KAFKA AND THE JCC LOGMINER LOADER KAFKA OPTION

Keith W. Hare

JCC Consulting, Inc.

December 21, 2018

Introduction

- Introduction to Kafka
  - What is Kafka?
  - Kafka Distributions and Versions
  - Kafka Topics
  - Kafka Consumers and Producers
- JCC LogMiner Loader Kafka Option
  - Replicating to Kafka
  - Kafka Option Prerequisites
  - Kafka Option Example
- Kafka Server – A Peak Inside the Black Box
  - Zookeeper
  - Kafka
  - Confluent Schema Registry
- Summary

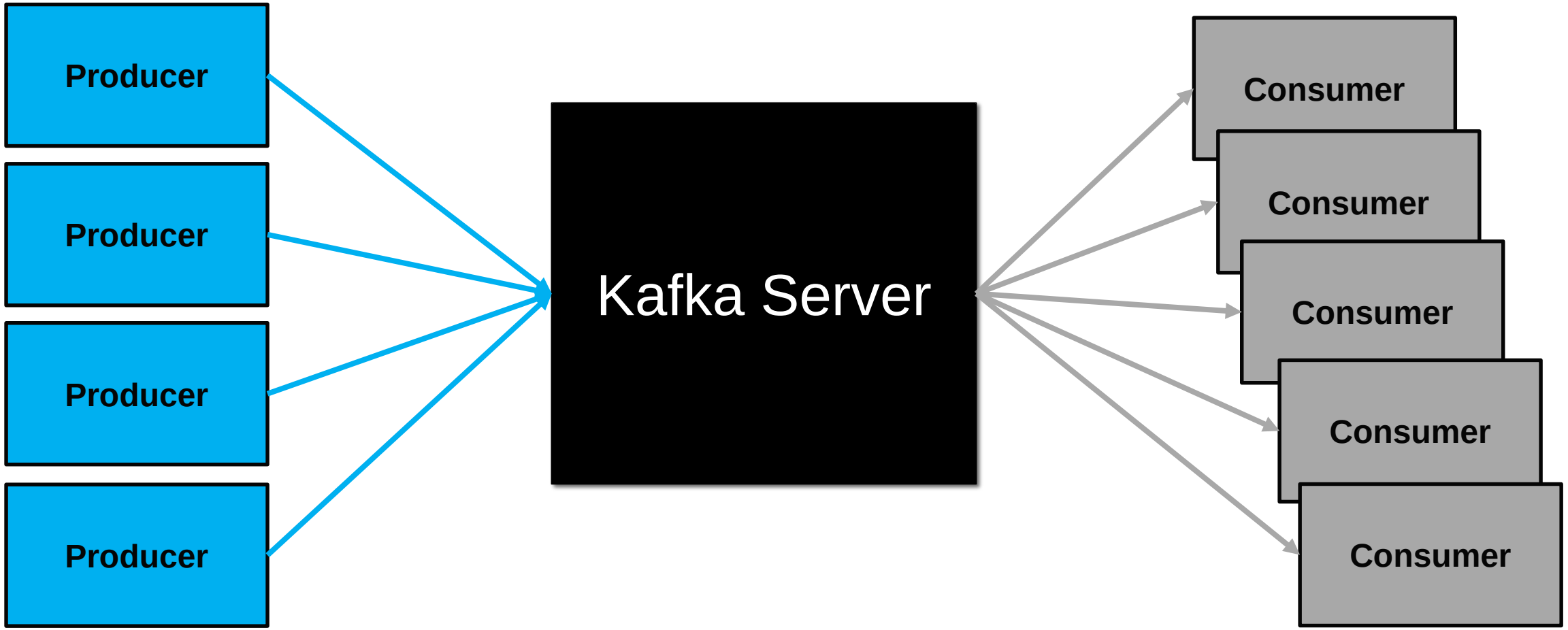
Introduction to Kafka

- Kafka is a Reliable Messaging Service
- Open Source
  - Apache Software Foundation
  - Apache 2 License
- Written in Java – requires Java 8
- Originally created by LinkedIn
- Installation kits for Linux and Windows
- Ongoing development, new releases quarterly
- Similar function as
  - MQ Series
  - DEC Reliable Transaction Router
  - Etc.

