Cassandra and Kafka

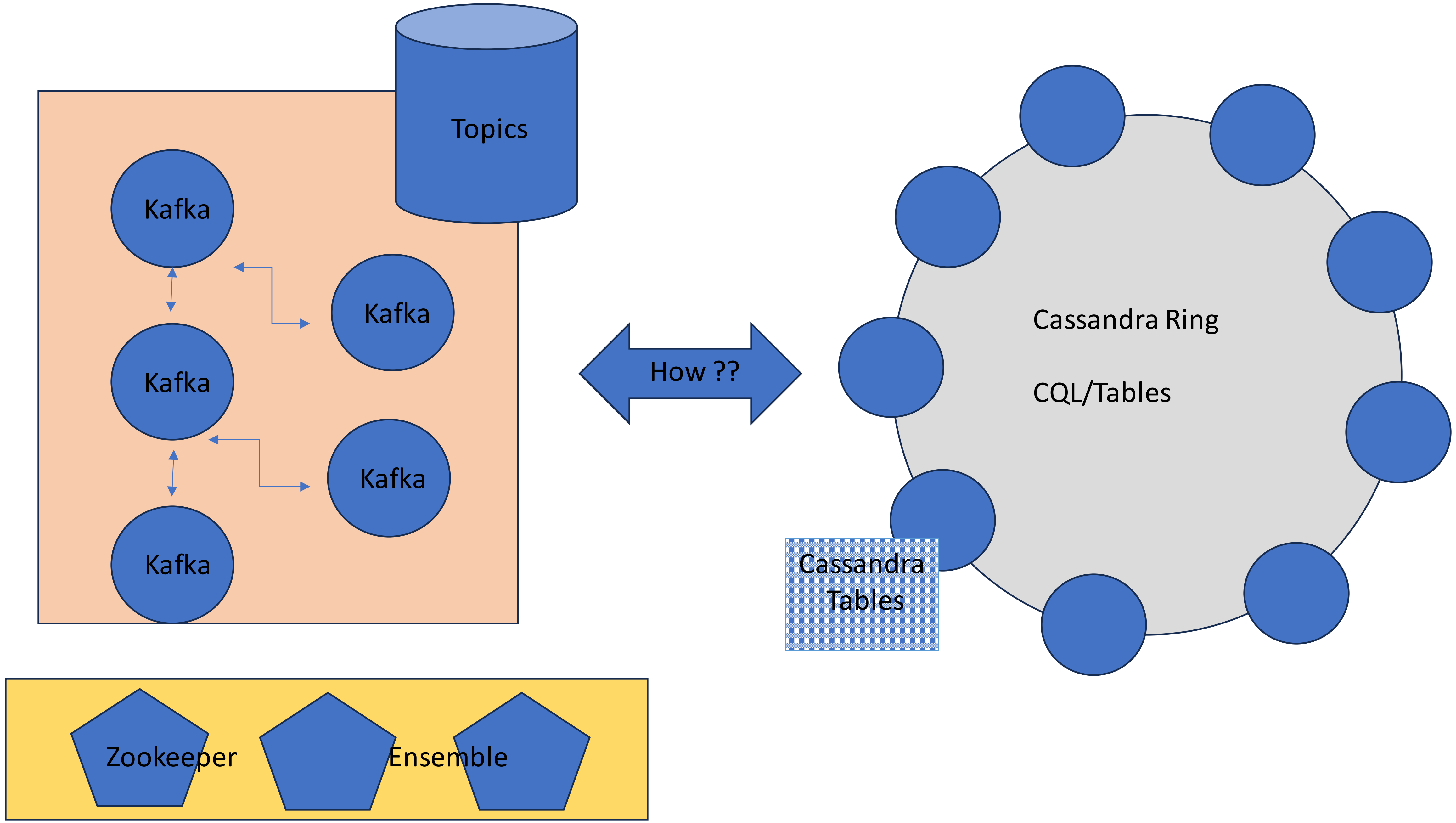
Two peas in a pod

Big Picture I

- Cassandra NoSQL performant database, still has one of the fastest I/O writes in the industry
- Kafka – zero copy technology, LinkedIn processes trillions of messages through their Kafka ecosystem
- Challenge how to get Kafka and Cassandra play nicely with each other
- **Complex scenario** – however, Kafka Connect to the rescue

