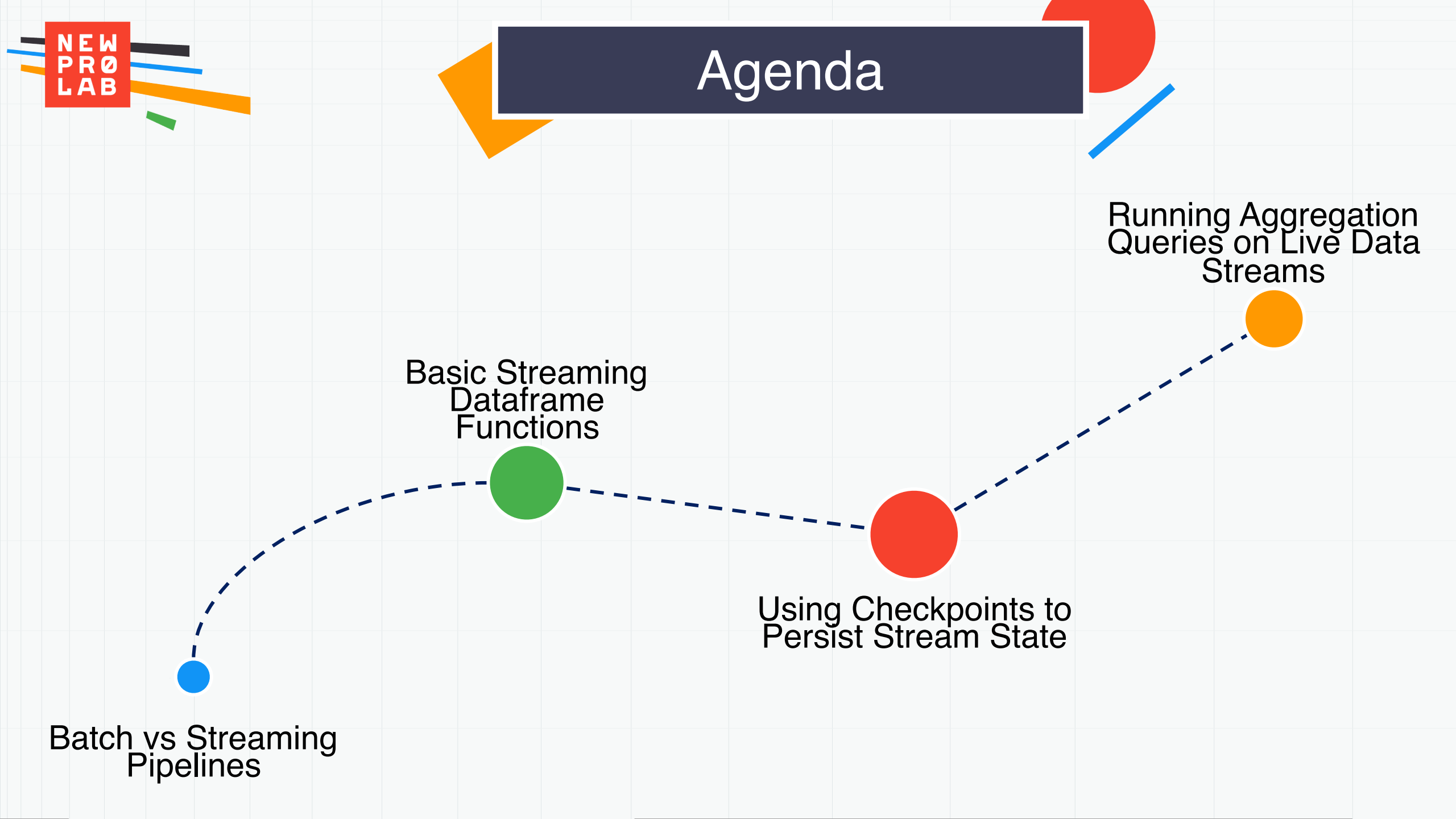
# Agenda

Batch vs Streaming  
Pipelines

Basic Streaming  
Dataframe  
Functions

Using Checkpoints to  
Persist Stream State


Running Aggregation  
Queries on Live Data  
Streams



# Batch vs Streaming Pipelines

# Batch vs Streaming

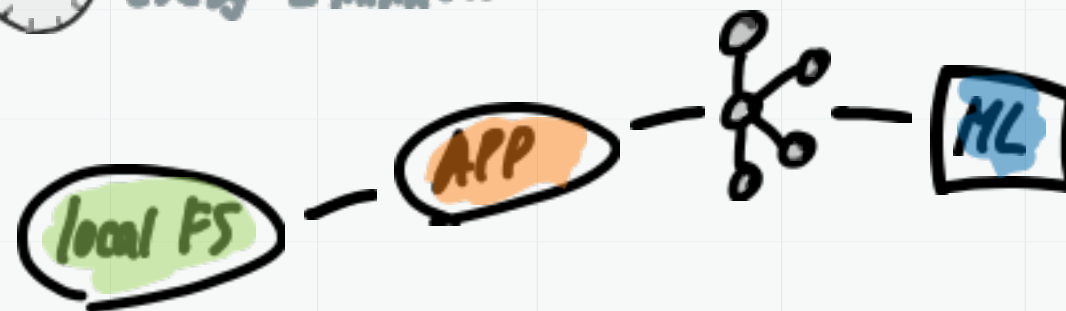
Batch

 every 4 hours



Streaming

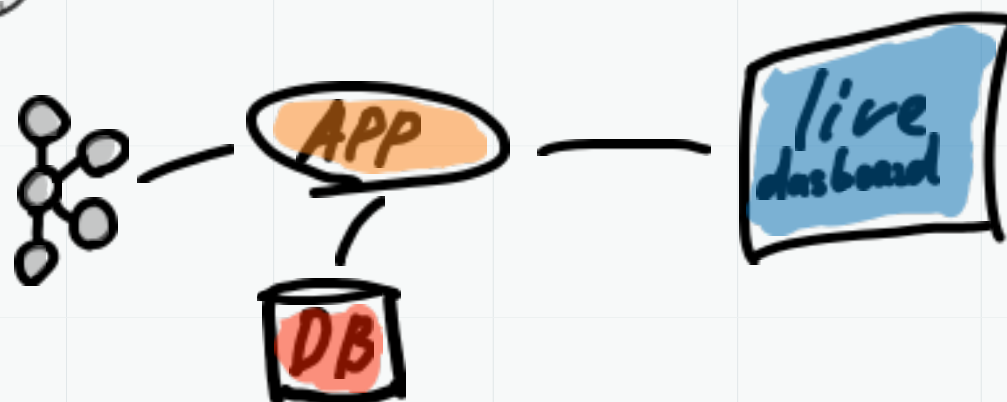
 every 2 minutes



 once a week



 every 10 seconds



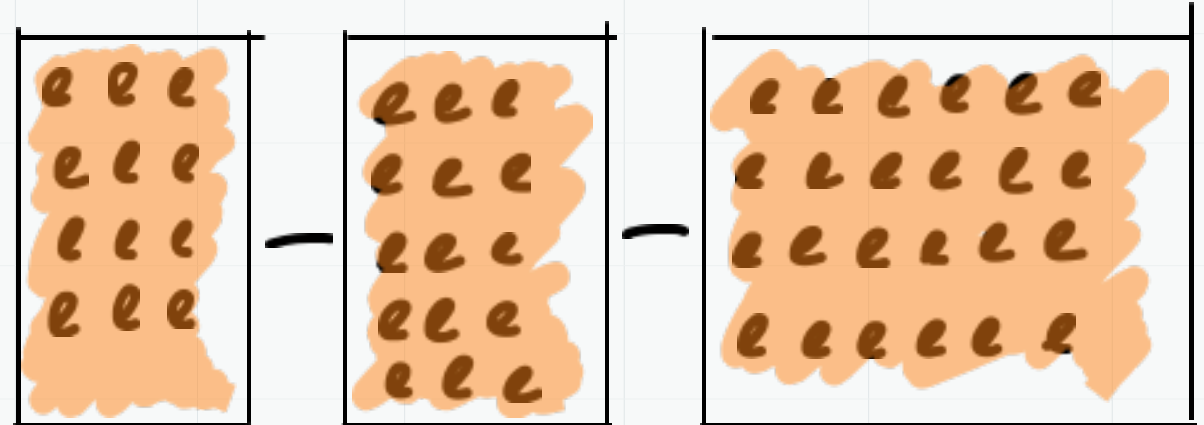
# Stream types

Real time



- Process one event at time
- No aggregations
- Low latency
- Low throughput

Micro batch



- Process event batch at time
- Aggregations are supported
- High latency
- High throughput

