# CS 696 Intro to Big Data: Tools and Methods
# Fall Semester, 2022
# Doc 29 Kafka Pipelines, Mircoservices
# May 3, 2022

# Big Data 3-5 V’s

Volume
- Large datasets

Clusters - Spark

Velocity
- Real time or near-real time streams of data

Kafka

Variety
- Different formats
- Structured, Numeric, Unstructured, images, email, etc.

NoSQL
Cassandra

Variability
- Data flows can be inconsistent

Veracity
- Accuracy

Complexity

# Microservices

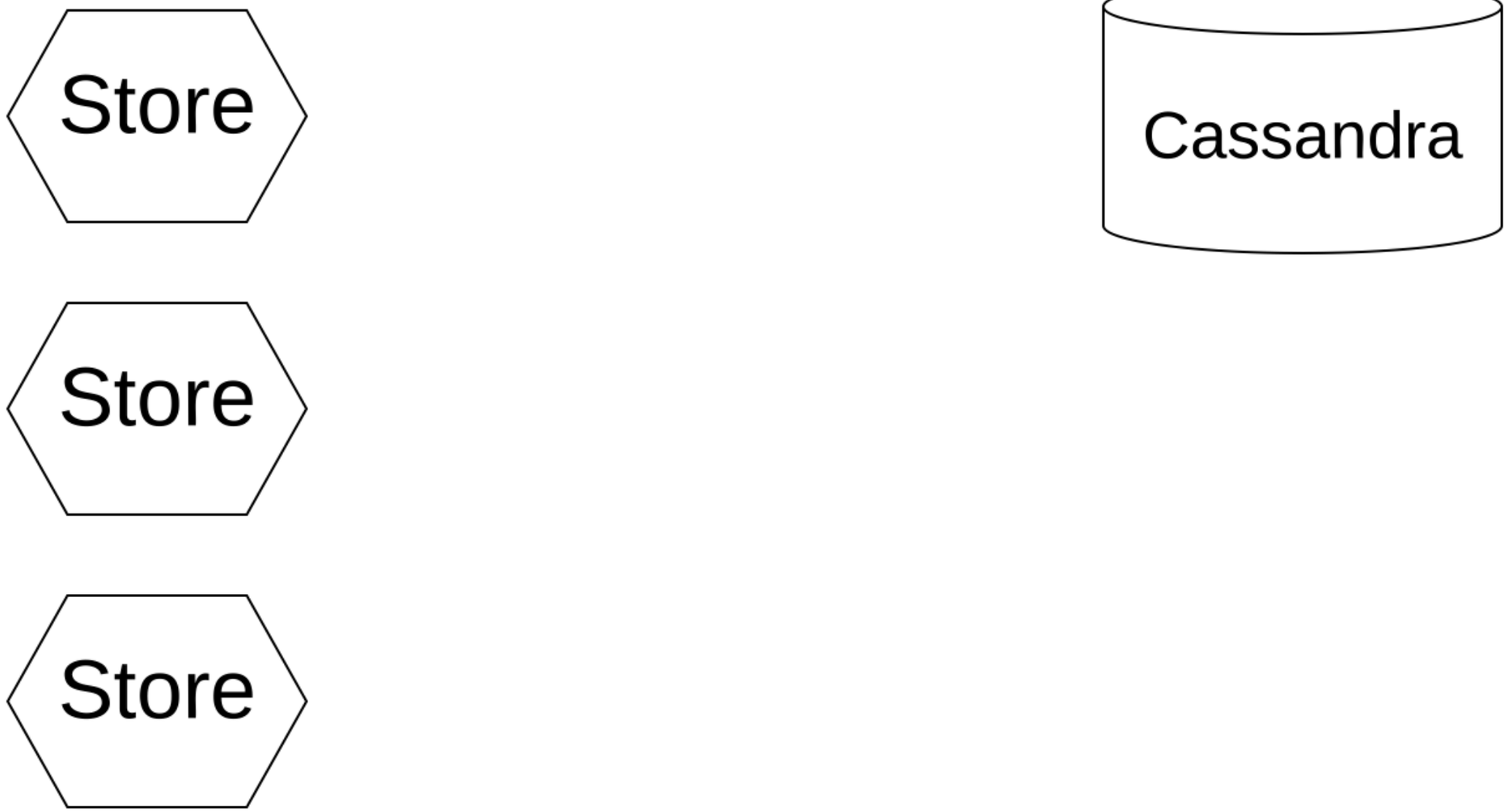
Application is a collection of small services