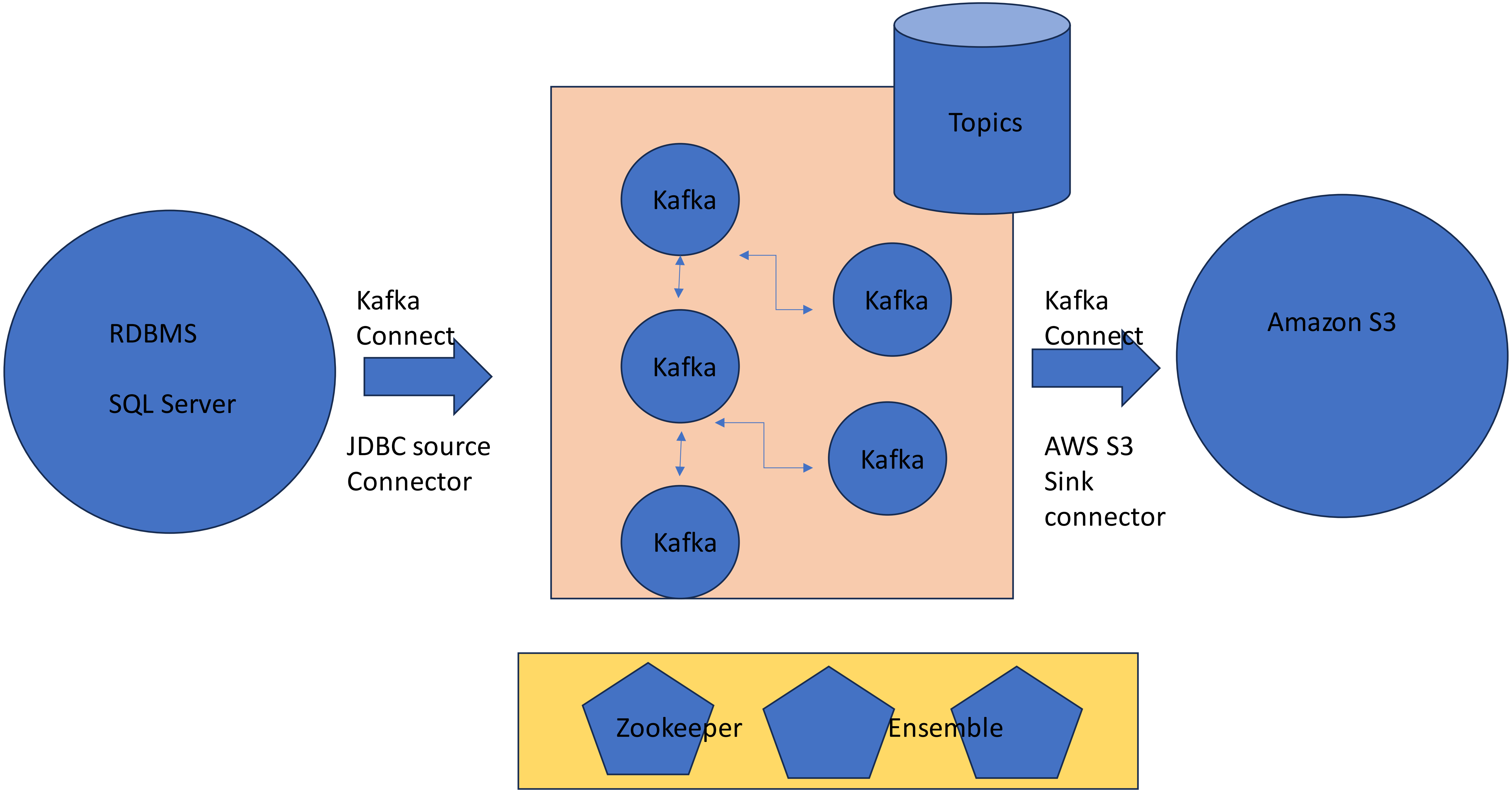
Big Picture II

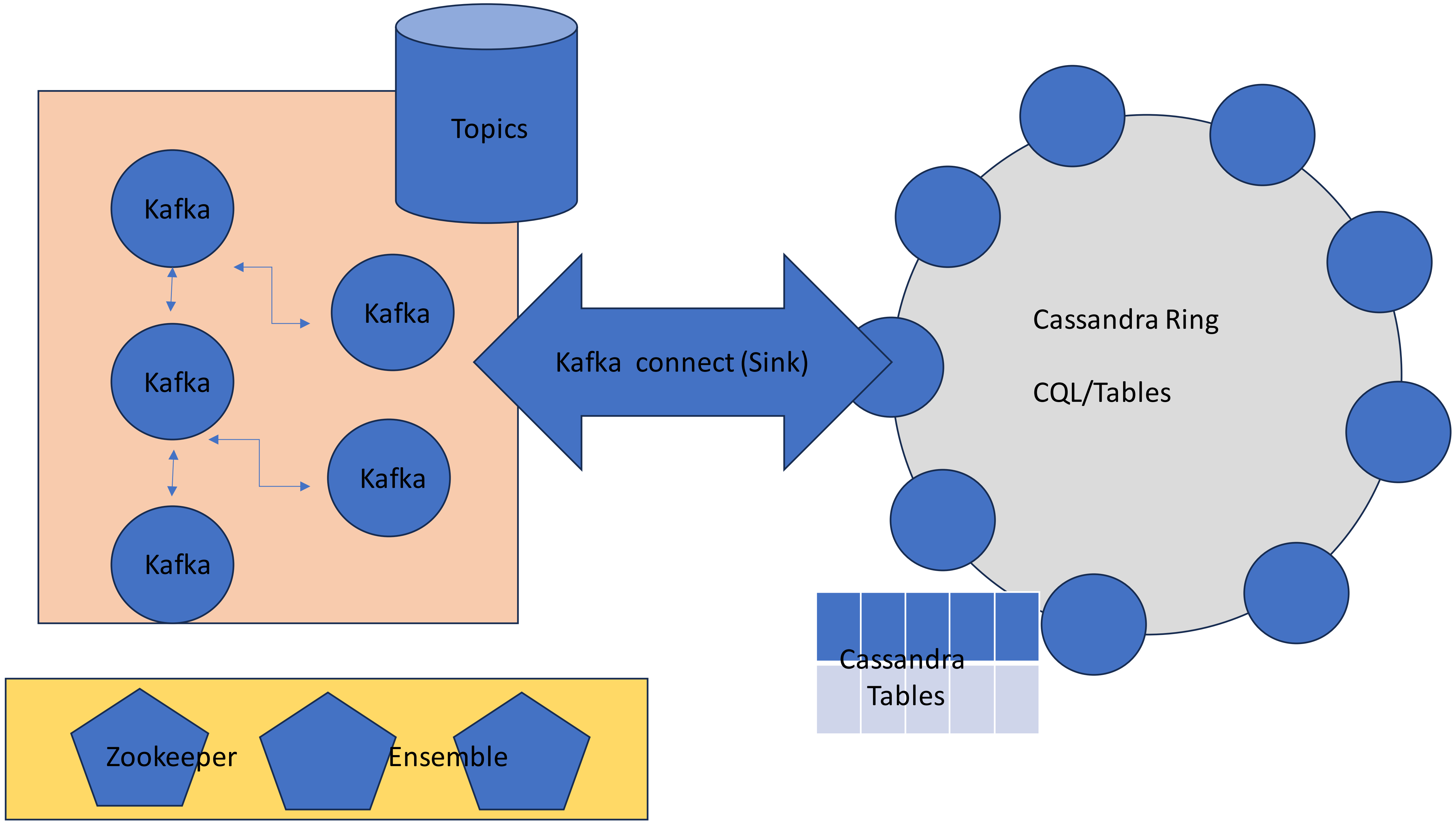
- Topics are the currency in Kafka
- Tables are the currency in Cassandra
- We want to do this
- Cassandra -> Kafka -> Cassandra
- Cassandra -> Kafka Connect -> Kafka -> Kafka Connect -> Cassandra
- Kafka Connect is the digital glue
- We will examine Kafka -> Kafka Connect (Sink) -> Cassandra
- Cassandra -> Kafka Connect(Source) -> Kafka (2nd half of slide deck)


Topics
Kafka
Kafka
Kafka
Kafka
Kafka
How ??
Cassandra Ring
CQL/Tables
Cassandra Tables
Zookeeper
Ensemble

Man, this is complex

Yes and No KAFKA CONNECT TO THE RESCUE

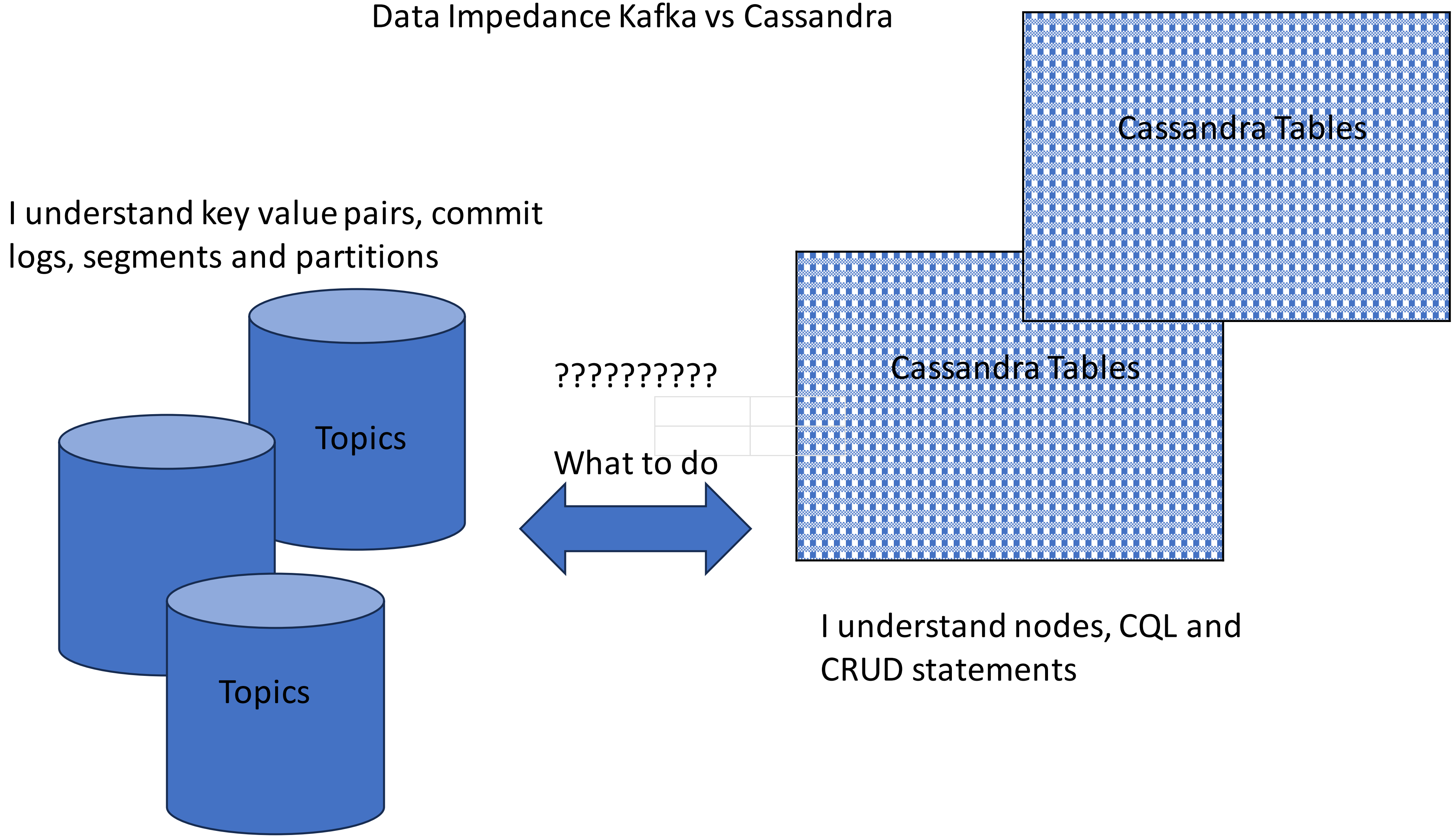

RDBMS
SQL Server
Kafka Connect
JDBC source Connector
Topics
Kafka
Kafka
Kafka
Kafka
Kafka
Kafka Connect
AWS S3 Sink connector
Amazon S3
Zookeeper
Ensemble


Topics
Kafka
Kafka
Kafka
Kafka
Kafka
Kafka connect (Sink)
Cassandra Ring
CQL/Tables
Cassandra Tables
Zookeeper
Ensemble