# Streaming Sources

Files



Local FS



amazon  
S3

# Streaming Sources

Files



Local FS



amazon  
S3

Databases



PostgreSQL



*cassandra*

ORACLE®

DATABASE

# Streaming Sources

Files



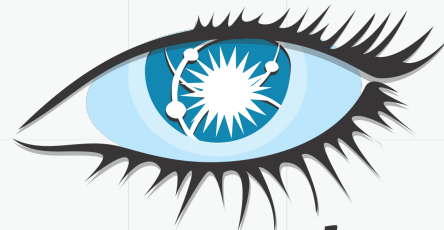
Local FS



Databases



PostgreSQL



cassandra



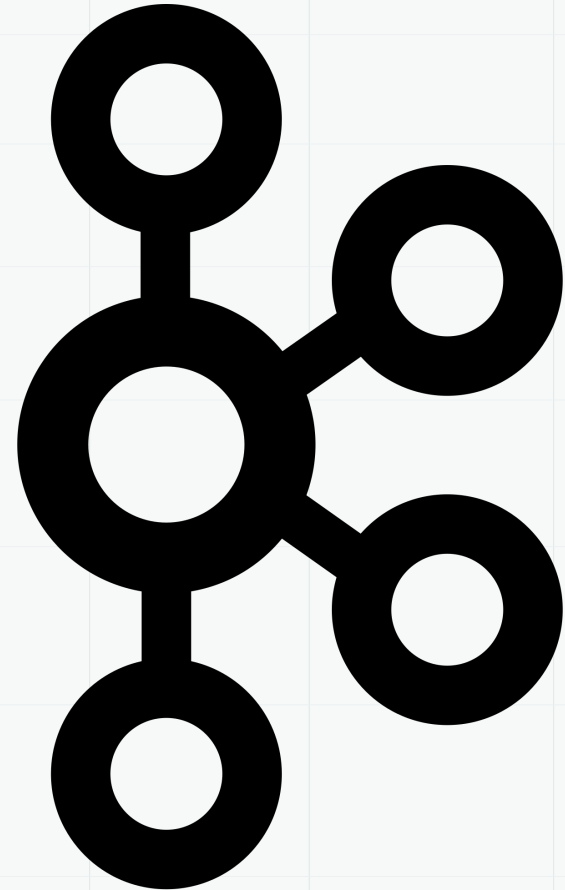
Streaming Systems



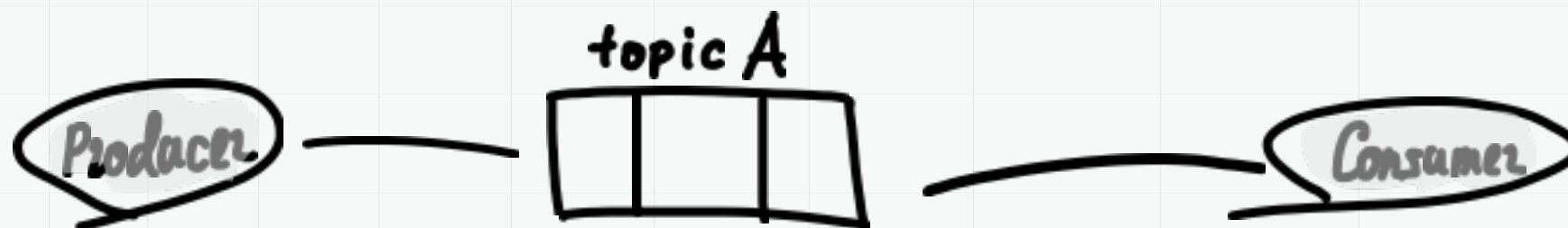


# Apache Kafka Overview

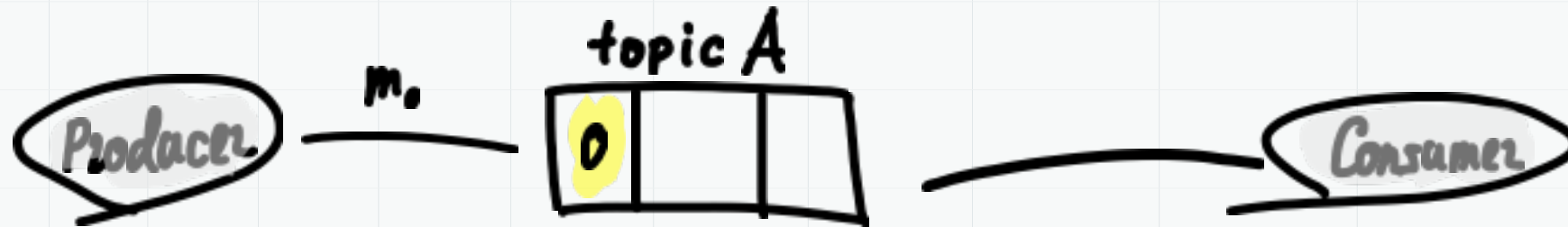
- Distributed streaming platform
- High throughput
- High availability
- Symmetric architecture