Kafka Pieces

- Kafka Server
  - Does the hard work
- Message Producers
  - Publish messages to Topics on a Kafka Server
  - Multiple producers can publish to the same topic
  - Each producer can publish to multiple topics
- Message Consumers
  - Read Messages from Topics on a Kafka Server
  - Multiple consumers can read from the same topic
  - Each consumer can read from multiple topics
  - If multiple consumers specify the same group, they will get different data

The Big Picture



Kafka Distributions

- Apache Kafka
  - Open Source
    - Apache License
    - Zookeeper
    - Kafka libraries
    - Producer & Consumer
- Confluent Kafka
  - Open Source
    - Apache License
    - Zookeeper
    - Kafka
    - Producer & Consumer libraries
    - Confluent Schema Registry
    - KSQL
  - Commercial
    - Supported
    - Additional Tools and monitoring

Kafka Versions

Apache Kafka	Confluent Platform	Confluent Kafka
1.0	4.0	1.0-cpl
1.1	4.1	1.1-cpl
1.1.1	4.1.2	1.1.1-cpl
2.0	5.0	2.0-cpl
2.1	5.1	2.1-cpl

Release Histories

Apache Kafka Releases

0.10.2.0 – February 2017
- 0.10.2.1 – April 2017
- 0.10.2.2 – July 2018

0.11.0.0 – June 2017
- 0.11.0.3 – July 2018

1.0.0 – October 2017
- 1.1.0 – February 2018
- 1.1.1 – July 2018

2.0.0 – June 2018
- 2.1.0 – November 2018

https://cwiki.apache.org/confluence/display/KAFKA/Future+release+plan

Confluent Platform Releases

1.0.0 – February 25, 2015

2.0.x – December 7, 2015

3.0.x – May 24, 2016
- 3.1.x – November 15, 2016
- 3.2.x – March 2, 2017
- 3.3.x – August 1, 2017

4.0.x – November 28, 2017
- 4.1.x – April 16, 2018

5.0.x – July 31, 2018
- 5.1.x – December 14, 2018

https://docs.confluent.io/current/installation/versions-interoperability.html

Kafka Web Sites

- Apache Kafka – https://kafka.apache.org
- Confluent Kafka – https://www.confluent.io

Kafka Clusters

- Zookeeper
  - Tracks Topics
  - Coordinates information across cluster nodes
- Kafka Cluster
  - Minimum of 3 nodes recommended
  - Nodes vote on cluster master
  - If node crashes, remaining nodes choose new master
  - Nodes can be different servers or different Docker Containers
    - What's a Docker Container? Exercise for the reader
- Topic Partitions
  - N partitions for parallel performance
- Topic Replicas
  - M copies of each partition for resilience
- Appropriate values for numbers of partitions and replicas
  - Highly dependent on workload
  - Beyond this scope of this presentation

Kafka Topics

- Named – producer and consumer need to agree on naming convention
- Producer publishes message to a topic
- Consumer reads message from a topic
- Can be partitioned
  - Messages within a partition are processed First In, First Out (FIFO)
  - Messages in separate partitions processed in parallel
- Partitions can be replicated – transparent to producer and consumer

Kafka Messages

- A Kafka Message consists of:
  - Key (Optional)
  - Actual Message
  - Timestamp added by Kafka
- Producer and Consumer need to agree on message format
- Kafka Server is agnostic about format
- Possible message formats:
  - Arbitrary byte string
  - XML
  - JSON
  - Avro with Schema Registration

Avro and Schema Registration

- Record definition (Schema) for each record type is registered with Schema Registry Service
- Registry Service tracks schema changes
- Consumer uses Registry Service to interpret message
- Schema changes can be handled automatically by consumer

Kafka Producer and Consumer Example

Kafka Producer Console

```
$ java -cp /confluent_kafka_path/*:. –
_$    kafka.tools.ConsoleProducer -
_$    --broker-list  confluent01.jcc.com:9093 -
_$    --topic Test.Topic -
_$    --producer.config KAFKA_AVRO.PROPERTIES
>this is a test message
>Second test message
>Another arbitrary message
>
```