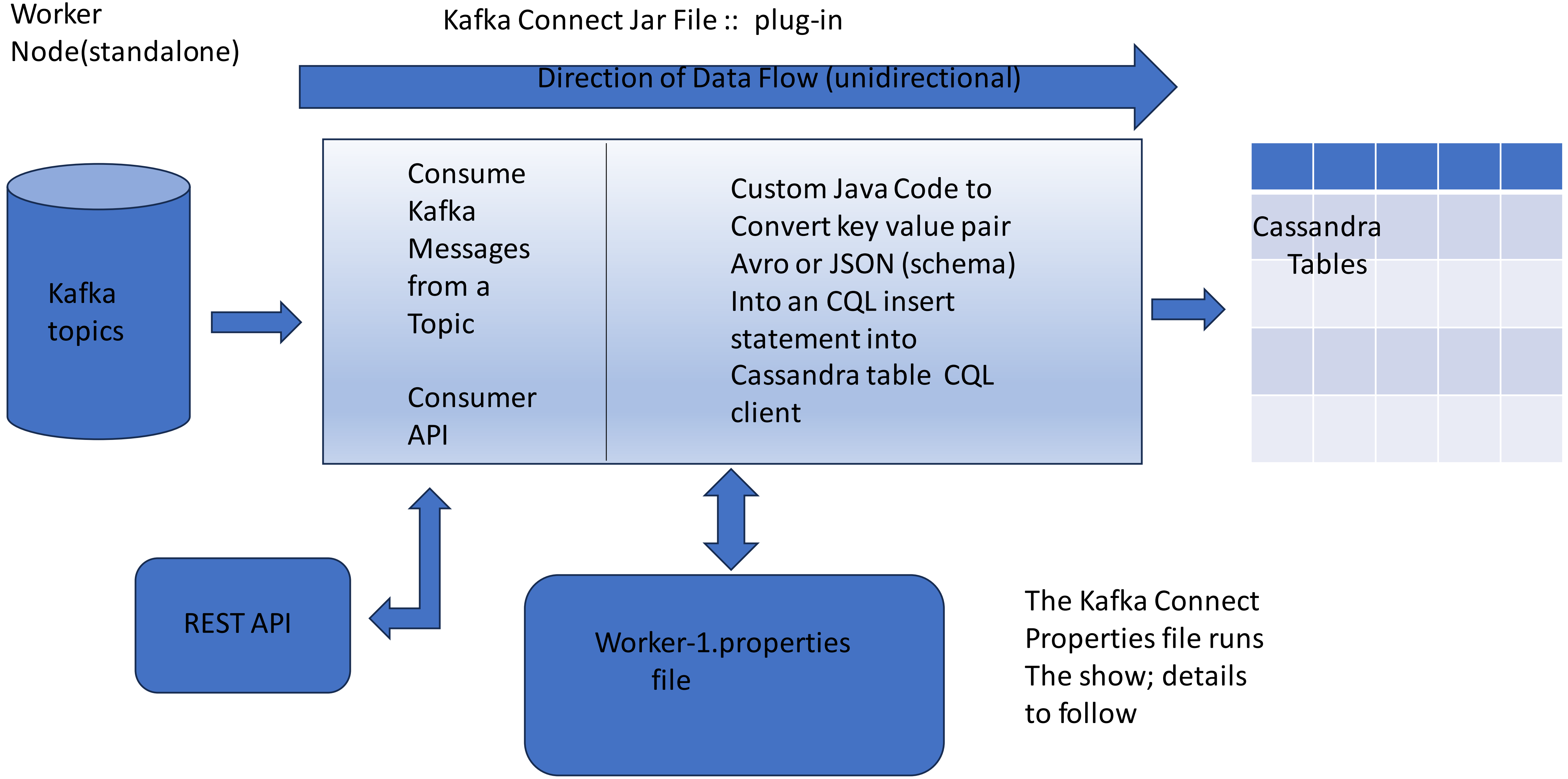
Kafka Connect(Sink)

- How much technology ? It is the properties file, stupid !!!
- Good news, no Java coding, just config files
- Two flavors – standalone and distributed
- Standalone good for dev, qa1 and qa2
- Distributed good for staging and prod
- Question – how do we go from Kafka topics to Cassandra tables
- All we have are 5 Kafka APIs
- Kafka Connect(Sink) is a Swiss Army knife – a smart toolset

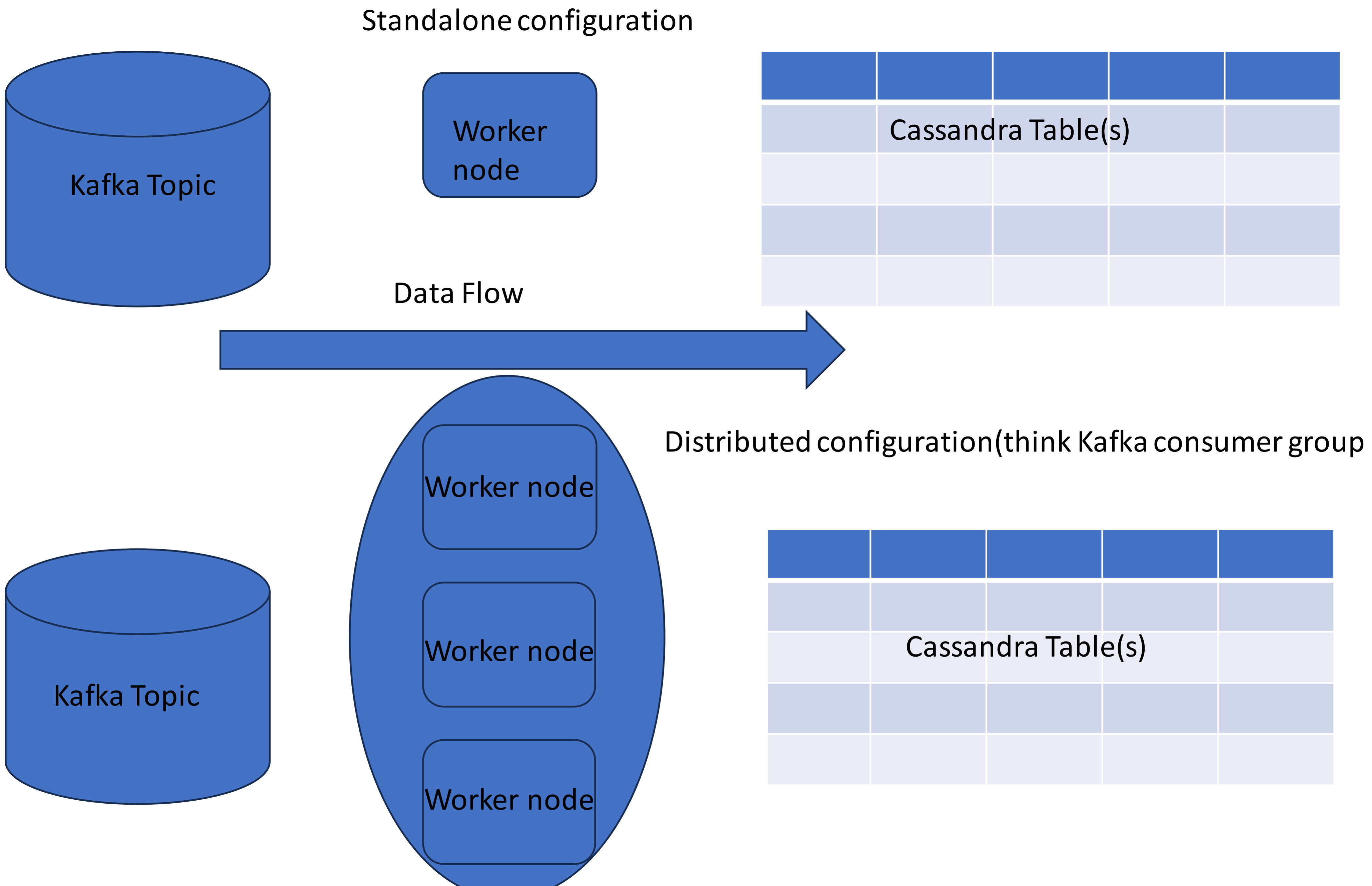
Data Impedance Kafka vs Cassandra


I understand key value pairs, commit logs, segments and partitions
Topics
Topics
??????????
What to do
Cassandra Tables
Cassandra Tables
I understand nodes, CQL and CRUD statements