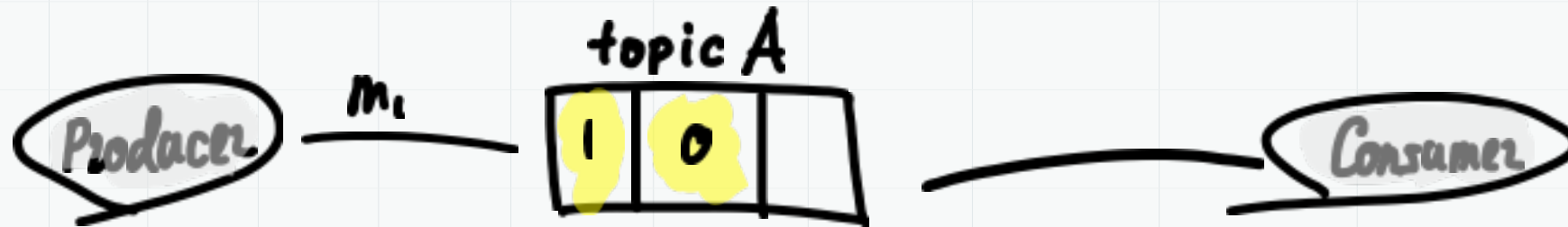
# Apache Kafka Overview



# Apache Kafka Overview



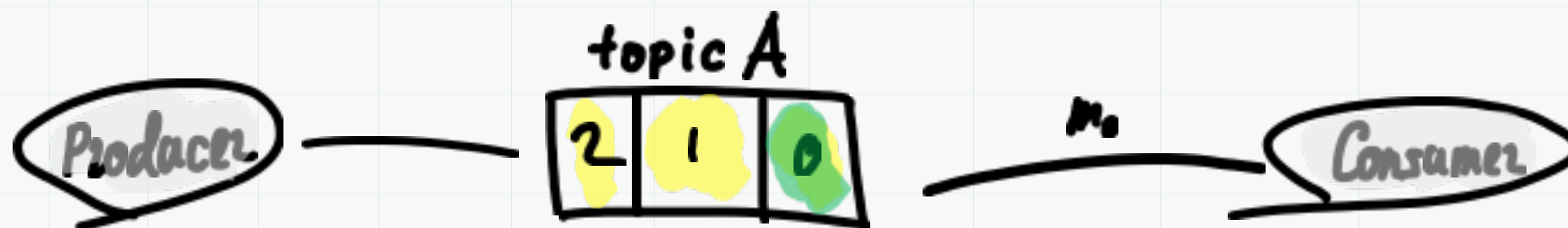
# Apache Kafka Overview



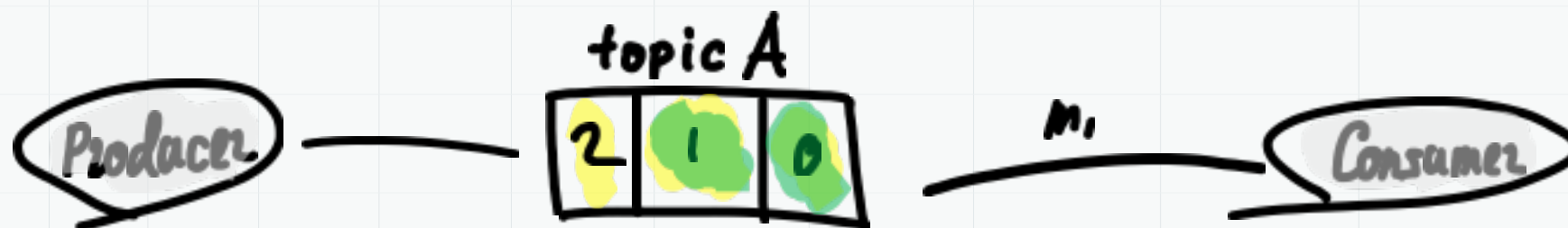
# Apache Kafka Overview



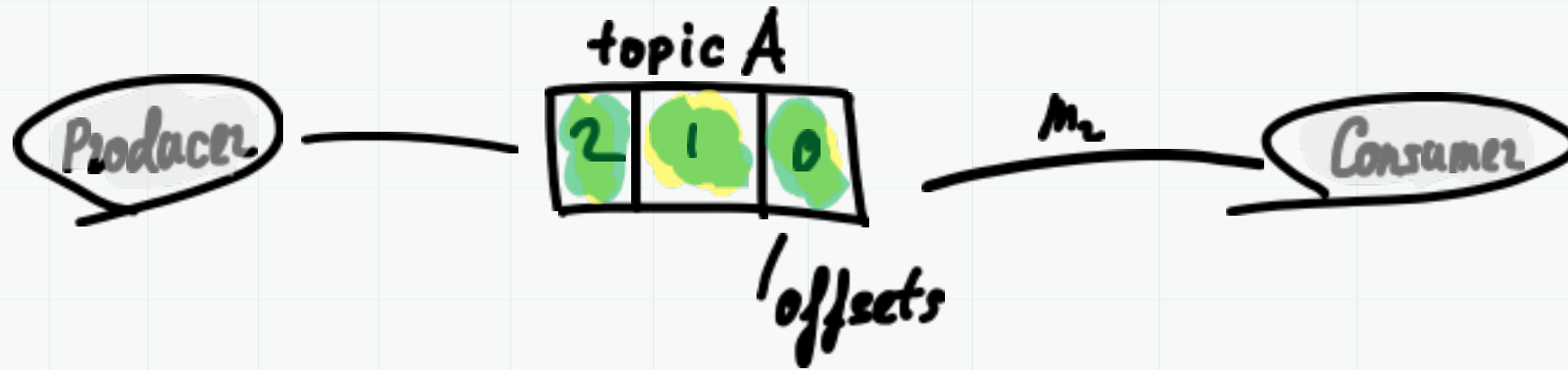
# Apache Kafka Overview



# Apache Kafka Overview

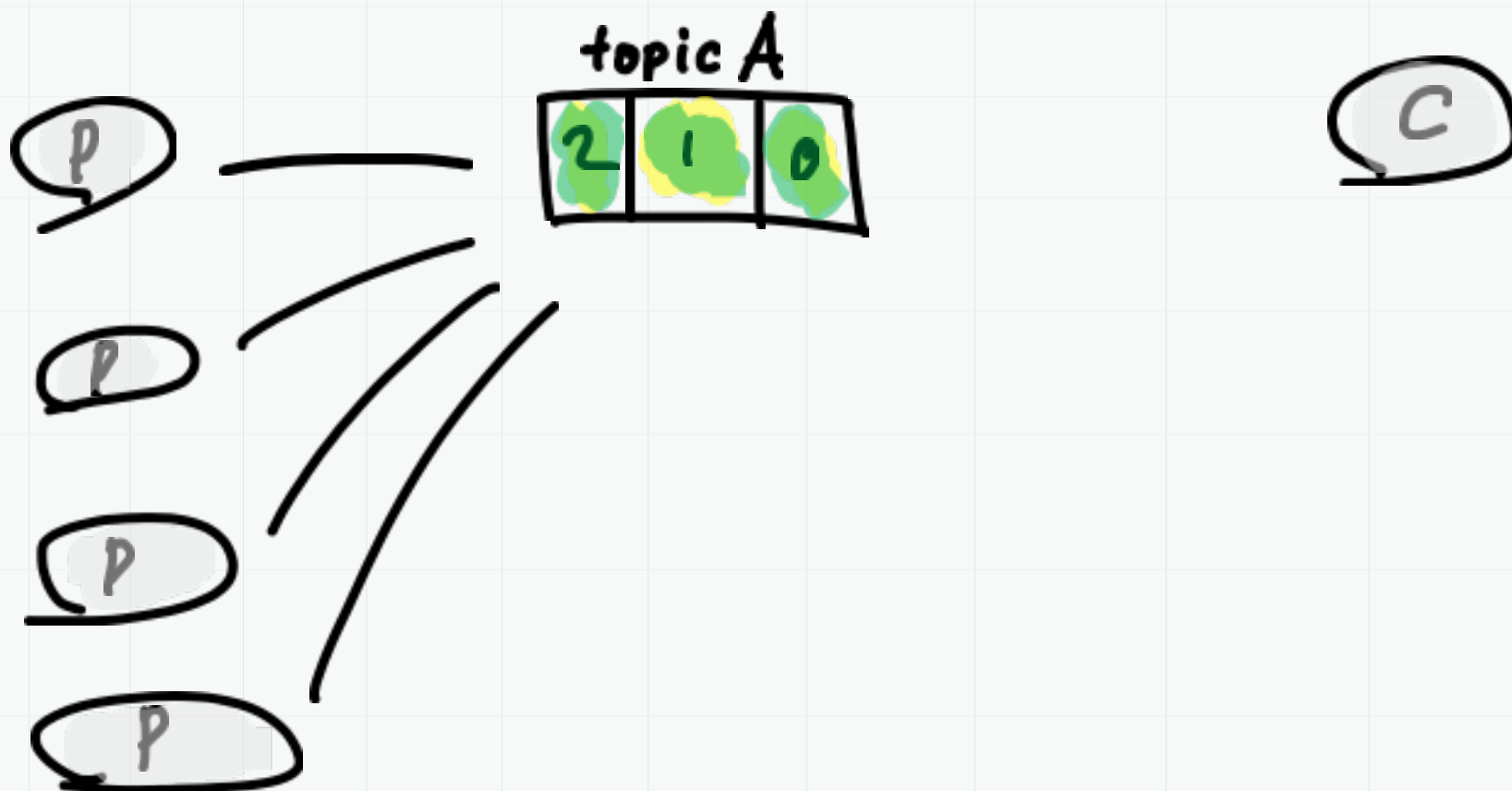


# Apache Kafka Overview

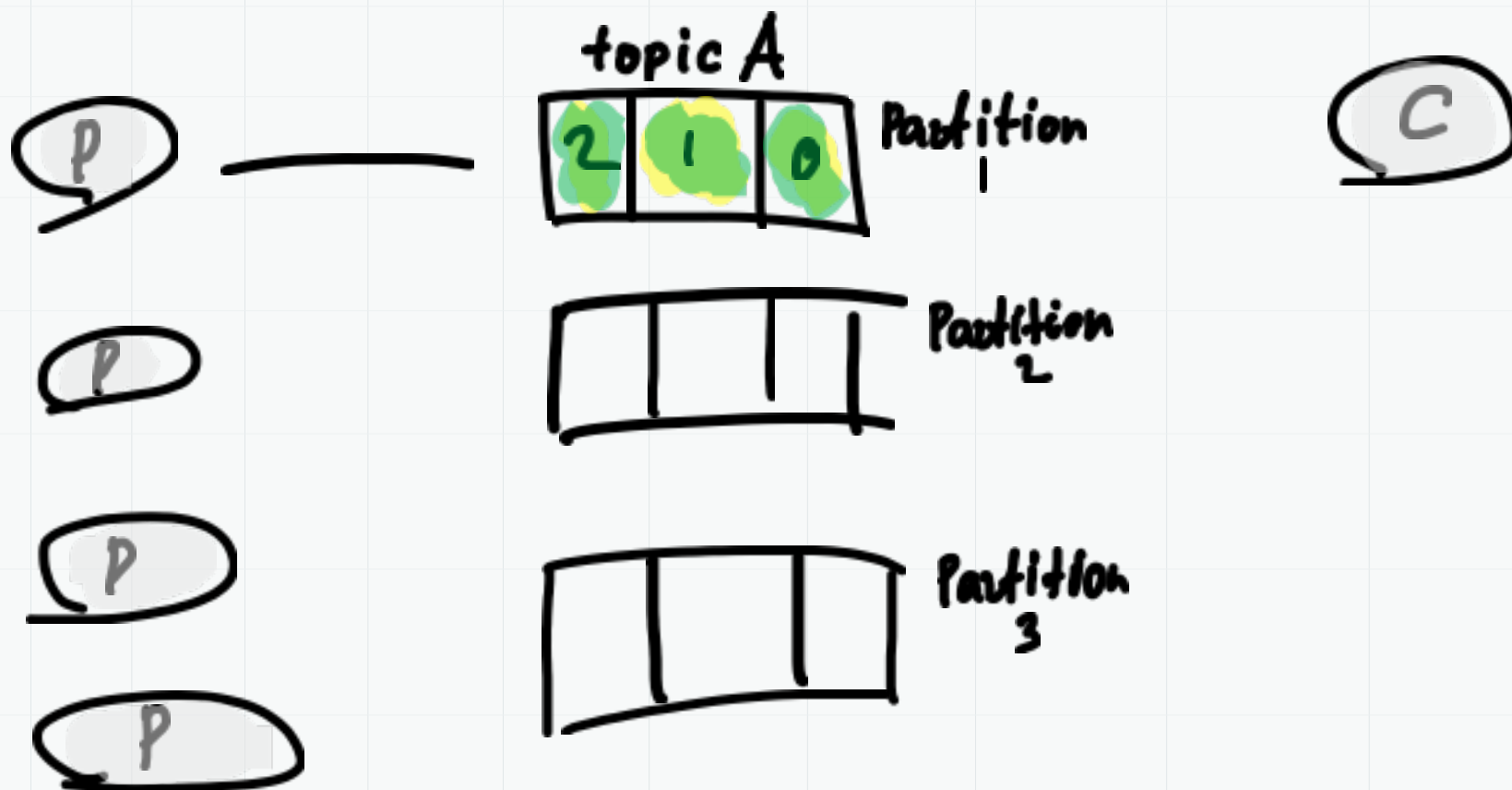




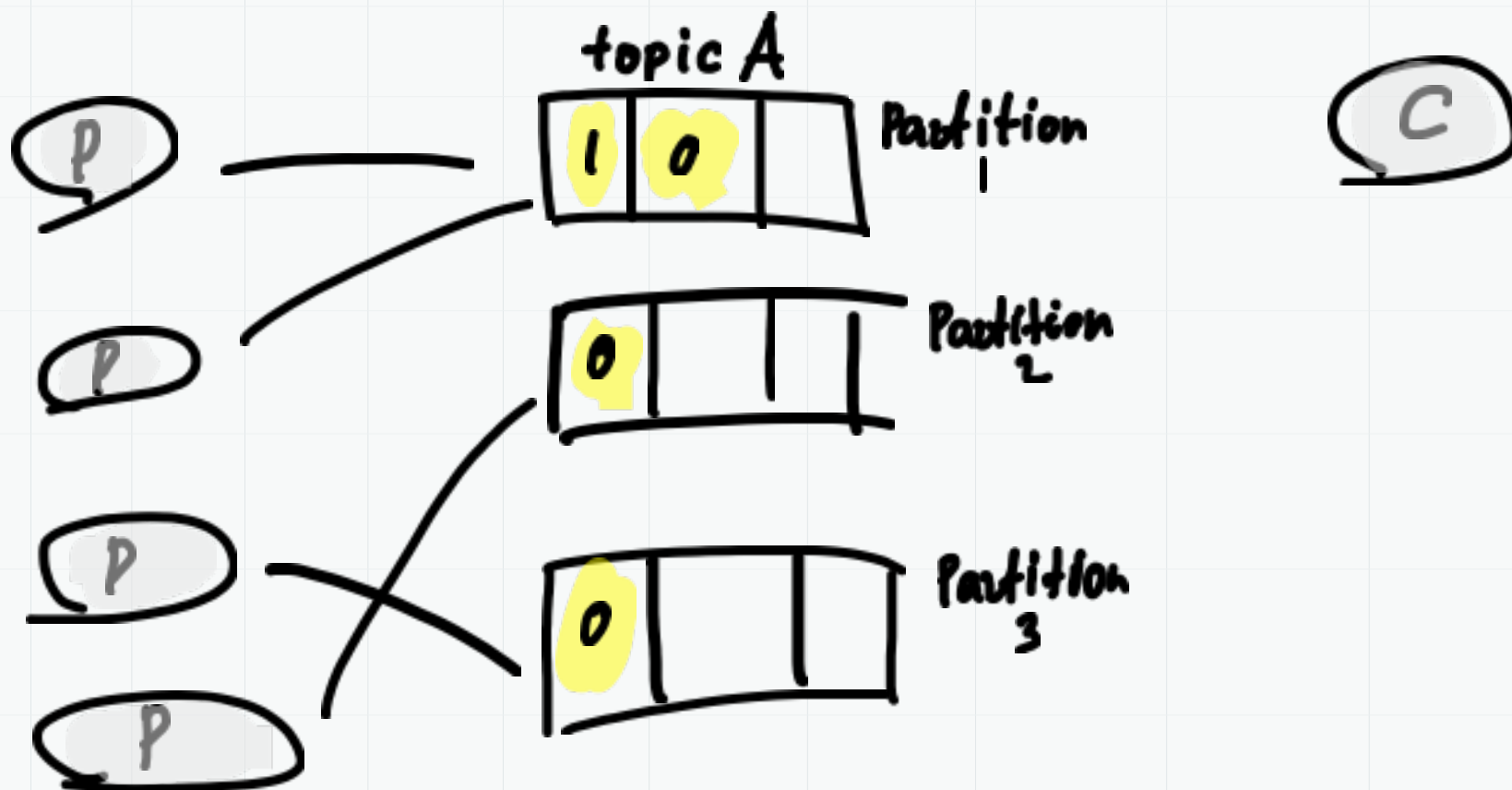
# Apache Kafka Overview



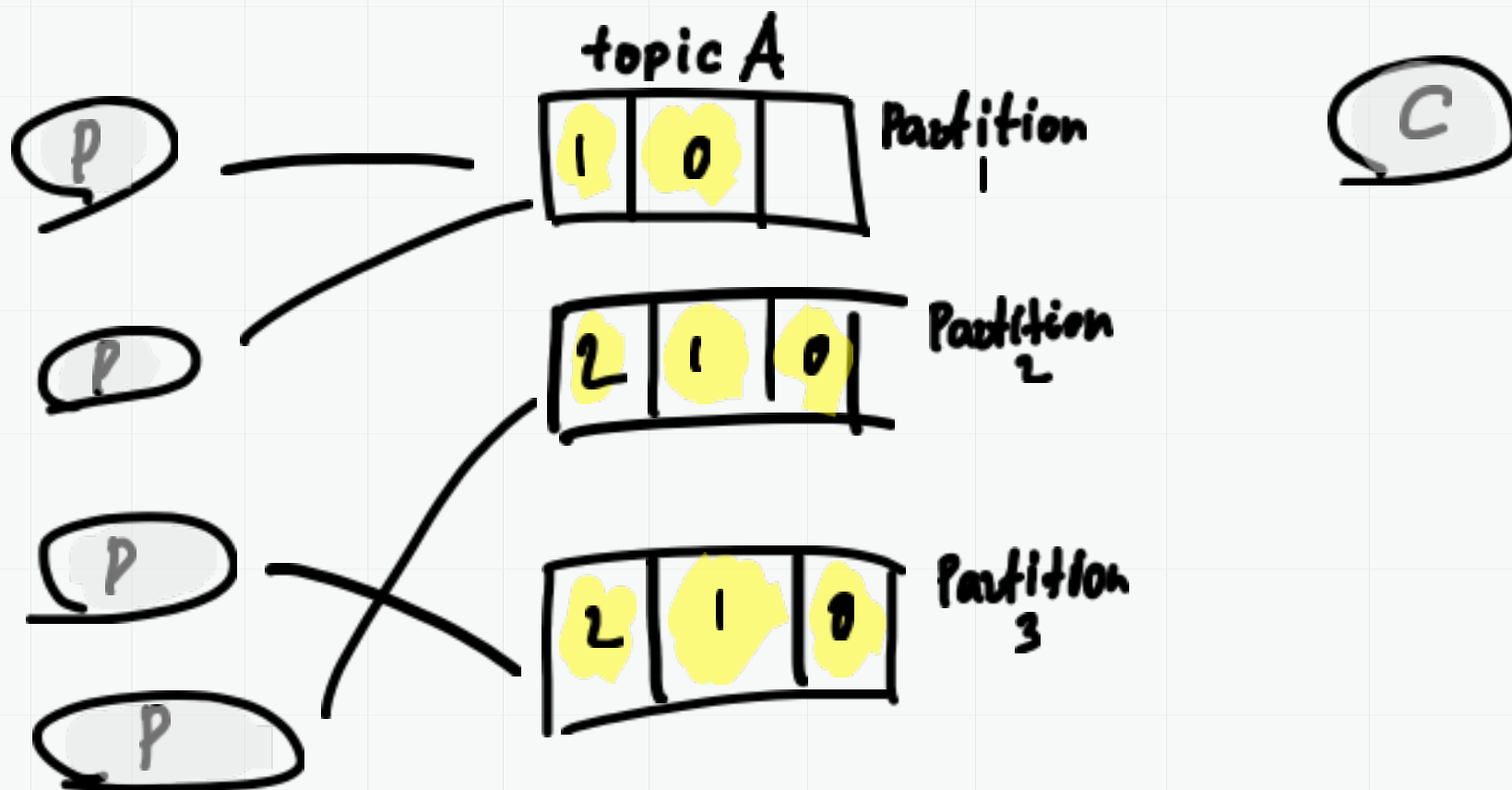
# Apache Kafka Overview



# Apache Kafka Overview

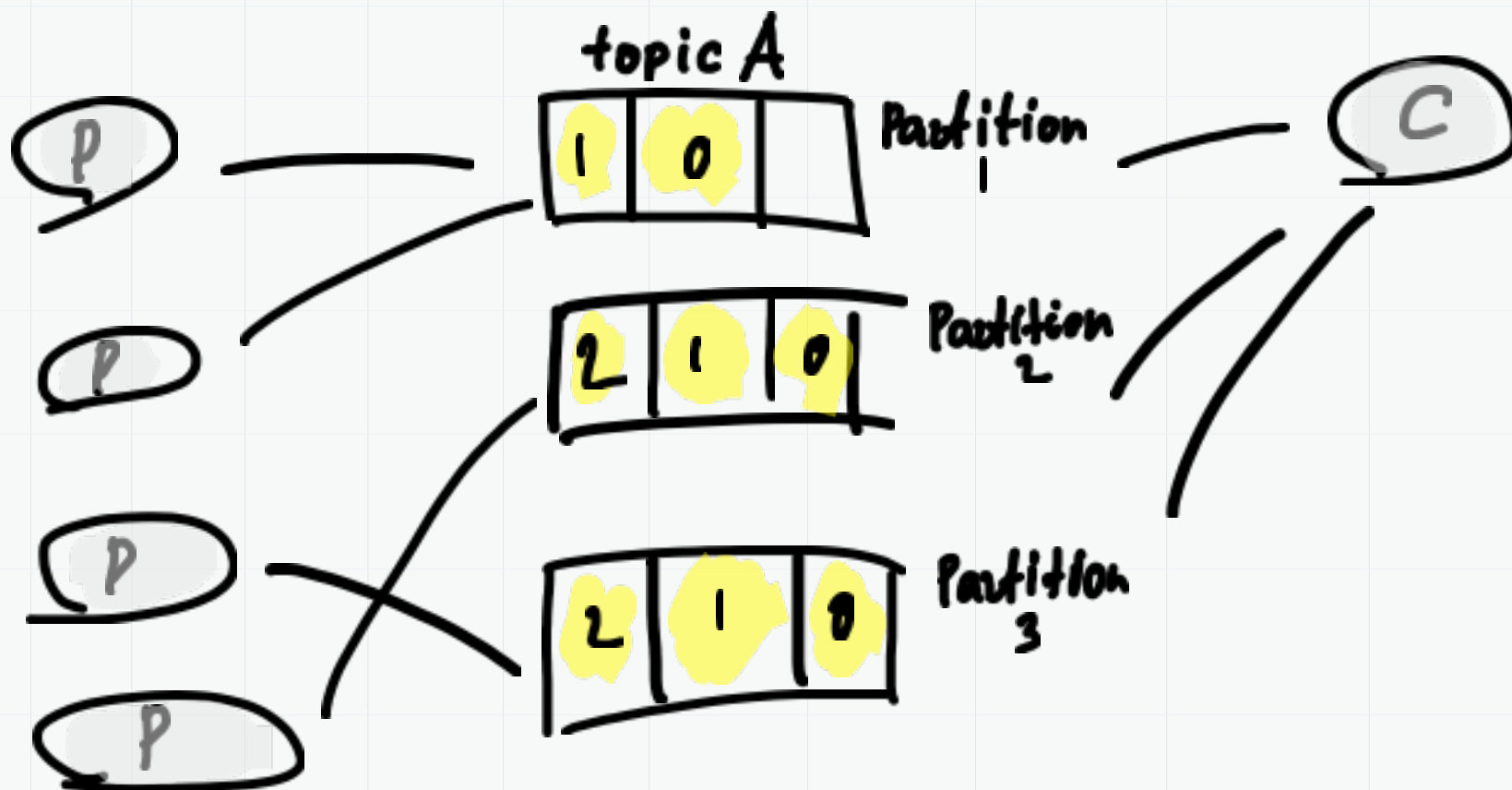


# Apache Kafka Overview



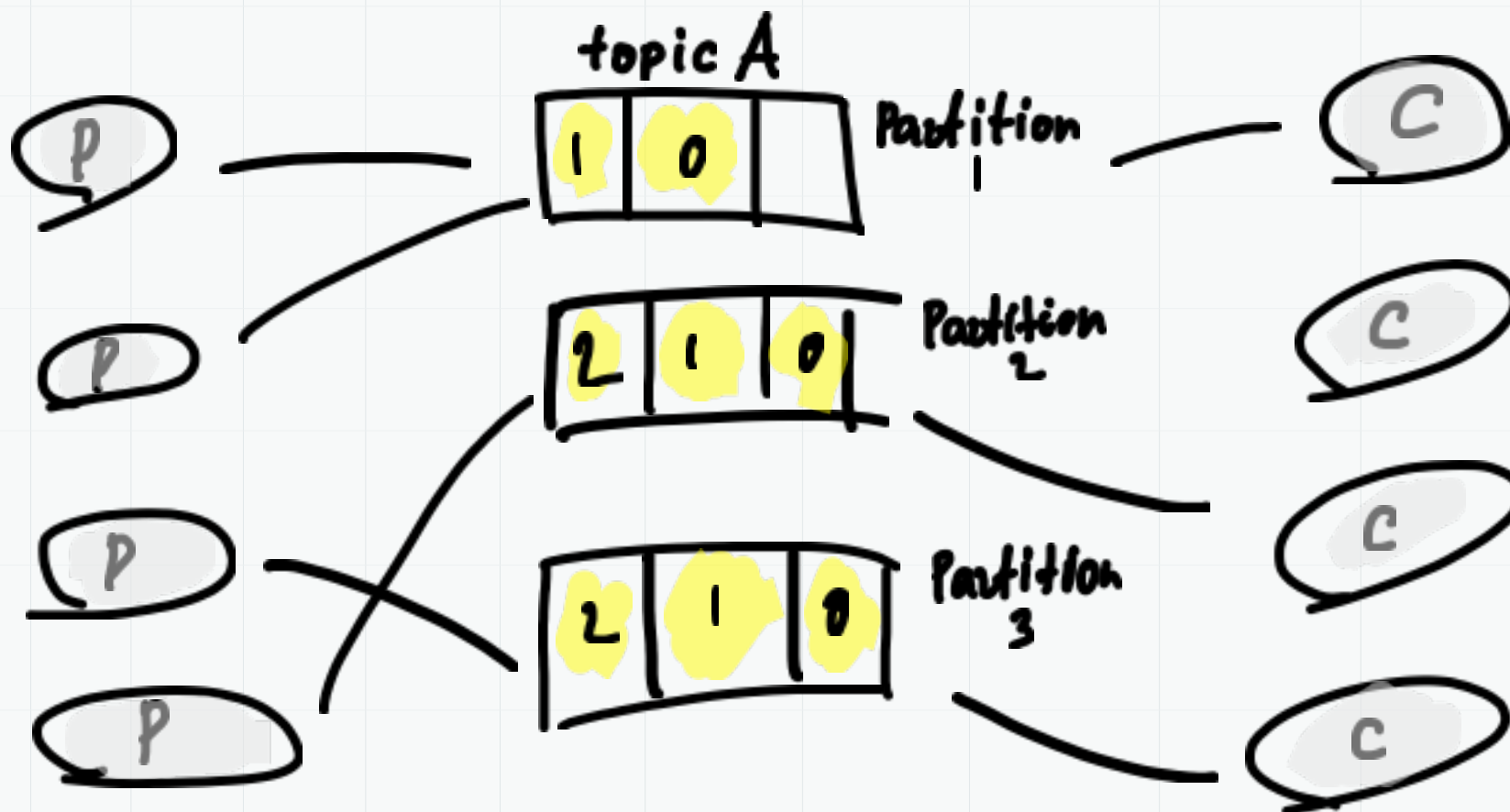
# Apache Kafka

## Overview

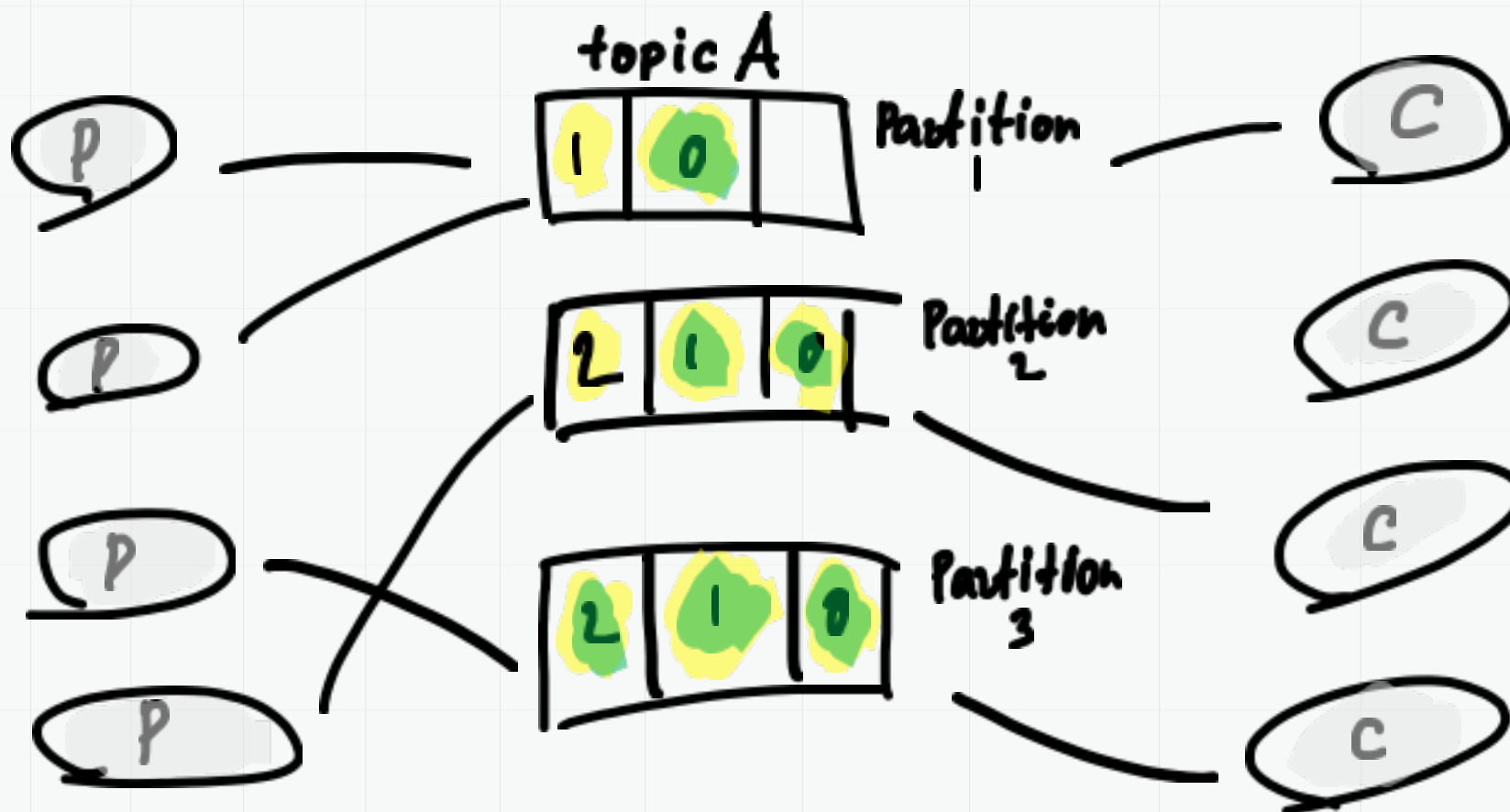