Kafka Consumer Console

```
$ java -cp /confluent_kafka_path/*:. –
_$   kafka.tools.ConsoleConsumer -
_$  --bootstrap-server confluent01.jcc.com:9093 -
_$  --topic Test.Topic -
_$  --consumer.config KAFKA_AVRO.PROPERTIES
this is a test message
Second test message
Another arbitrary message
```

Kafka Commands – List Topics

```
$ java -cp /confluent_kafka_path/*:. kafka.admin.TopicCommand -
_$ --zookeeper confluent01.jcc.com:2181 -
_$ --list
KeithTest
Personnel.EMPLOYEES
Personnel.SALARY_HISTORY
Personnel.SRC_TBL_CMB
RegTest.EMPLOYEES
RegTest.SALARY_HISTORY
Test.Topic
TestTopic
__consumer_offsets
__transaction_state
_schemas
$
```

Kafka Commands – Describe Topic

```
$ java -cp /confluent_kafka_path/*:. kafka.admin.TopicCommand -
_$   --zookeeper confluent01.jcc.com:2181 -
_$   --topic Test.Topic -
_$   --describe
Topic:Test.Topic        PartitionCount:3        ReplicationFactor:1     Configs:
        Topic: Test.Topic       Partition: 0    Leader: 0       Replicas: 0     Isr: 0
        Topic: Test.Topic       Partition: 1    Leader: 0       Replicas: 0     Isr: 0
        Topic: Test.Topic       Partition: 2    Leader: 0       Replicas: 0     Isr: 0
$
```

Context for Previous Examples

```
$ @jcc_tool_com:jcc_lml_user 3.6
Setting JCC LogMiner Loader version 3.6
$ jcc_version
JCC Version T03.06.00 (built 17-DEC-2018 13:16:10.58)

$ show log confluent*
(LNM$PROCESS_TABLE)
(LNM$JOB_943DF8C0)
(LNM$GROUP_000030)
(JCC_CLML_03_06)

  "CONFLUENT_COMMON" = "JCC_TOOL_ROOT:[JAVA.LIB.KAFKA.CONFLUENT-4-1-1.CONFLUENT-COMMON]"
  "CONFLUENT_KAFKA" = "JCC_TOOL_ROOT:[JAVA.LIB.KAFKA.CONFLUENT-4-1-1.KAFKA]"
  "CONFLUENT_KAFKA_PATH" = "JCC_TOOL_ROOT:[JAVA.LIB.KAFKA.CONFLUENT-4-1-1.KAFKA-SERDE-TOOLS]"
        = "JCC_TOOL_ROOT:[JAVA.LIB.KAFKA.CONFLUENT-4-1-1.CONFLUENT-COMMON]"
        = "JCC_TOOL_ROOT:[JAVA.LIB.KAFKA.CONFLUENT-4-1-1.KAFKA]"
  "CONFLUENT_KAFKA_SERDE" = "JCC_TOOL_ROOT:[JAVA.LIB.KAFKA.CONFLUENT-4-1-1.KAFKA-SERDE-TOOLS]"
(LNM$SYSTEM_TABLE)
(LNM$SYSCLUSTER_TABLE)
(DECW$LOGICAL_NAMES)
$ define mfp_ssl JCC_ROOT:[KEITH.SQL_CLASS.LML_KAFKA.ssl]
$ set proc/parse_style=extended
```

Kafka and SSL

- Kafka Server can be configured to support Plaintext and/or SSL
- Plaintext – usually port 9092
  - Network packets are human readable
- SSL – usually port 9093
  - Network packets are encrypted
  - Can be used to authenticate
    - Client to Server
    - Server to Client
  - Can be used for both Producer and Consumer
  - Requires certificates, etc.
    - Can have a certificate per client