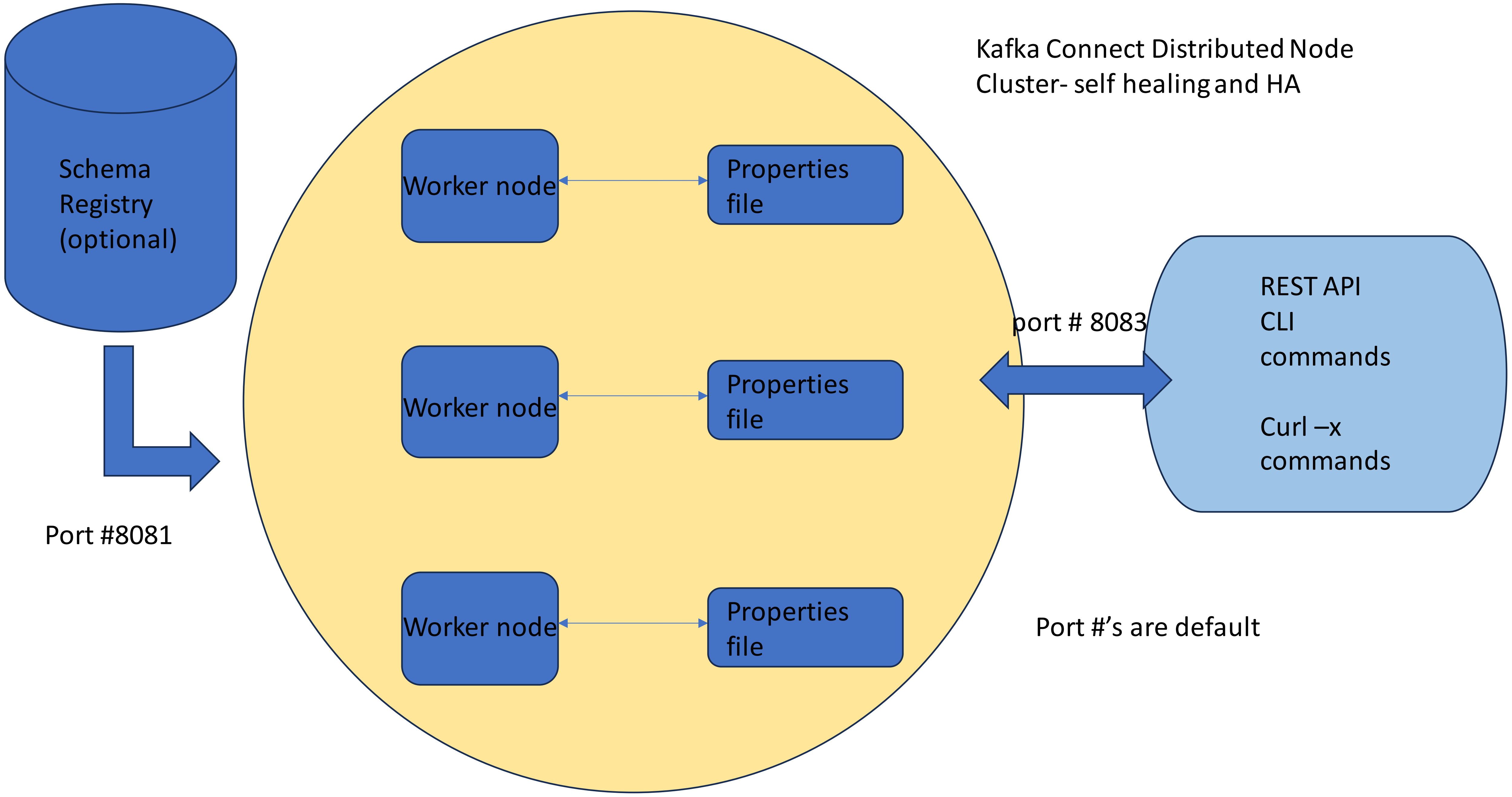
Kafka Connect Worker Node(standalone)


Kafka Connect Jar File :: plug-in
Direction of Data Flow (unidirectional)
Kafka topics
Consume Kafka Messages from a Topic
Consumer API
Custom Java Code to Convert key value pair Avro or JSON (schema) Into an CQL insert statement into Cassandra table CQL client
Cassandra Tables
REST API
Worker-1.properties file

The Kafka Connect Properties file runs The show; details to follow


Standalone configuration
Kafka Topic
Worker node
Cassandra Table(s)
Data Flow
Distributed configuration(think Kafka consumer group
Worker node
Worker node
Worker node
Kafka Topic
Cassandra Table(s)

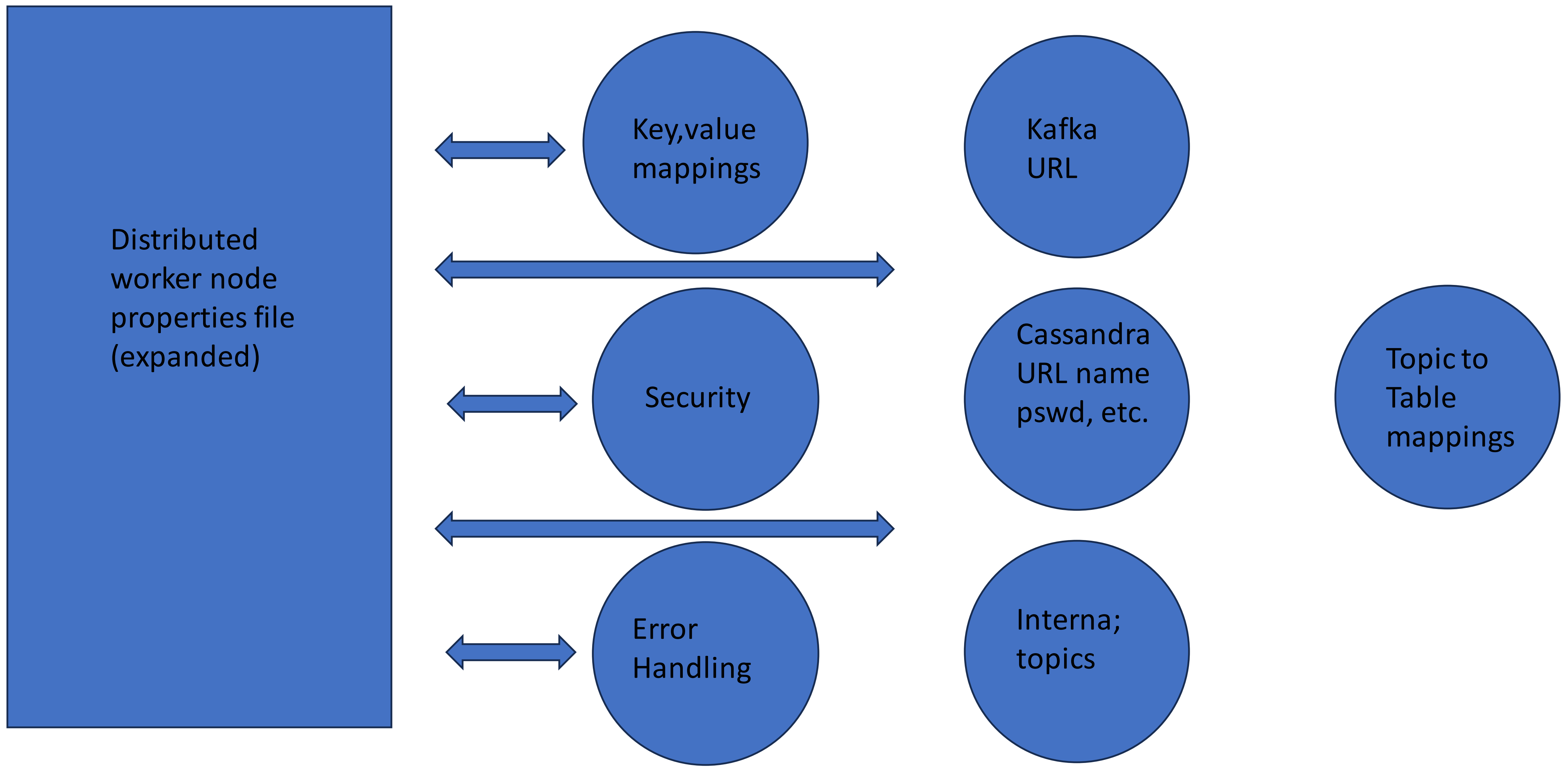
Kafka Connect Distributed Node Cluster- self healing and HA



Port #'s are default

Deep dive into worker node

- one property file per worker node
- principal use of distributed configuration – load balancing of data traffic ingestion into the Cassandra ring; don't worry Cassandra can keep up
- remember the secret sauce is the plug-in jar file does tge heavy lifting
- Property file points to Kafka broker list; Cassandra port/url string(aha!)
- Property file controls mapping of Kafka topics to Cassandra tables
- One worker node per Cassandra node for Cassandra keyspace
- Good news, no Java or Python scripting, some complexity in config file(s)


Distributed worker node properties file (expanded)
Key,value mappings
Kafka URL
Security
Cassandra URL name pswd, etc.
Topic to Table mappings
Error Handling
Interna; topics

Where to get the jar files(plug-in)