- Each running in its own process
- Loosely coupled
- Independently deployable
- Built around business capabilities

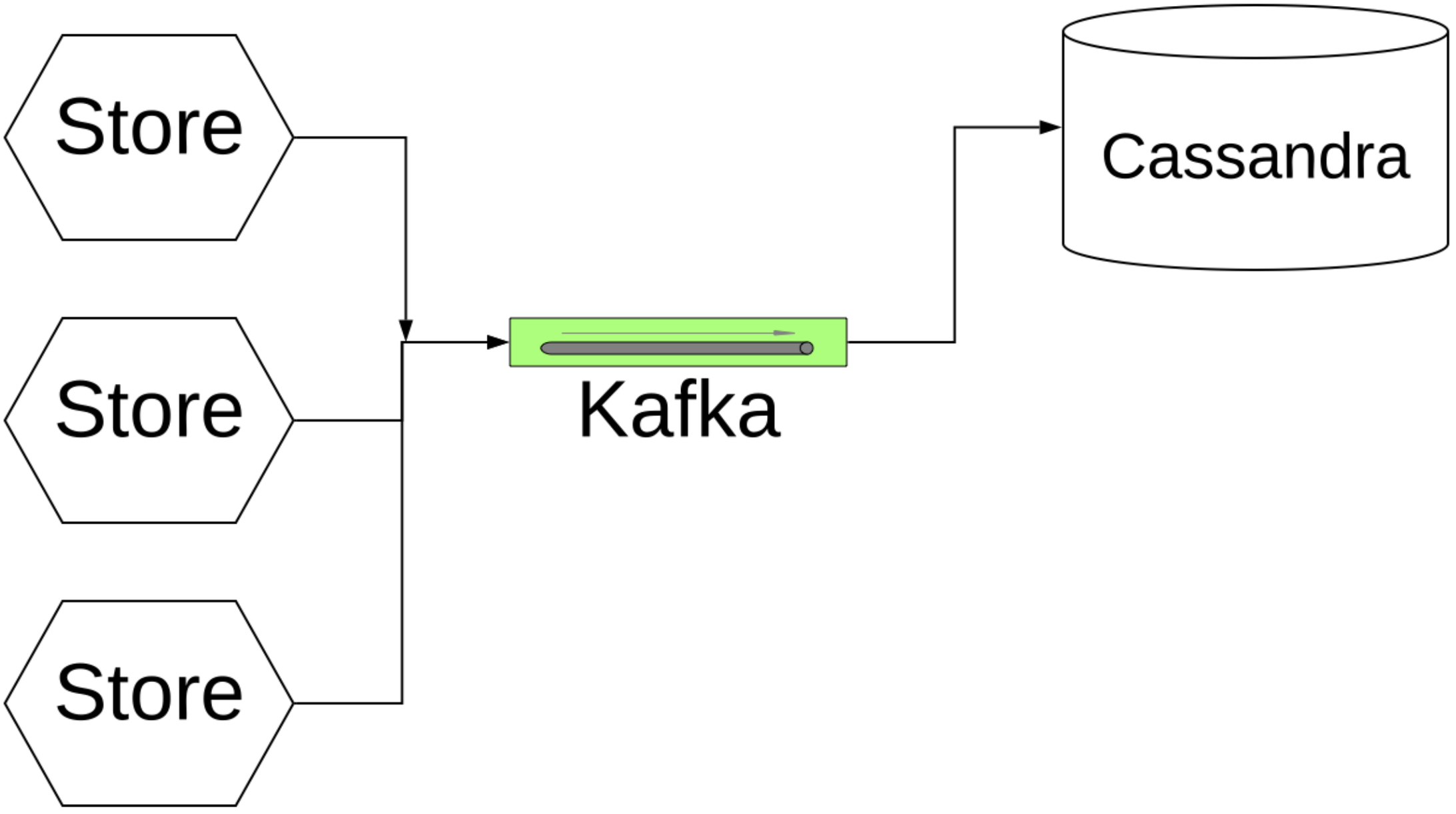
https://microservices.io

https://martinfowler.com/microservices/

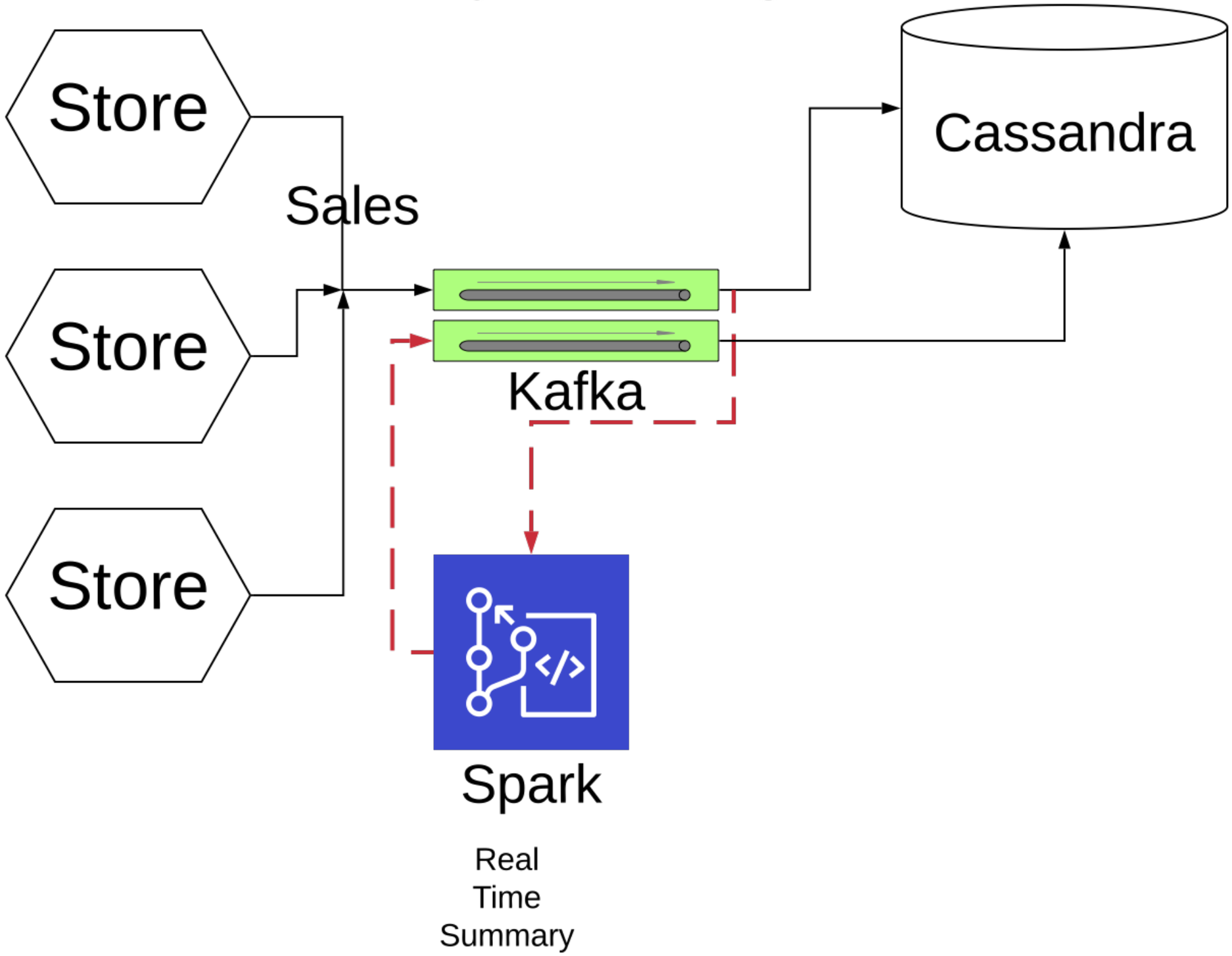
# Store Example



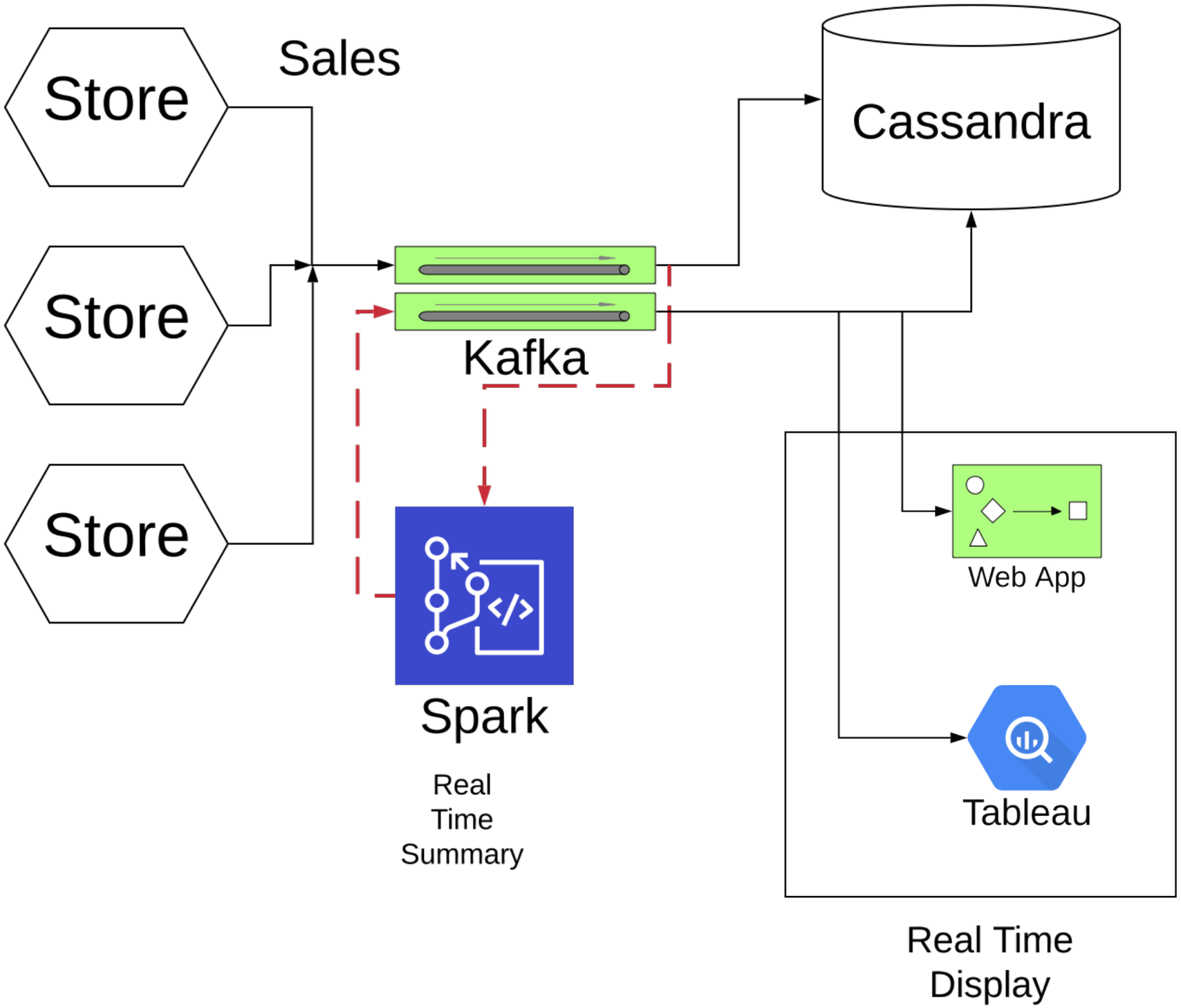
# Streaming Real Time Sales via Kafka



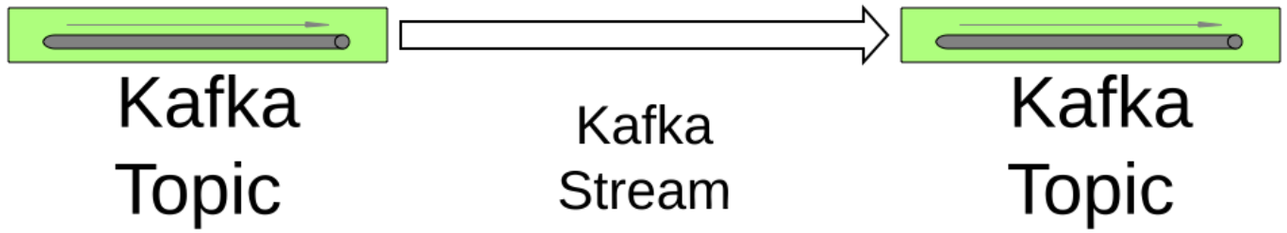
# Real Time Analytics Using Spark



# Displaying Real Time Analytics



# Kafka Streams




Kafka Stream

- Reads from Kafka topic
- Does some processing
- Writes to another Kafka topic

Runs on Kafka cluster

- Scalable
- Fault-tolerant

Java or Scala

# Stream Word Count Example

```
final Serde<String> stringSerde = Serdes.String();  // Serializers/deserializers
final Serde<Long> longSerde = Serdes.Long();

KStream<String, String> textLines = builder.stream(
    "streams-plaintext-input",                          // read from streams-plaintext-input topic
    Consumed.with(stringSerde, stringSerde)
  );

KTable<String, Long> wordCounts = textLines
    .flatMapValues(value -> Arrays.asList(value.toLowerCase().split("\\W+")))
    .groupBy((key, value) -> value)
    .count();

wordCounts.toStream().
            to("streams-wordcount-output", Produced.with(Serdes.String(), Serdes.Long()));
```