Kafka and Access Control Lists

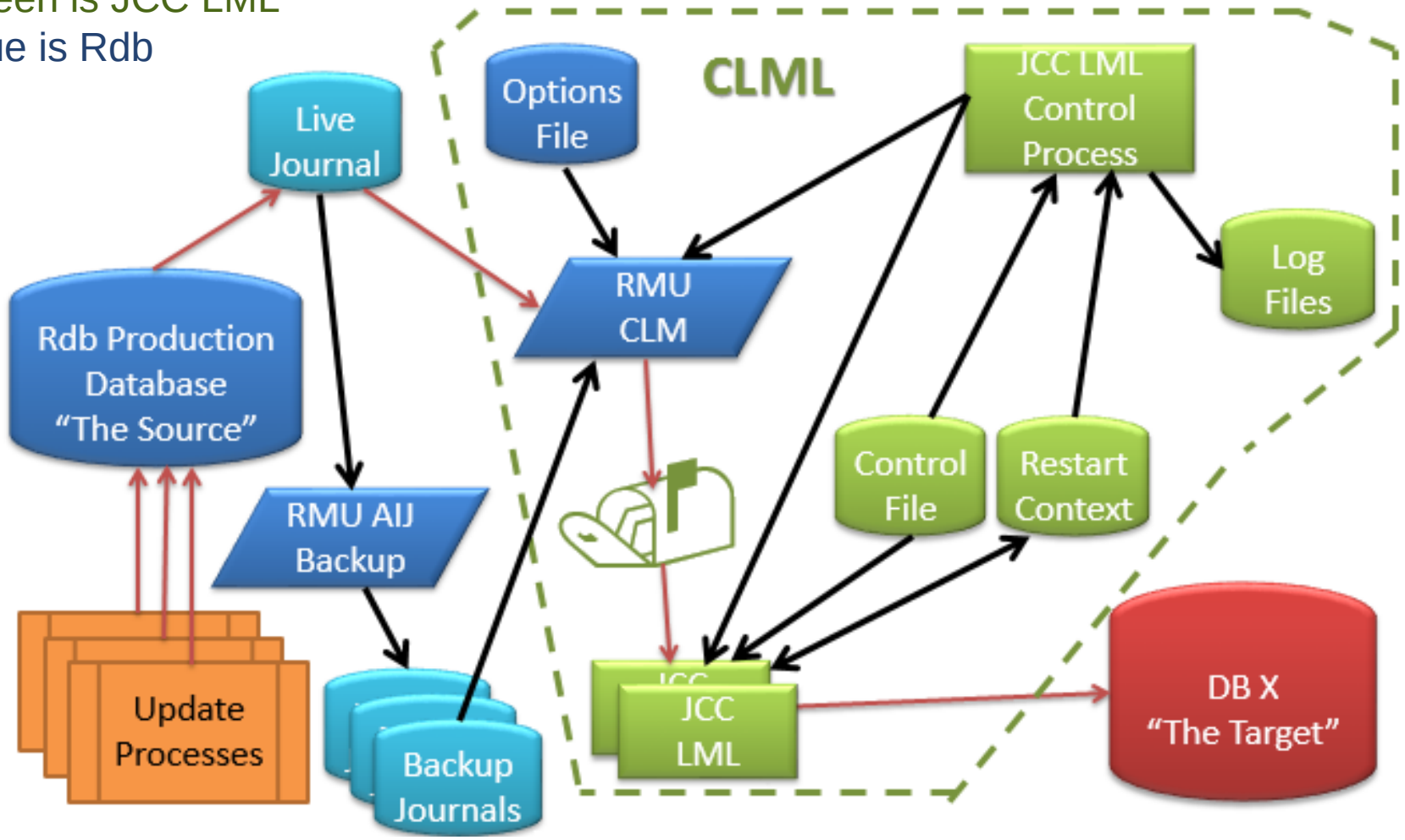
- Possible to define Access Control Lists (ACLs) on Topics
- We have not yet looked at Kafka ACLs

JCC LogMiner Loader Kafka Option

- Separately Licensed Option
  - Requires a new license key
  - Contact JCC Consulting for a quote
- How Kafka fits with other targets
- Prerequisites
  - Java 8
  - Kafka Java Libraries on OpenVMS
- Kafka Configuration Examples