- go to **http://www.confluent.io/hub/**
- in the search bar, type Cassandra Sink Connector; hit enter key
- follow the directions for the download(s)
- save the jar file(plug-in) copy the file into the worker node directory
- it is fully supported by Confluent
- don't be alarmed the zip file has a /lib directory of about 30+ jar files; transfer everything
- copy the entire directory (copy this library into the worker node directory, e.g., /usr/kafka/plugin

# General worker configuration	Name of the worker node ,group id
# Kafka and Cassandra converters configuration	Set up the key value converters
# Set the plugin path to the directory containing Cassandra Sink Connector JAR	Point to the magic sauce plugin jar filesssserrrr
# Cassandra Sink Connector configuration for Worker 1	Self explanatory
# Cassandra connection details for Worker 1	Pointers to the key space, username,password, port, etc.
# Kafka topic to Cassandra table mapping for Worker 1	Mapping from Kafka topic(s) to Cassandra table(s)
#Security	SASL/SSL/truststores(tbd)

```
Name=worker-1
Bootstrap.servers=kafka-broker-1:9092, kafka-broker-2:9092;kafka-broker-3:9092
group-.id=connect-cluster
#Kafka and Cassandra converters configuration
key.converter=org.apache.kafka.connect.json.JsonConverter
value.converter=org.apache.kafka.connect.json.JsonConverter
Key.converter.schemas.enable=false
Value.converter.schemas.enable=false

#set the plugin path to the directory containing Cassandra Sink Connector
Plug-in.path=/path/to/connectors

#Cassandra Sink Connector configuration for worker-1
Connector.class=io.confluent.connect.Cassandra.CassandraSinkConnector
tasks=1

#Cassandra connection details for worker -1
Cassandra.connection.host=cassandra-node-1
Cassandra.connection.port=9042
Cassandra.connection.username=my-username
Cassandra.connection.password=my-password
Cassandra.keyspace=my_keyspace
```

```
#Kafka topic to Cassandra table mapping for worker -1

topic.myworker_1-topic_1=Cassandra_table_1
topic.myworker_1-topic_2=Cassandra_table_2
topic.myworker_1-topic_3=Cassandra_table_3
```

```
Name=worker-2
Bootstrap.servers=kafka-broker-1:9092, kafka-broker-2:9092;kafka-broker-3:9092
group-.id=connect-cluster
#Kafka and Cassandra converters configuration
key.converter=org.apache.kafka.connect.json.JsonConverter
value.converter=org.apache.kafka.connect.json.JsonConverter
Key.converter.schemas.enable=false
Value.converter.schemas.enable=false

#set the plugin path to the directory containing Cassandra Sink Connector
Plug-in.path=/path/to/connectors

#Cassandra Sink Connector configuration for worker-2
Connector.class=io.confluent.connect.Cassandra.CassandraSinkConnector
tasks=1

#Cassandra connection details for worker -1
Cassandra.connection.host=cassandra-node-1
Cassandra.connection.port=9042
Cassandra.connection.username=my-username
Cassandra.connection.password=my-password
Cassandra.keyspace=my_keyspace
```

```
#Kafka topic to Cassandra table mapping for worker -1

topic.myworker_2-topic_1=Cassandra_table_1
topic.myworker_2-topic_2=Cassandra_table_2
topic.myworker_2-topic_3=Cassandra_table_3
```

```
Name=worker-3
Bootstrap.servers=kafka-broker-1:9092, kafka-broker-2:9092;kafka-broker-3:9092
group-.id=connect-cluster
#Kafka and Cassandra converters configuration
key.converter=org.apache.kafka.connect.json.JsonConverter
value.converter=org.apache.kafka.connect.json.JsonConverter
Key.converter.schemas.enable=false
Value.converter.schemas.enable=false

#set the plugin path to the directory containing Cassandra Sink Connector
Plug-in.path=/path/to/connectors

#Cassandra Sink Connector configuration for worker-3
Connector.class=io.confluent.connect.Cassandra.CassandraSinkConnector
tasks=1

#Cassandra connection details for worker -1
Cassandra.connection.host=cassandra-node-1
Cassandra.connection.port=9042
Cassandra.connection.username=my-username
Cassandra.connection.password=my-password
Cassandra.keyspace=my_keyspace
```

```
#Kafka topic to Cassandra table mapping for worker -3

topic.myworker_3-topic_1=Cassandra_table_4
topic.myworker_3-topic_2=Cassandra_table_5
topic.myworker_3-topic_3=Cassandra_table_6
```

Cassandra Settings(Common)

```
#Cassandra connection details
cassandra.contact.points=localhost #IP address
cassandra.port=9042
cassandra.username=your username
cassandra.password=your password
cassandra.keyspace=your keyspace
cassandra.retryPolicy=DefaultRetryPolicy
cassandra.consistencyLevel=QUORUM
```

Error Handling(common)

```
#Error Handling
Errors.tolerance=all
Errors.log.enable=true
Errors.log.include.messages=true
Errors.deadletterqueue.topic.name=dlq-topic
Errors.deadletterqueue.contet.headers.enable=true
Errors.deadletterqueue.topic.replicationfactor=3
Errors.retry.delay.mas.ms=60000
Errors.retry.timeout=0
```

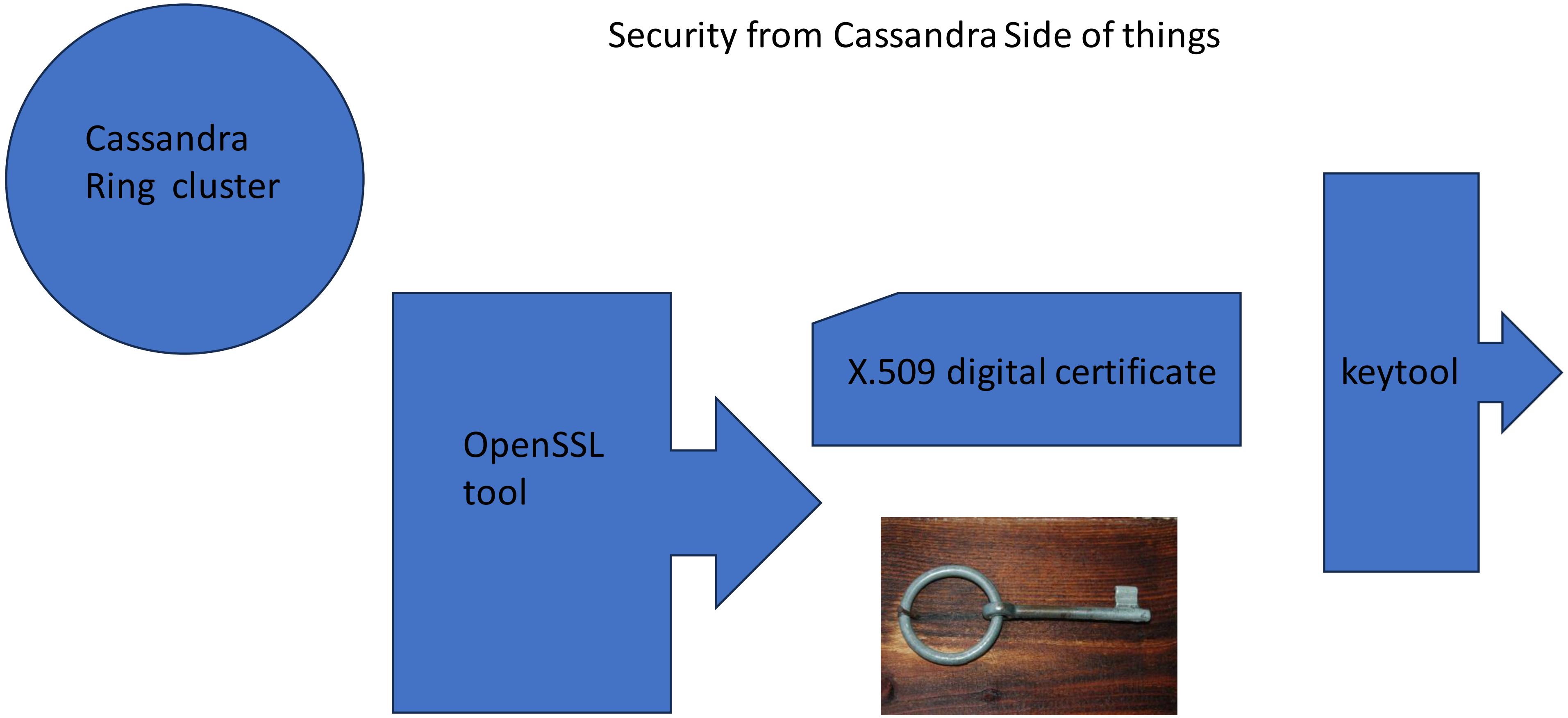
Internal topics Provisioning(Common)

```
#internal topics used by Kafka Connect
config.storage.topic=connect-configs
status.storage.topic=connect-status
offset.storage.topic=onnect-offsets
offset.storage.partitions=1
offset.storage.replication.factor=1
#Note:  all three properties files must share the same internal topic
# names – they must be the same name(s)
```