https://kafka.apache.org/25/documentation/streams/quickstart

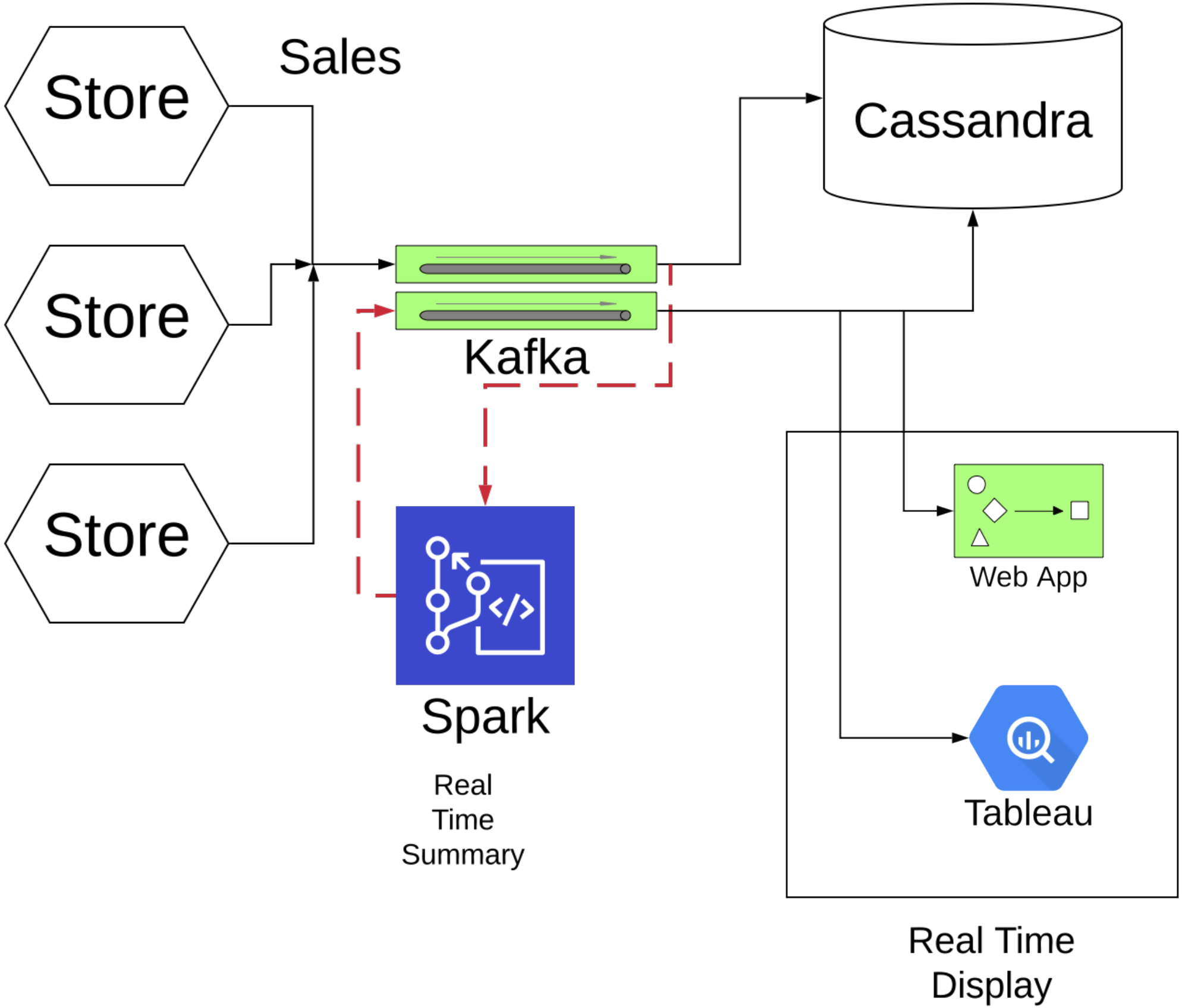
# Kafka Connect


Streaming data between Kafka and other systems

Runs on Kafka connect cluster

REST interface to manage connectors

Transformers to act on individual records


Store
Sales
Store
Store
Cassandra
Kafka
Spark
Real
Time
Summary
Web App
Tableau
Real Time
Display