# Apache Kafka

## Overview

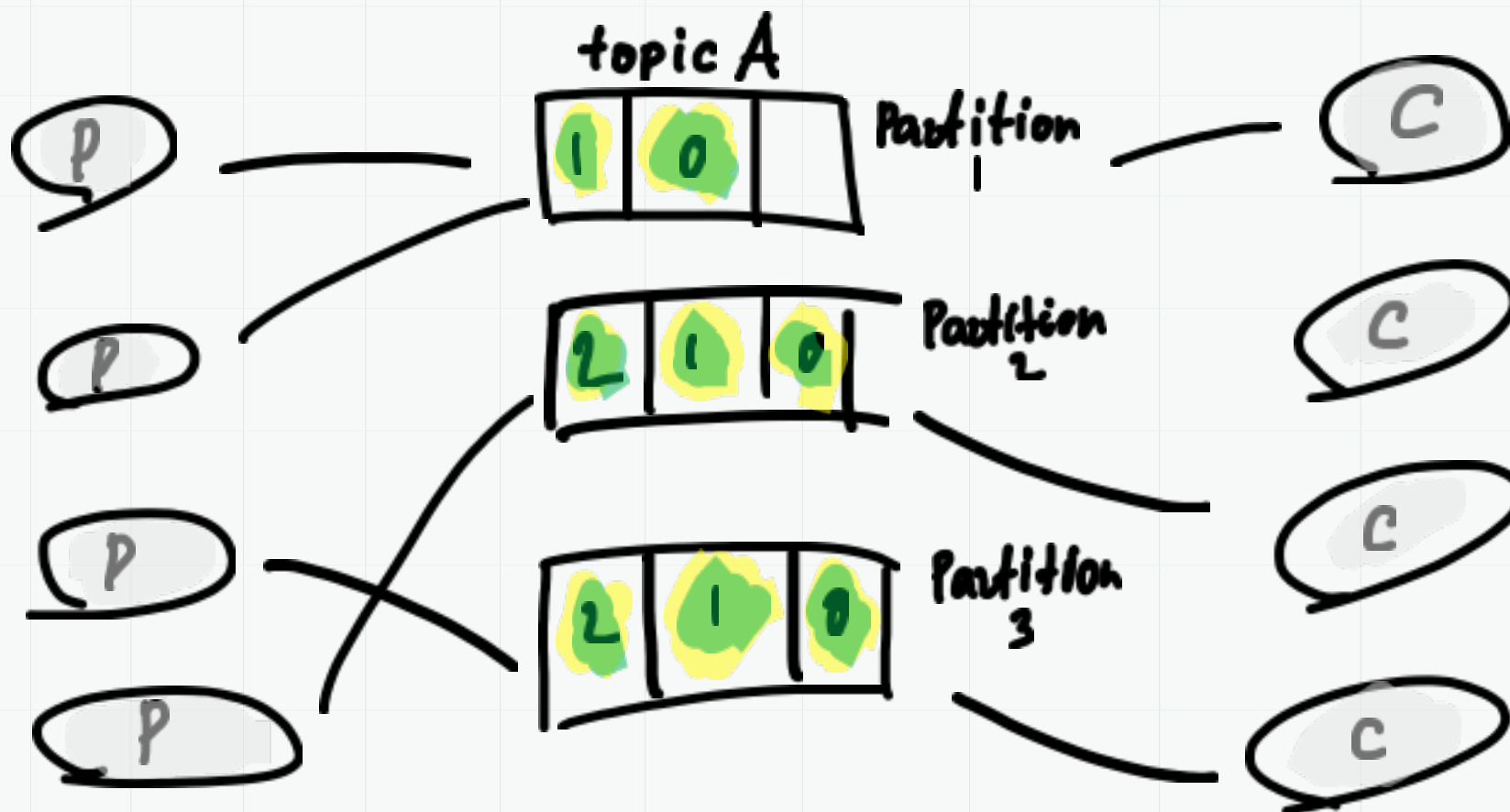


# Apache Kafka Overview



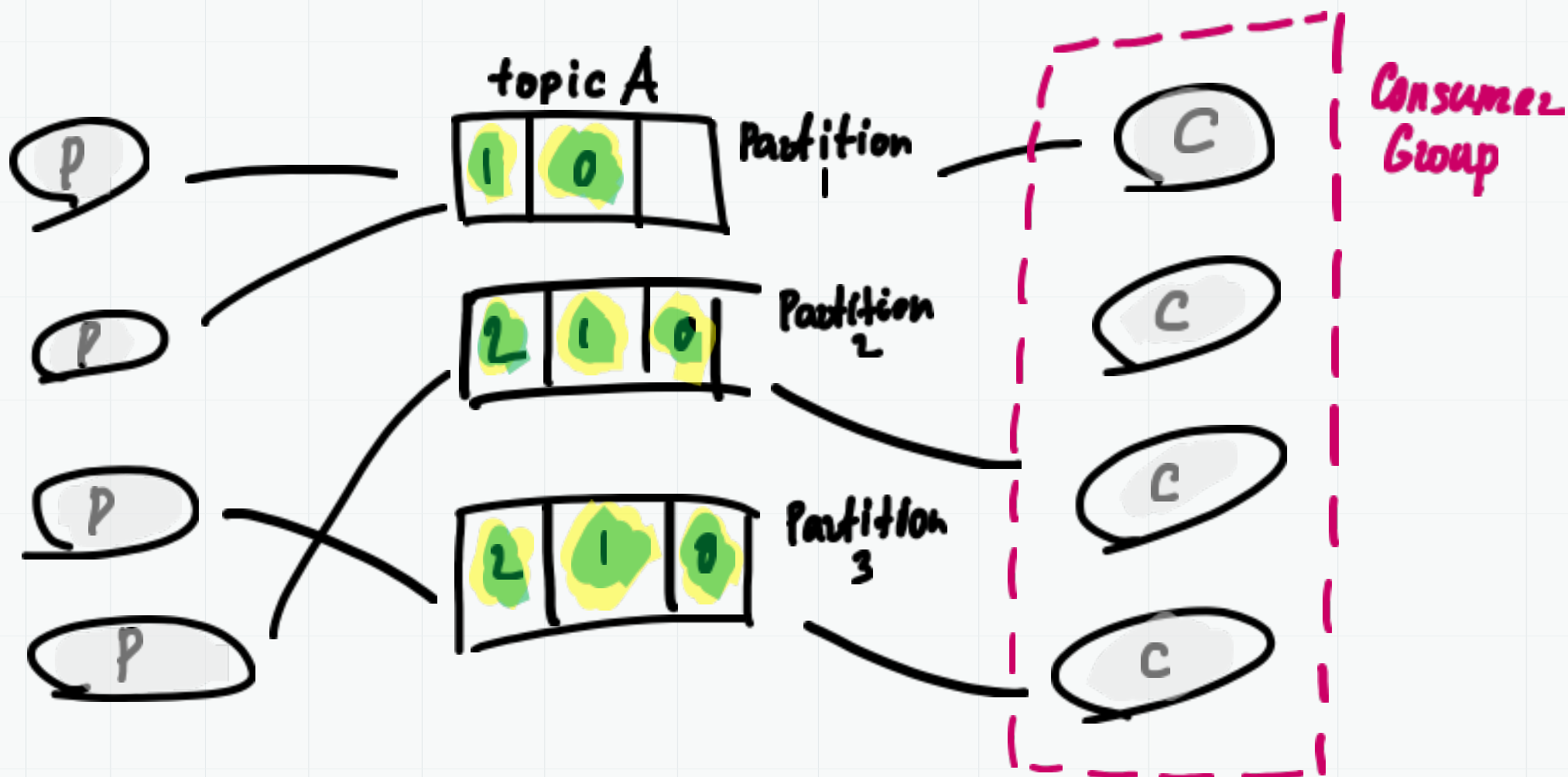
# Apache Kafka

## Overview



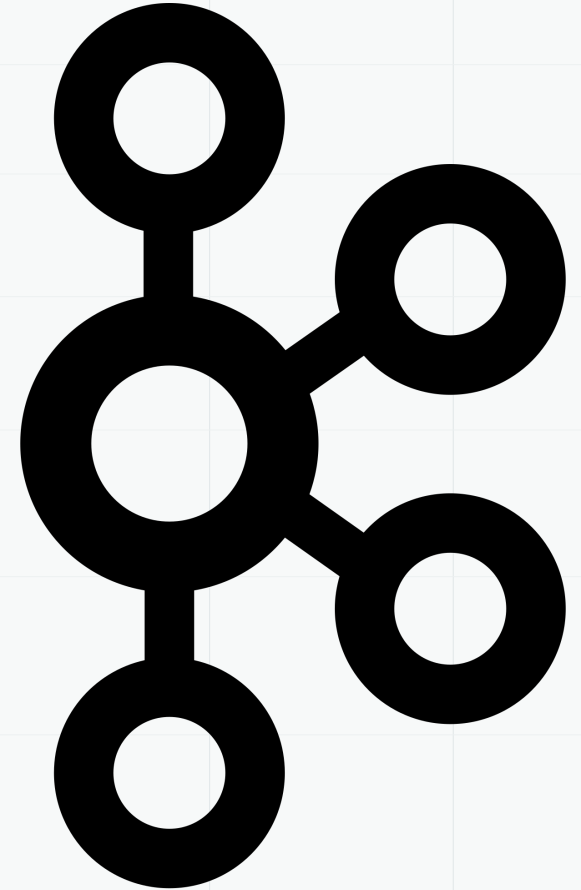


# Apache Kafka Overview



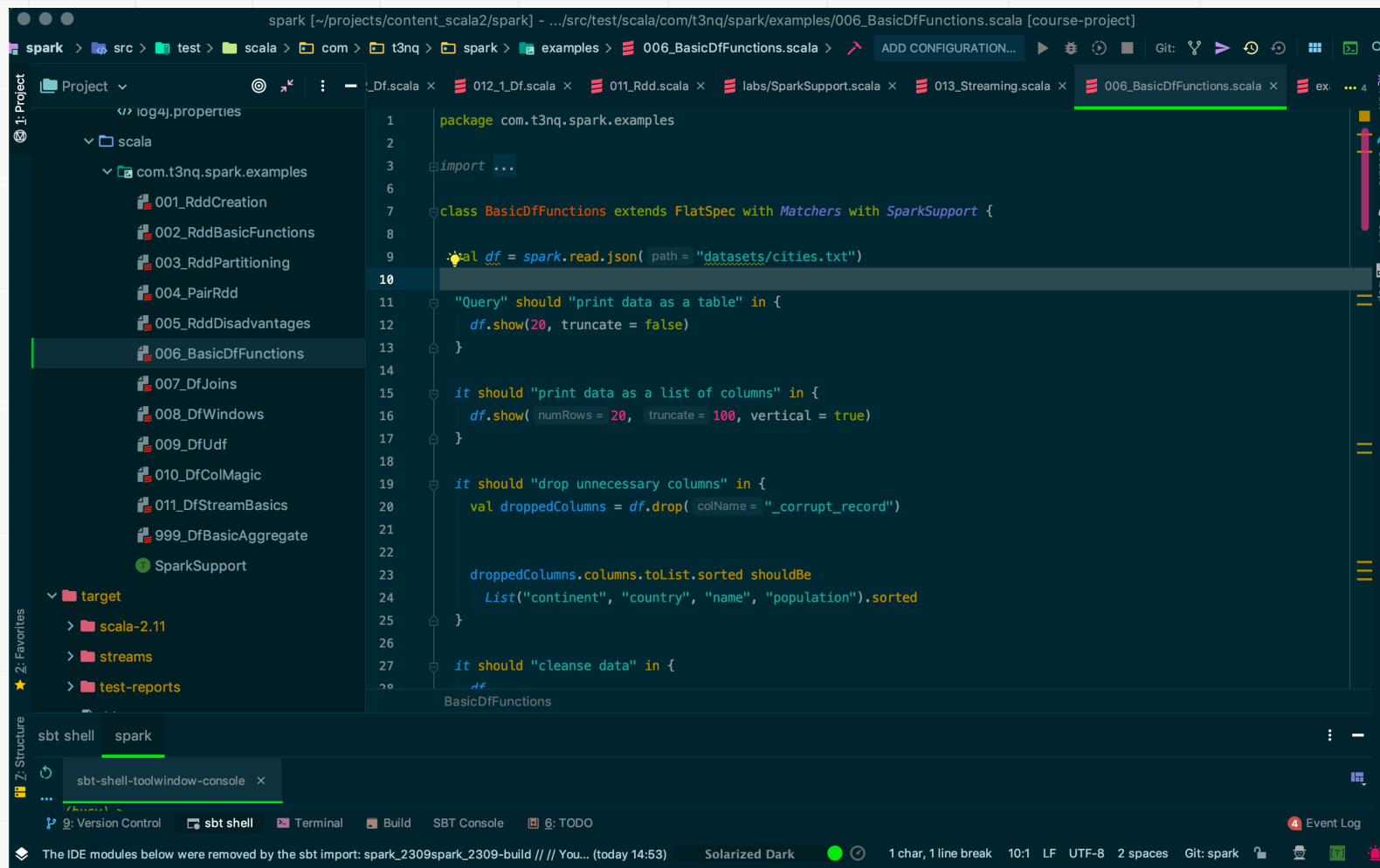
# Apache Kafka Overview

- Single Kafka cluster contains multiple topics
- Each topic consists of multiple partitions
- Message location is determined by (TOPIC, PARTITION, OFFSET) tuple
- Each partition contains replicas
- SINGLE Partition is always read by SINGLE consumer within consumer group



# Basic Streaming DF Functions

# Basic SDF Functions



```
spark [~/projects/content_scala2/spark] - .../src/test/scala/com/t3nq/spark/examples/006_BasicDfFunctions.scala [course-project]