Security I (common – tbd)

```
#enable SSL/TLS
Security.protocol=SSL
#location of the truststore containing trusted CA certificates
Ssl.truststore.location=/path/to/truststore.jks
Ssl.truststore.password=truststore password
#if required
Ssl.keystore.location=/path/to/keystore.jks
Ssl.keystore.password=keystore password
Ssl.key.password= key password
```

Security from Cassandra Side of things

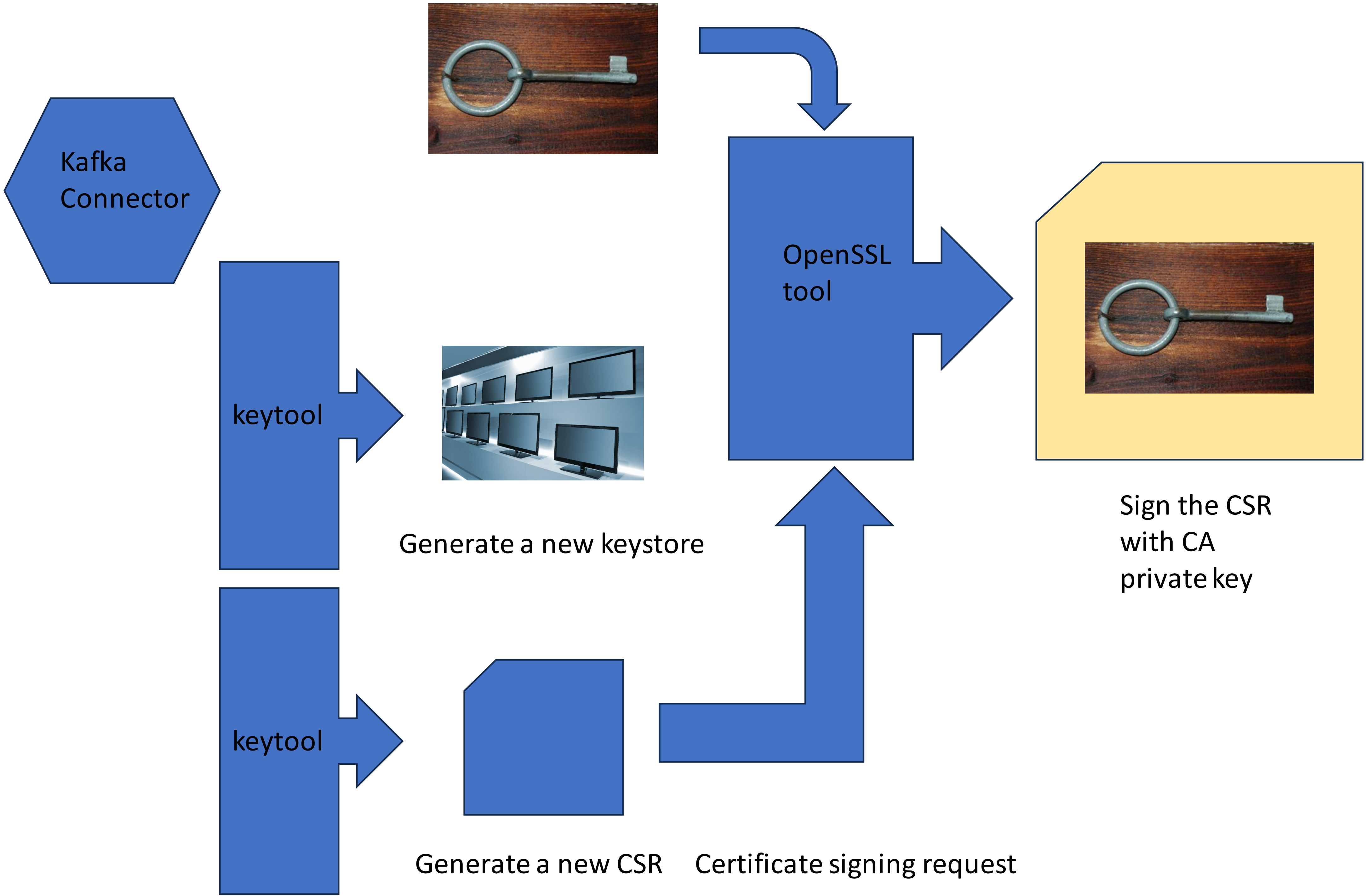


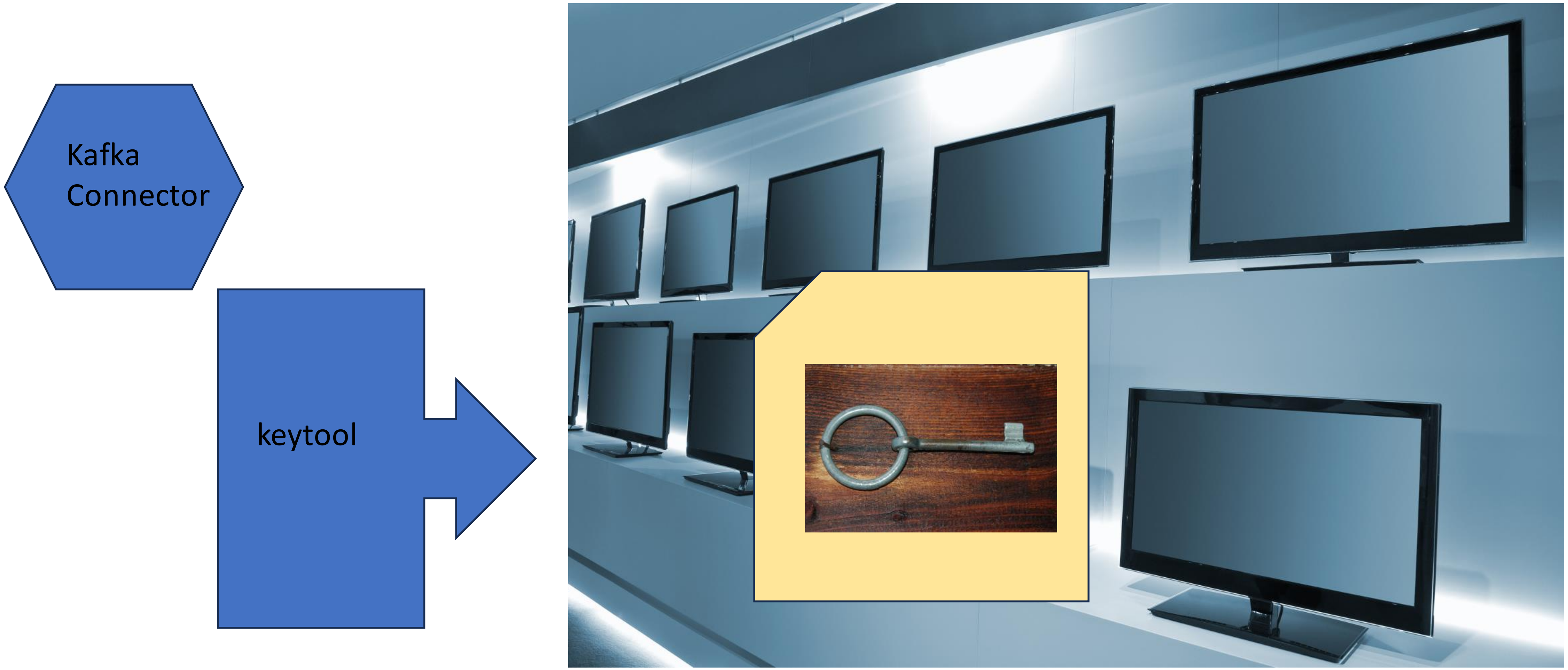
Import X.509 cert into Cassandra’s truststore .jks

Generate the digital certificate and the private key at the same time with OpenSSL tool

private key

Best practice:always secure the private key, do not transmit over the wire use KMS or a key vault system


Kafka Connector
keytool
Generate a new keystore
keytool
Generate a new CSR
Certificate signing request
OpenSSL tool
Sign the CSR with CA private key



Store signed CSR in Cassandra sink connector keystore

Strength of SSL/TLS: private key is never transmitted over the wire, it is secured

In the Kafka connect signed CSR with Cassandra's private key; On the Cassandra side of things it has the private key to decipher the signed CSR to match up and authenticate the sending party(Kafka connect side)