# Example Using Python

Store sell two products

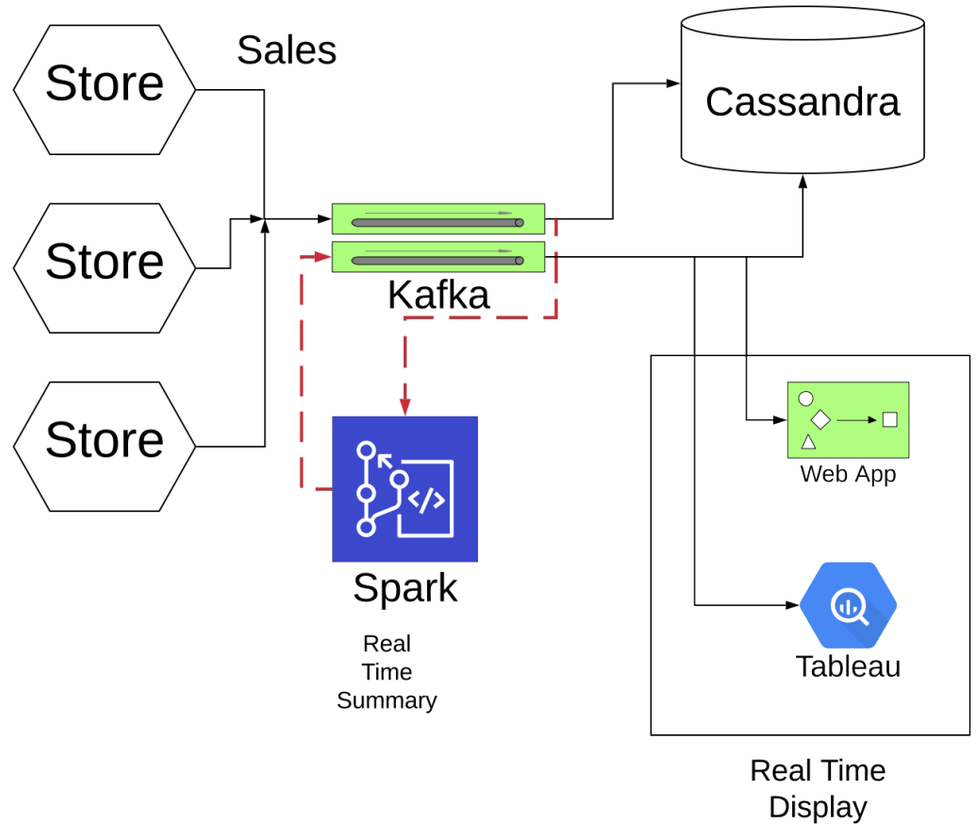
a

b

Each time a store sell an item sends to Kafka

JSON object

{'item': 'a', 'amount': 20, 'time': 1}



# Starting Servers

Al pro 14->bin/zookeeper-server-start.sh config/zookeeper.properties

Al pro 15->bin/kafka-server-start.sh config/server.properties

Al pro 20->cassandra -f

# Create kafka topics

bin/kafka-topics.sh --create --zookeeper localhost:2181
--replication-factor 1 --partitions 1 --topic **sales**

bin/kafka-topics.sh --create --zookeeper localhost:2181
--replication-factor 1 --partitions 1 --topic **sales-summary**

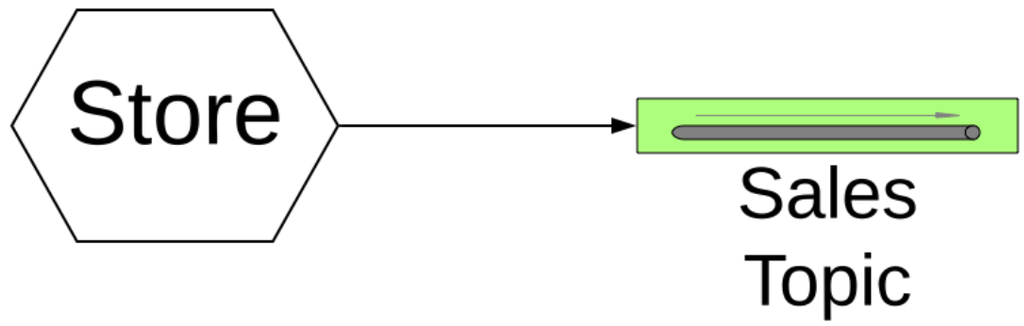
# Cassandra Setup

```
cqlsh> create keyspace store with replication = {'class': 'SimpleStrategy', 'replication_factor': 1}