spark > src > test > scala > com > t3nq > spark > examples > 006_BasicDfFunctions.scala > ADD CONFIGURATION...

Project
├── log4j.properties
├── scala
│   └── com.t3nq.spark.examples
│       ├── 001_RddCreation
│       ├── 002_RddBasicFunctions
│       ├── 003_RddPartitioning
│       ├── 004_PairRdd
│       ├── 005_RddDisadvantages
│       └── 006_BasicDfFunctions
│           ├── 007_DfJoins
│           ├── 008_DfWindows
│           ├── 009_DfUdf
│           ├── 010_DfColMagic
│           ├── 011_DfStreamBasics
│           └── 999_DfBasicAggregate
│               └── SparkSupport
├── target
│   ├── scala-2.11
│   ├── streams
│   └── test-reports
└── 2: Favorites
    ├── 3: Structure
    └── 4: sbt shell
        └── spark
            └── sbt-shell-toolwindow-console

1 package com.t3nq.spark.examples
2
3 import ...
4
5
6
7 class BasicDfFunctions extends FlatSpec with Matchers with SparkSupport {
8
9     val df = spark.read.json(path = "datasets/cities.txt")
10
11     "Query" should "print data as a table" in {
12         df.show(20, truncate = false)
13     }
14
15     it should "print data as a list of columns" in {
16         df.show(numRows = 20, truncate = 100, vertical = true)
17     }
18
19     it should "drop unnecessary columns" in {
20         val droppedColumns = df.drop(colName = "_corrupt_record")
21
22         droppedColumns.columns.toList.sorted shouldBe
23             List("continent", "country", "name", "population").sorted
24     }
25
26     it should "cleanse data" in {
27         df
28     }
29 }
```