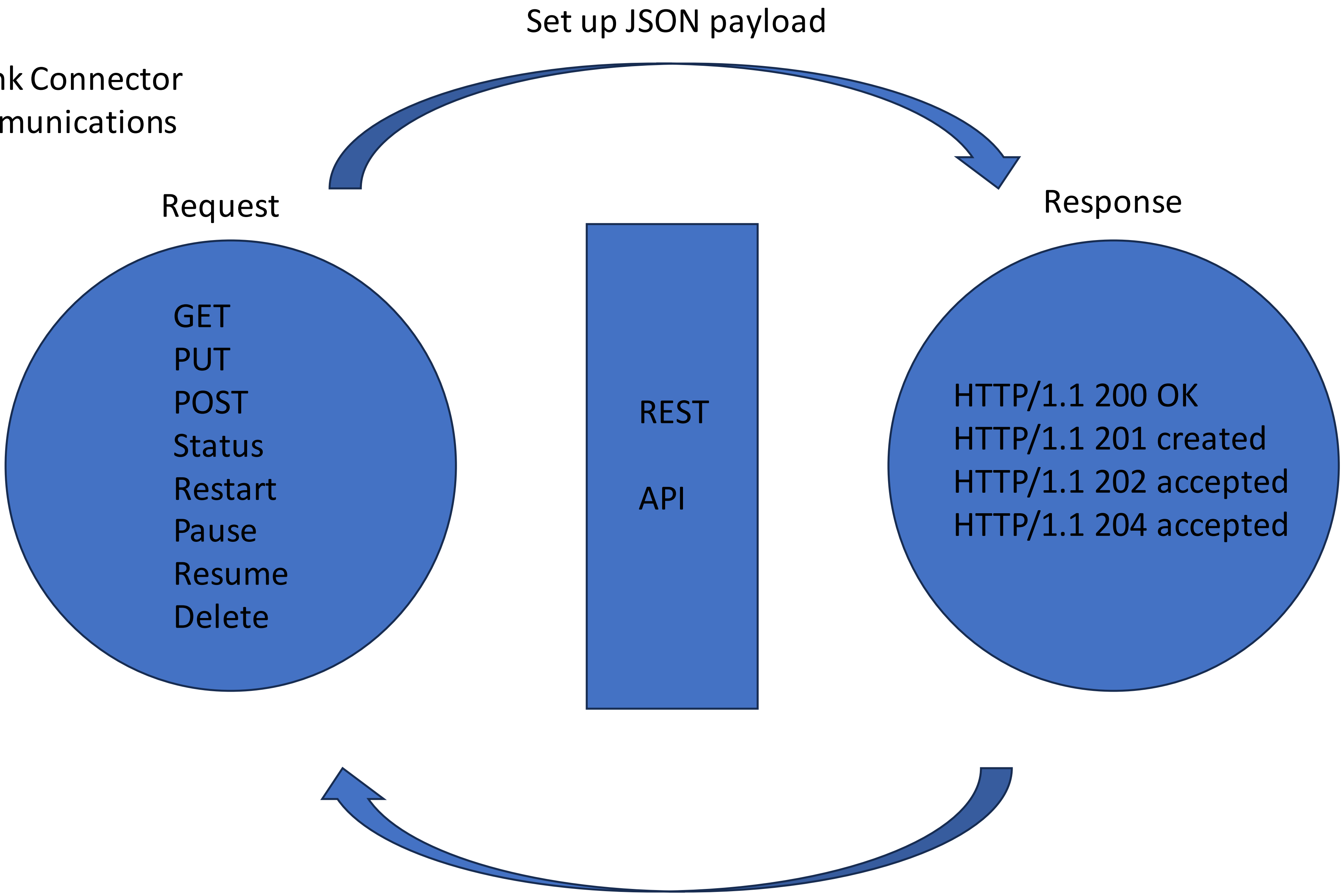
Security II (TBD) common

```
#enable SASL authentication

Security.protocol=SASL_PLAINTEXT
#SASL mechanism PLAIN, SCRAM-SHA-256,SCRAM-SHA-512
Sasl.mechanism=PLAIN
Sasl.jaas.config=org.apache.kafka.common.Security.plain.PlainLoginModule
#username and password
username=your username
Password=your password
p
```

REST API(Common)

```
#rest api for distributed  worker node


Rest.port=8083
Rest.advertised.host.name=localhost  #real IP address
Rest.advertised.port=8083
```