cqlsh:store> create table sales (item text, amount int, time int PRIMARY KEY);
```

# Fake Store

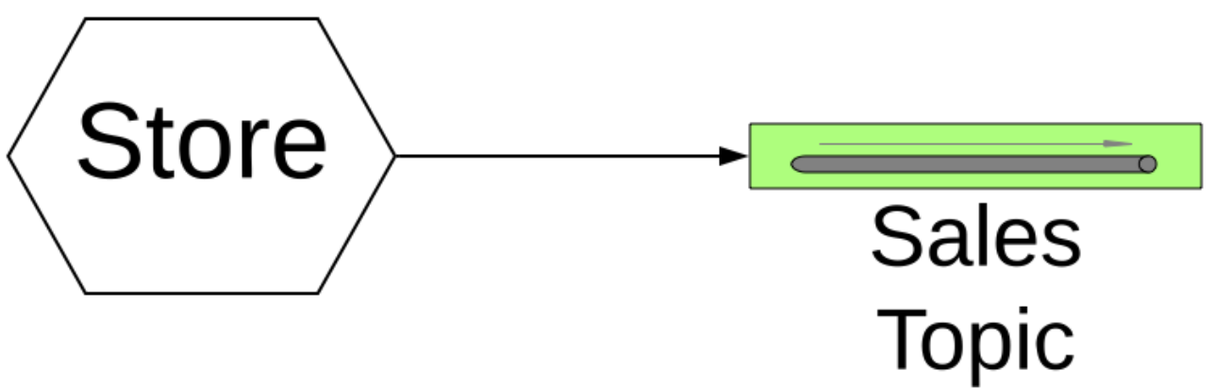
```
import numpy as np
import json
from kafka import KafkaProducer
```

```
def item(p, rng):           # Select item at random
    selected = rng.binomial(2,p)
    if selected == 1:
        return 'a'
    return 'b'

def amount(rng):            # How many were bought
    return int(rng.lognormal(2, 1))

def purchase(rng):
    what = item(.7,rng)
    quantity = amount(rng)
    return what, quantity

def store(n, rng, kafka):
    for count in range(0,n):
        what, quantity = purchase(rng)
        kafka.send('sales', {'item': what, 'amount': quantity,'time': count})
```