The IDE modules below were removed by the sbt import: spark\_2309spark\_2309-build /// You... (today 14:53)

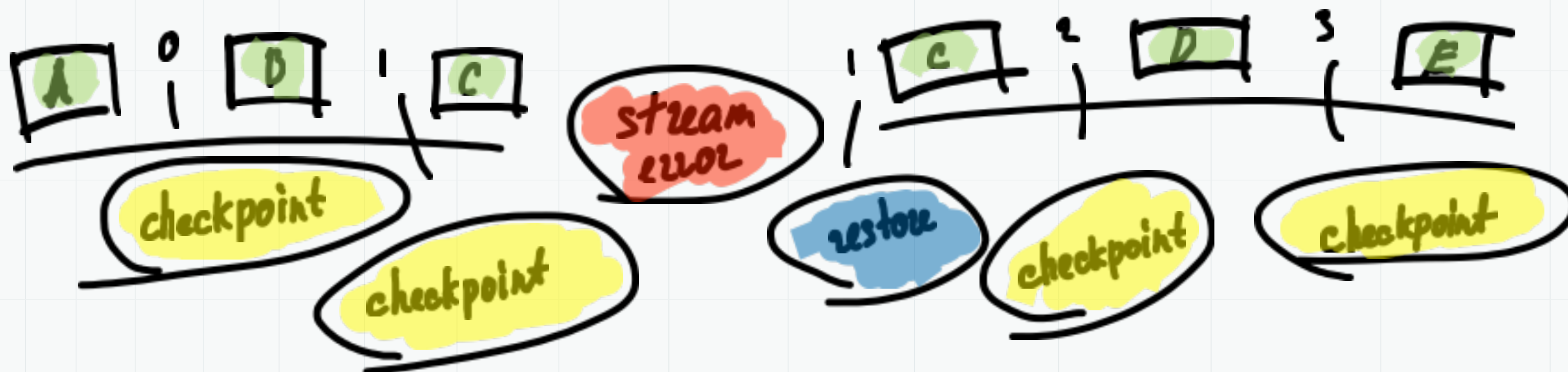
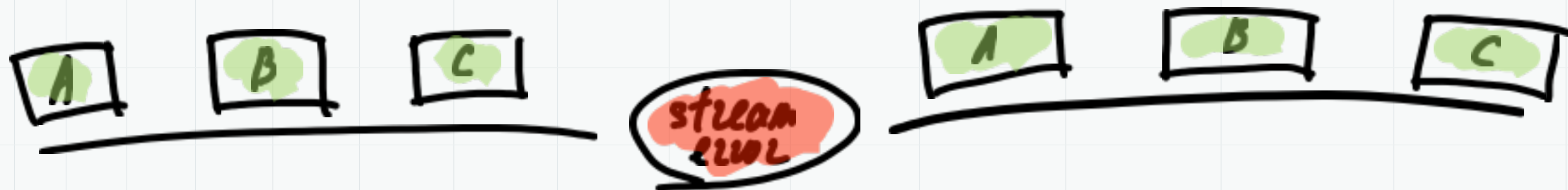
Solarized Dark 1 char, 1 line break 10:1 LF UTF-8 2 spaces Git: spark

# Persisting Stream State

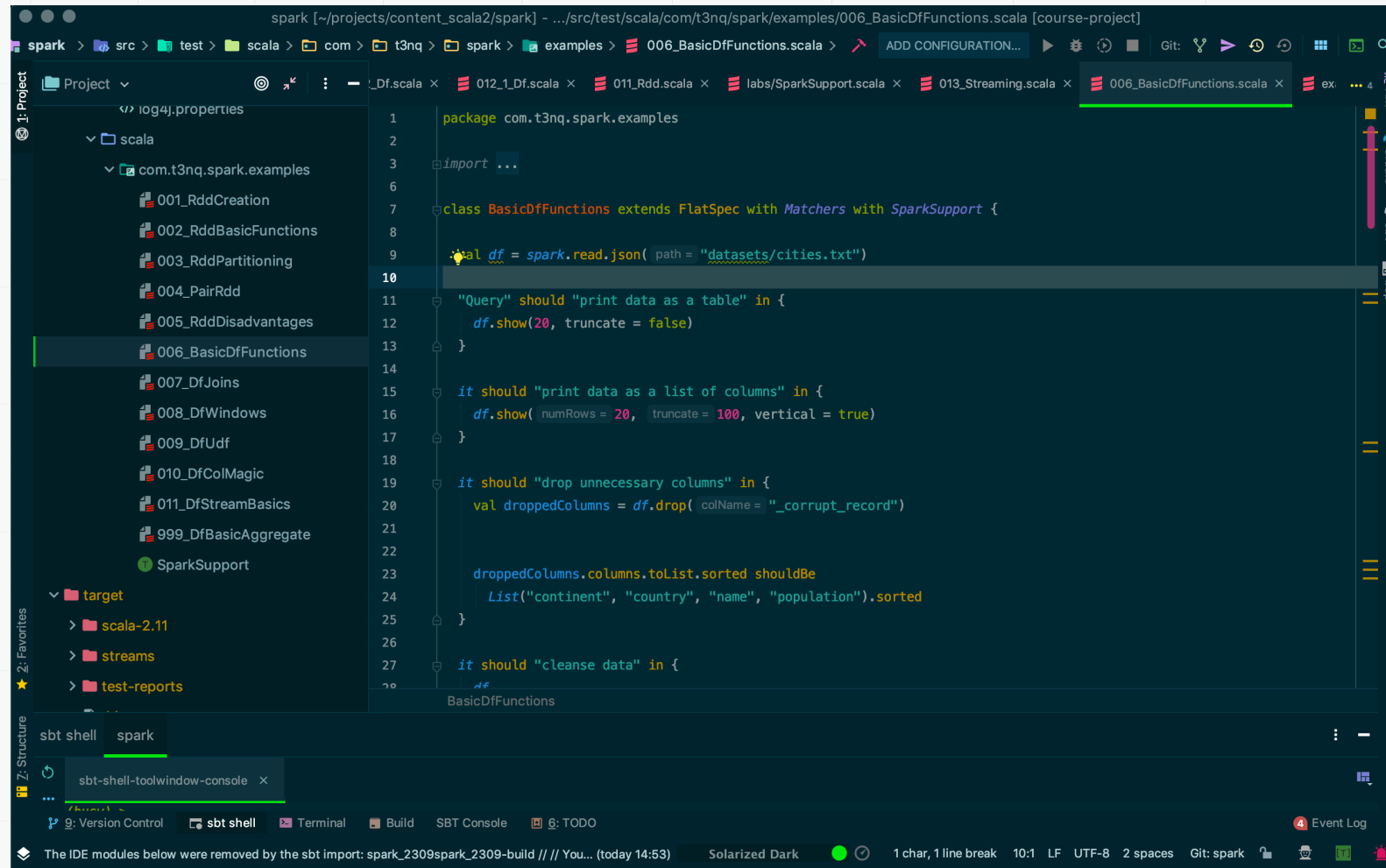
# Checkpoints



# Checkpoints



# Using Checkpoints



```
spark [~/projects/content_scala2/spark] - .../src/test/scala/com/t3nq/spark/examples/006_BasicDfFunctions.scala [course-project]

spark > src > test > scala > com > t3nq > spark > examples > 006_BasicDfFunctions.scala > ADD CONFIGURATION...

Project
├── log4j.properties
├── scala
│   └── com.t3nq.spark.examples
│       ├── 001_RddCreation
│       ├── 002_RddBasicFunctions
│       ├── 003_RddPartitioning
│       ├── 004_PairRdd
│       ├── 005_RddDisadvantages
│       └── 006_BasicDfFunctions
│           ├── 007_DfJoins
│           ├── 008_DfWindows
│           ├── 009_DfUdf
│           ├── 010_DfColMagic
│           ├── 011_DfStreamBasics
│           └── 999_DfBasicAggregate
│               └── SparkSupport
├── target
│   ├── scala-2.11
│   ├── streams
│   └── test-reports
└── 2: Favorites
    ├── scala-2.11
    ├── streams
    └── test-reports

1 package com.t3nq.spark.examples
2
3 import ...
4
5
6
7 class BasicDfFunctions extends FlatSpec with Matchers with SparkSupport {
8
9     val df = spark.read.json(path = "datasets/cities.txt")
10
11     "Query" should "print data as a table" in {
12         df.show(20, truncate = false)
13     }
14
15     it should "print data as a list of columns" in {
16         df.show(numRows = 20, truncate = 100, vertical = true)
17     }
18
19     it should "drop unnecessary columns" in {
20         val droppedColumns = df.drop(colName = "_corrupt_record")
21
22         droppedColumns.columns.toList.sorted shouldBe
23             List("continent", "country", "name", "population").sorted
24     }
25
26     it should "cleanse data" in {
27         df
28     }
29 }

BasicDfFunctions

sbt shell spark
sbt-shell-toolwindow-console x
```