Cassandra Sink Connector
Bilateral communications

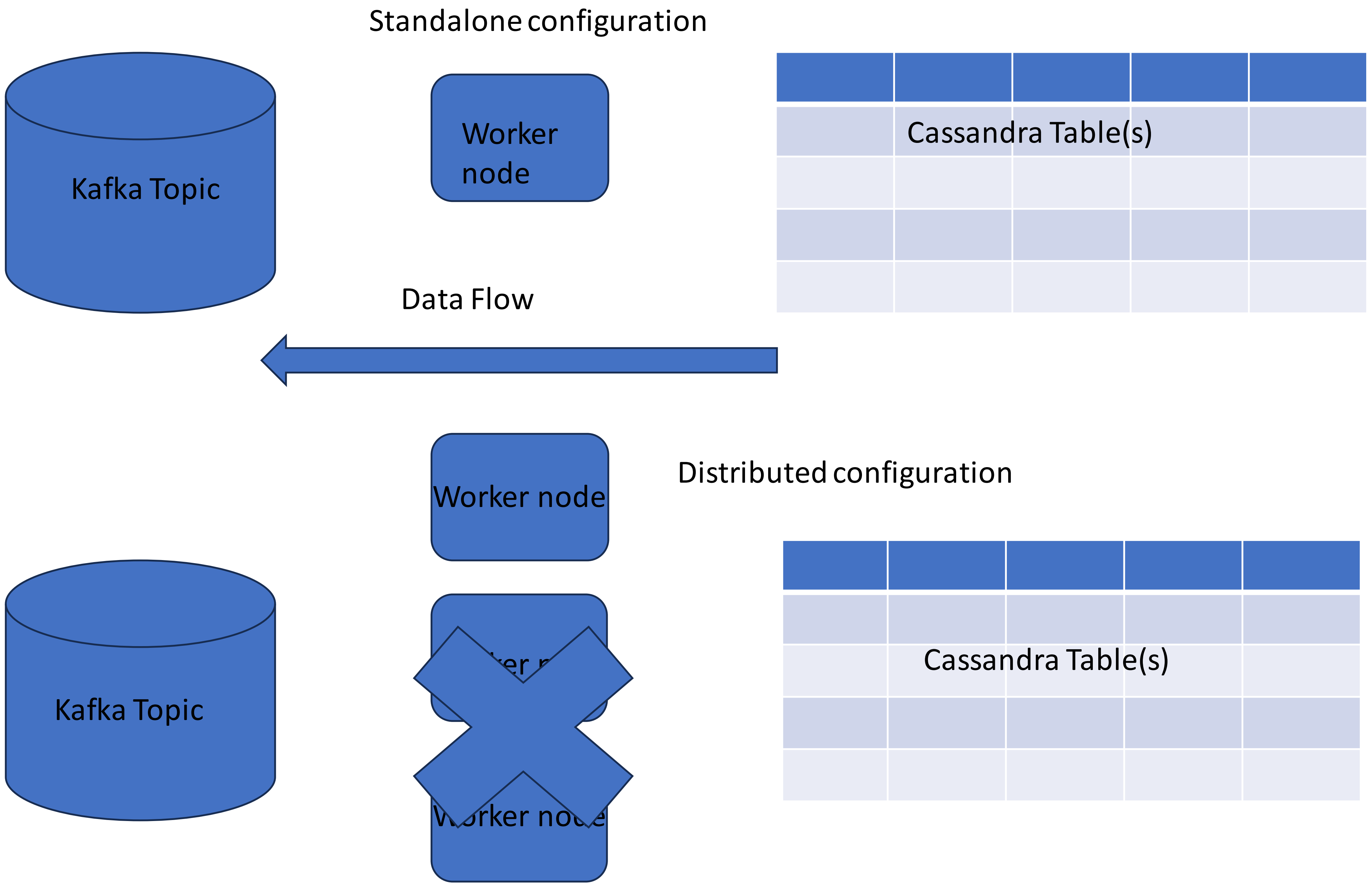


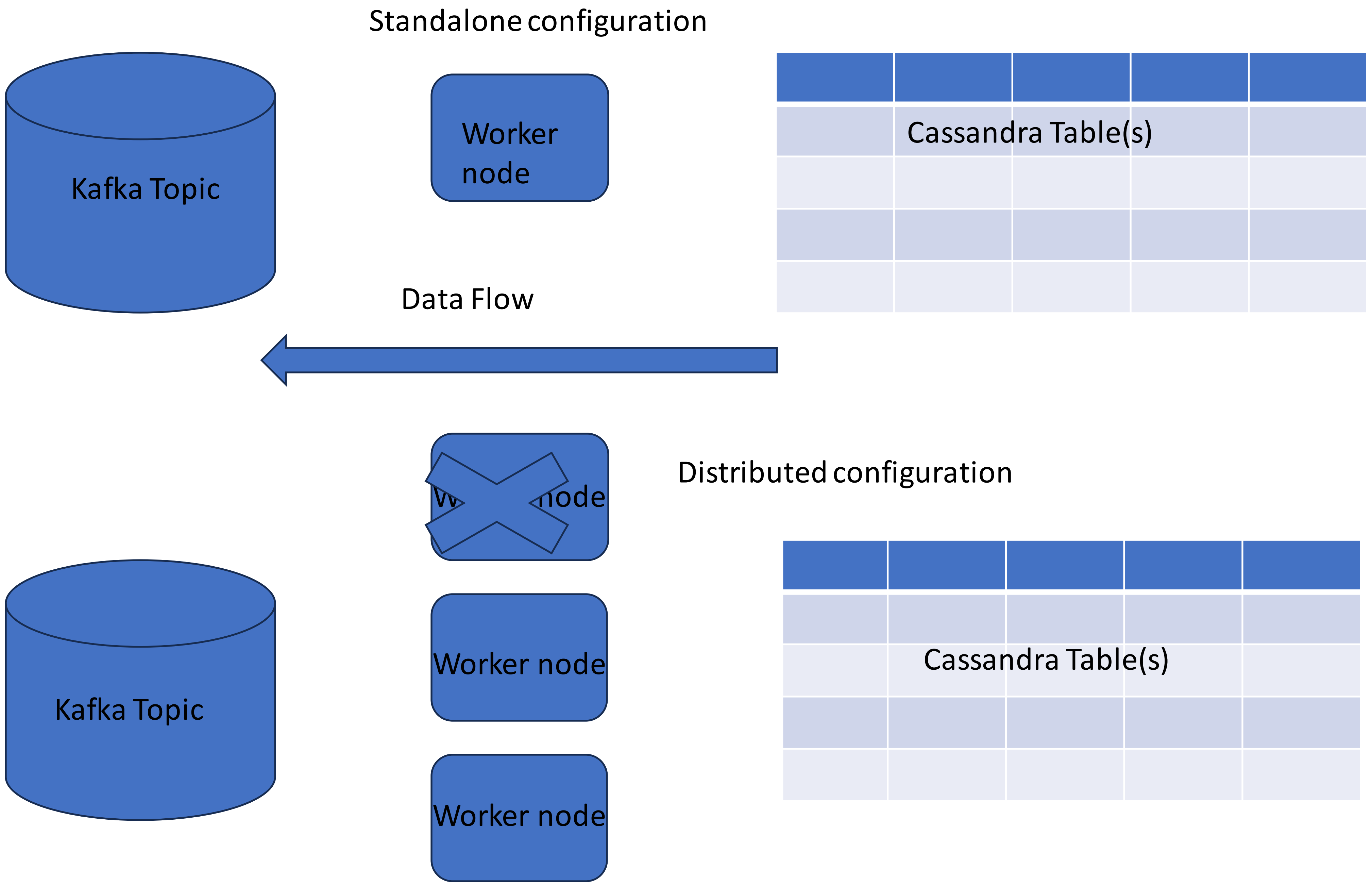
We set up the distributed node properties

- time to execute !!!
- how do we run Kafka Connect Cassandra sink connector(s) ???
- two ways get the connect-distributd.sh shell script from Apache directly or standalone connect-standalone.sh script
- get the shell scripts from Confluent distribution subscription /bin
- **./bin/connect-distributed.sh config/connect-distributed.properties**
- connection-distribution properties is the "quarterback of this Kafka Connect ecosystem

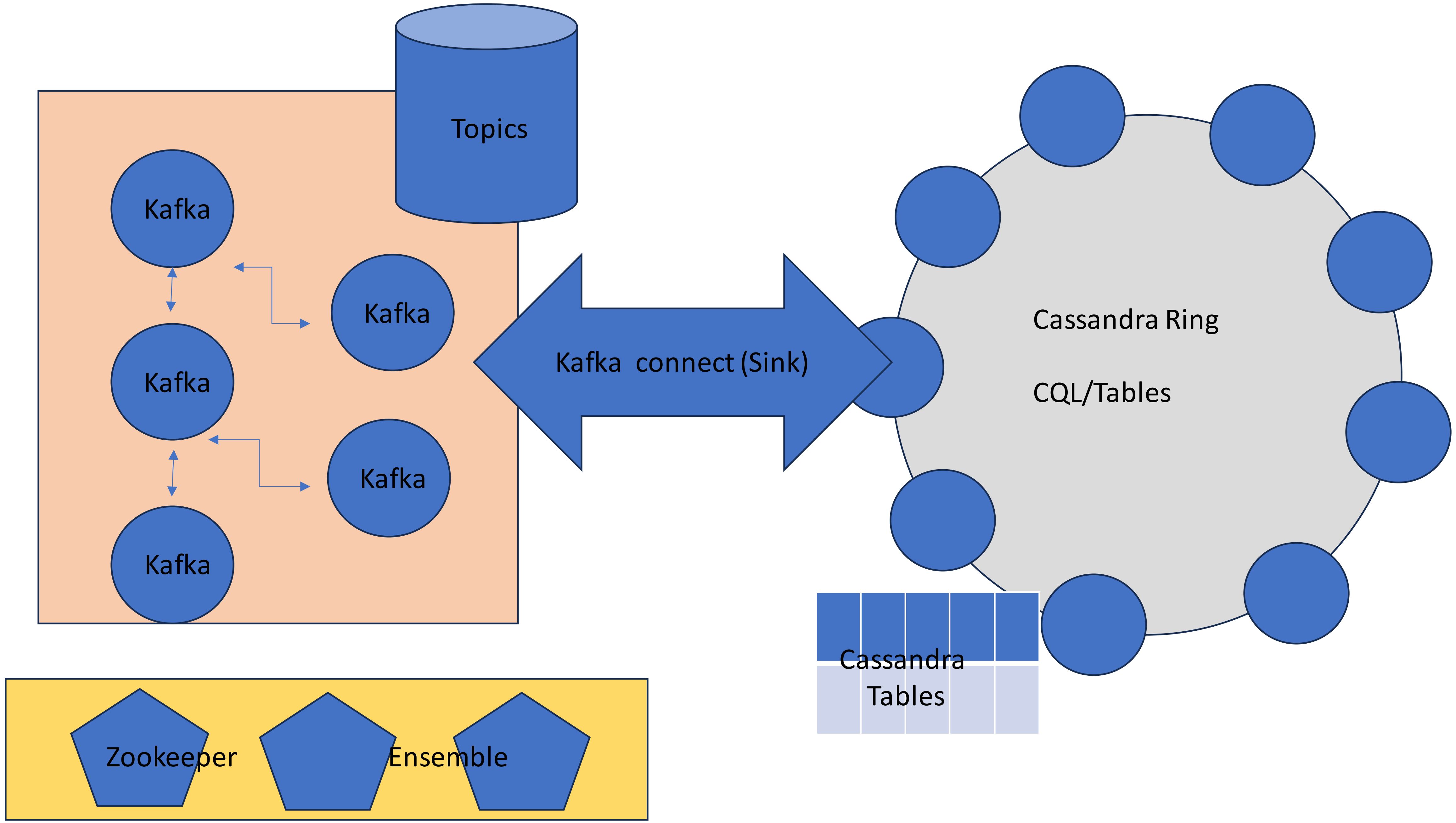
We have shown Cassandra as a Sink

- provision for 3 worker nodes running in distributed mode
- not much change in the various properties file
- just worry about the Kafka topics to Cassandra tables, that’s it
- what about Cassandra as a source
- what is in bold face is what needs to change for Cassandra to be the source


Standalone configuration
Kafka Topic
Worker node
Cassandra Table(s)
Data Flow
Worker node
Distributed configuration
Kafka Topic
Cassandra Table(s)


Standalone configuration
Kafka Topic
Worker node
Cassandra Table(s)
Data Flow
Worker node
Distributed configuration
Kafka Topic
Worker node
Worker node
Cassandra Table(s)

- Confluent has no official Cassandra Source connector available
- Check with github or open source for possible downloads
- BYO a lot of work; deep knowledge of Cassandra and Kafka internals to come to fruition
- KIP – Kafka Improvement Process it is like an RFP for the Kafka Developer Community
- For now, only the Cassandra Sink Connector is available


Topics
Kafka
Kafka
Kafka
Kafka
Kafka
Kafka connect (Sink)
Cassandra Ring
CQL/Tables
Cassandra Tables
Zookeeper
Ensemble

Thank you

The Brillio Team