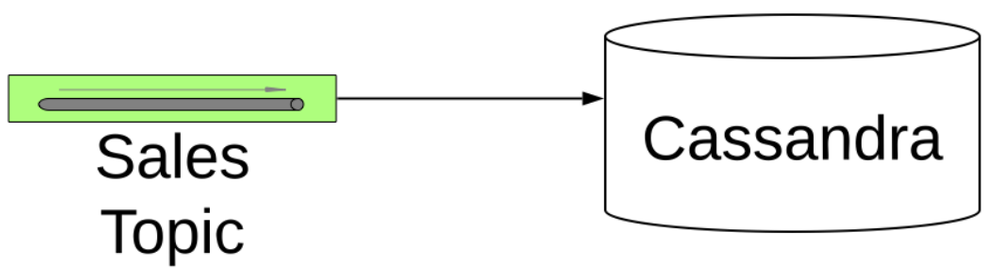
# Running the Store

```
rng = np.random.default_rng()
producer = KafkaProducer(bootstrap_servers='localhost:9092',
                         value_serializer=lambda m: json.dumps(m).encode('ascii'))

store(5000,rng,producer)
```

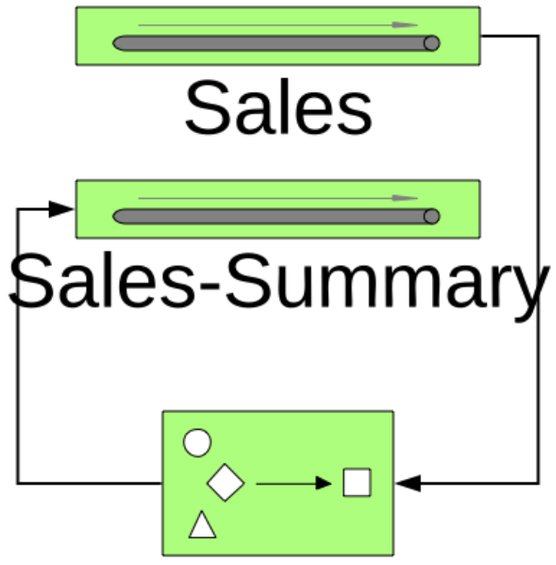


# Adding to Cassandra

```
import json
from kafka import KafkaConsumer
from cassandra.cluster import Cluster

sales = KafkaConsumer('sales',
                      bootstrap_servers='localhost:9092',
                      value_deserializer=lambda m: json.loads(m.decode('ascii')))

cluster = Cluster()
session = cluster.connect('store')

def add_to_db(sales_dic, session):
    session.execute(f"INSERT INTO sales (item, amount, time)
                        VALUES ('{sales_dic['item']}', {sales_dic['amount']}, {sales_dic['time']})")

for msg in sales:
    add_to_db(msg.value, session)
```



# Computing Total For 5 Time Units




```
import json
from kafka import KafkaProducer
from kafka import KafkaConsumer

sales = KafkaConsumer('sales',
                      bootstrap_servers='localhost:9092',
                      value_deserializer=lambda m: json.loads(m.decode('ascii')))

sales_summary = KafkaProducer(bootstrap_servers='localhost:9092',
                              value_serializer=lambda m: json.dumps(m).encode('ascii'))

total_amount = 0

for msg in sales:
    total_amount = total_amount + msg.value['amount']
    if (msg.value['time'] % 5) == 0:
        sales_summary.send('sales-summary',{'amount': total_amount,'time': msg.value['time']})
        total_amount = 0
```

# Importing Data to Tableau