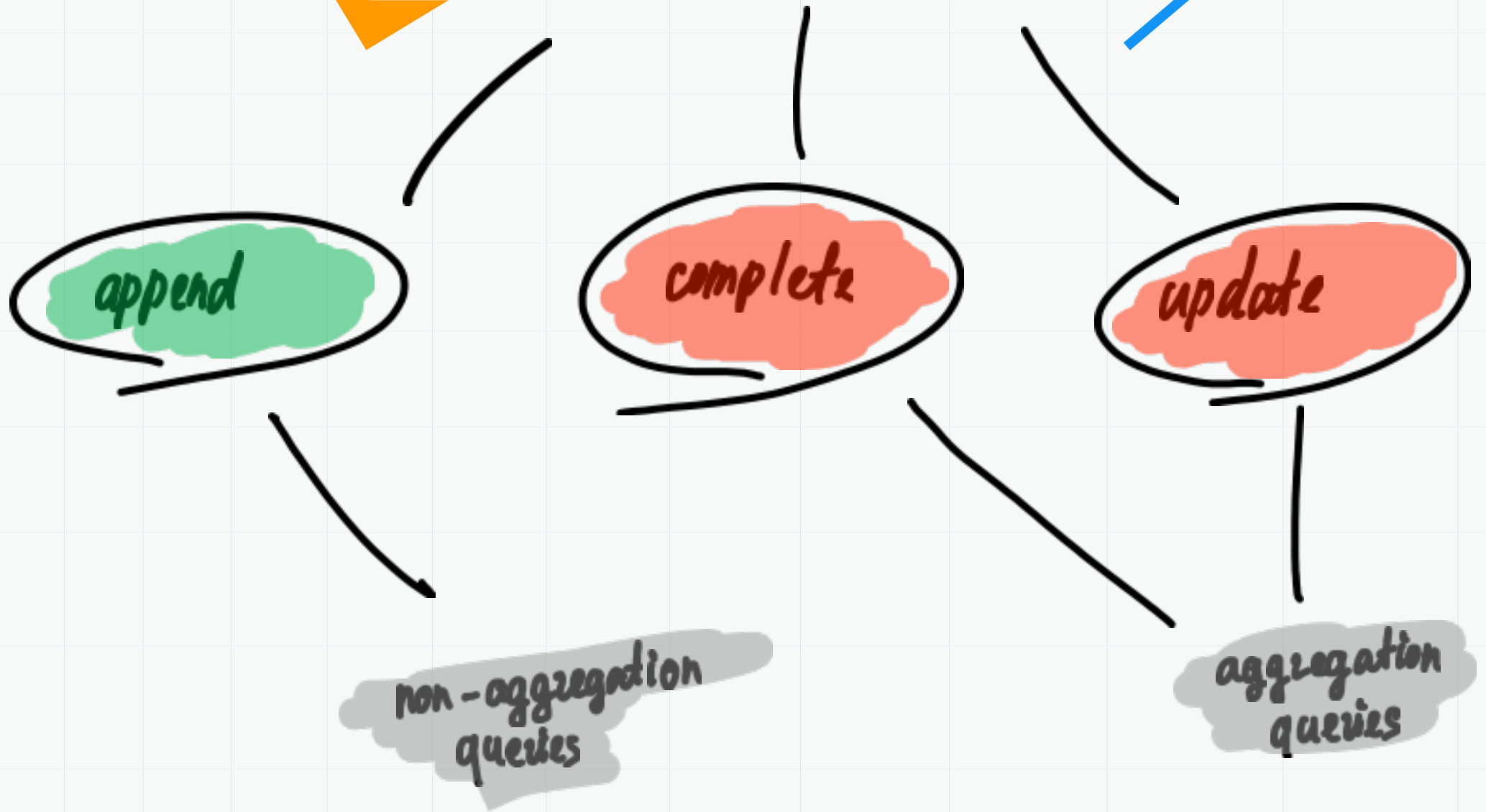
The IDE modules below were removed by the sbt import: spark\_2309spark\_2309-build /// You... (today 14:53)

Solarized Dark 1 char, 1 line break 10:1 LF UTF-8 2 spaces Git: spark



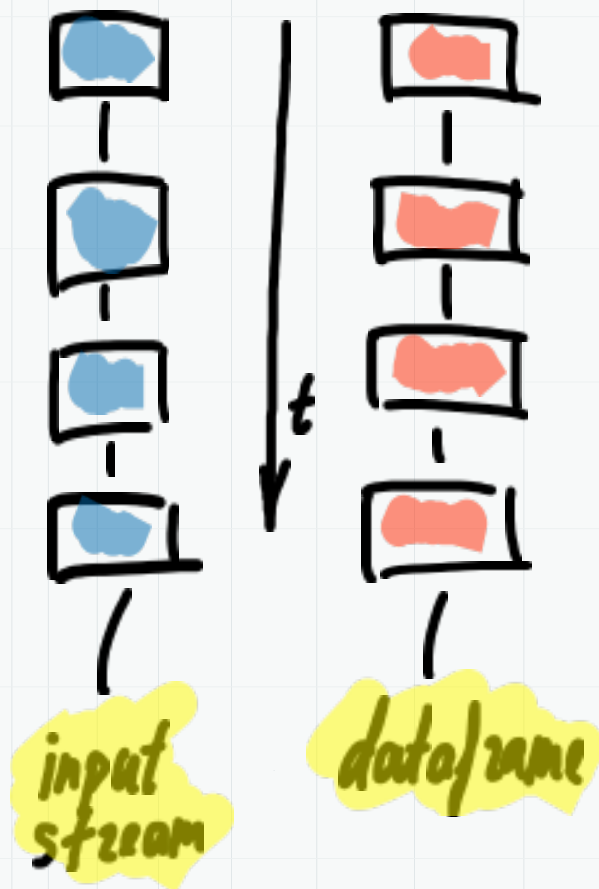
# Running aggregations queries

# Output Modes



# Output Modes

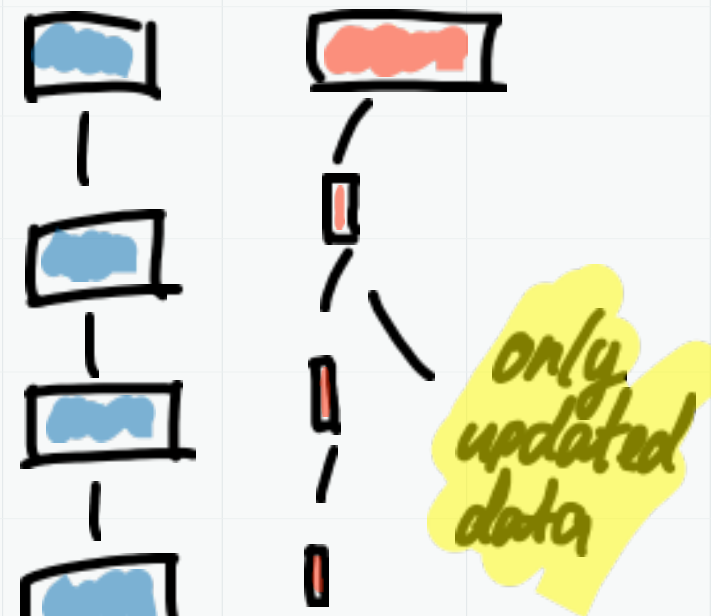
## Append



## Complete



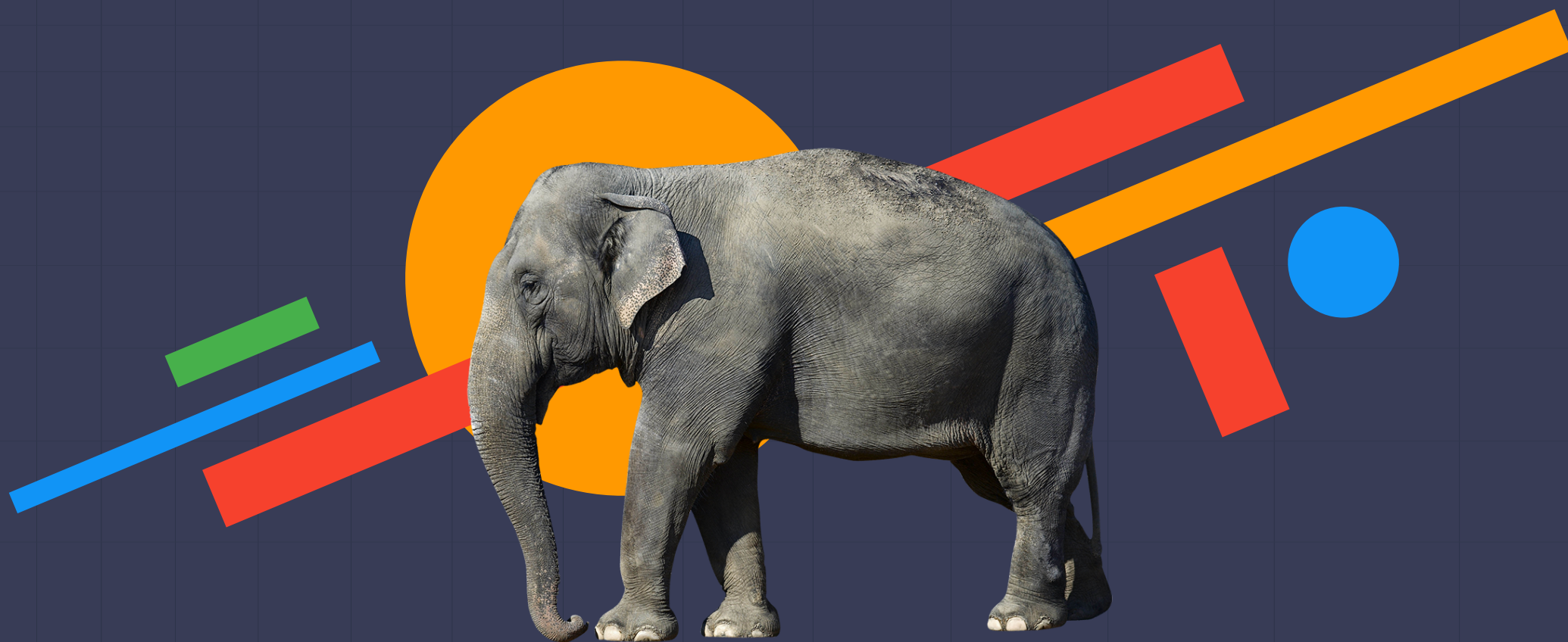
## Update



# Output Modes

```
1 package com.t3nq.spark.examples
2
3 import ...
4
5
6
7 class BasicDfFunctions extends FlatSpec with Matchers with SparkSupport {
8
9   val df = spark.read.json(path = "datasets/cities.txt")
10
11   "Query" should "print data as a table" in {
12     df.show(20, truncate = false)
13   }
14
15   it should "print data as a list of columns" in {
16     df.show(numRows = 20, truncate = 100, vertical = true)
17   }
18
19   it should "drop unnecessary columns" in {
20     val droppedColumns = df.drop(colName = "_corrupt_record")
21
22     droppedColumns.columns.toList.sorted shouldBe
23       List("continent", "country", "name", "population").sorted
24   }
25
26   it should "cleanse data" in {
27     // ...
28   }
29 }
```

The IDE modules below were removed by the sbt import: spark\_2309spark\_2309-build /// You... (today 14:53)



# Big Data is Love

NEWPROLAB.COM