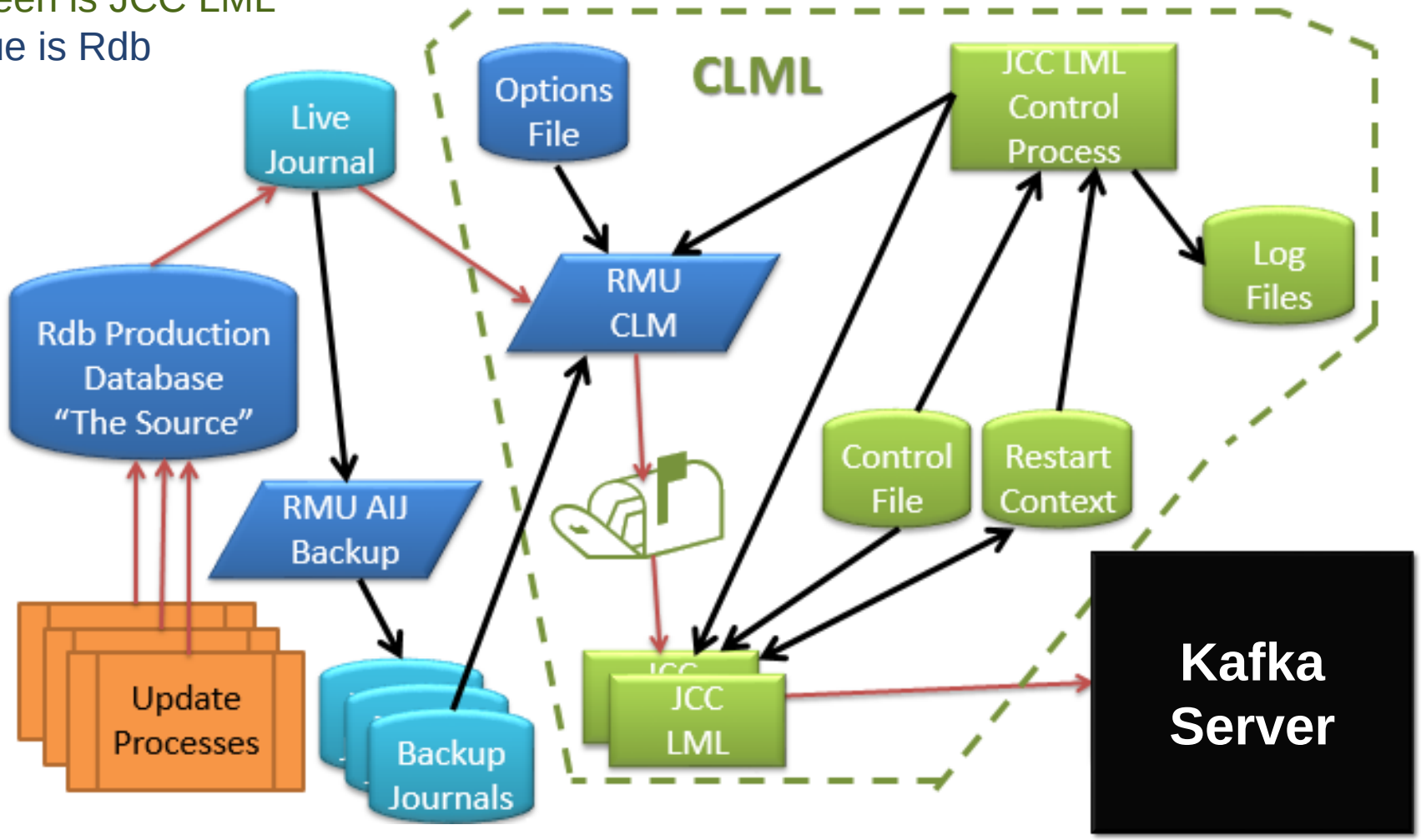
Replicating to a Database Target

Green is JCC LML
Blue is Rdb



Replicating to Kafka

Green is JCC LML
Blue is Rdb



Replicating to Kafka

- JCC LogMiner Loader Kafka Option publishes committed transaction to a Kafka Server
- Message Formats
  - XML
  - JSON
  - Avro
- Some new configuration options specific to Kafka
- What about a Kafka Consumer?
  - JCC LogMiner Loader Kafka Option does not provide a Kafka consumer
  - Can provide consulting services on designing a Kafka consumer and data lake

Prerequisite: Java 8

- JCC LogMiner Loader Kafka Option uses Kafka Libraries
- Kafka libraries require Java 8
- Java 8 is available only on OpenVMS IPF
- Can be installed version or as a separate JRE

Prerequisite: Kafka Java Libraries on OpenVMS

- Extract Kafka Libraries from a Kafka Server
- Confluent Open Source Kafka installation, Linux server
  - Required libraries are in the following folders:
    - /usr/share/java/confluent-common
    - /usr/share/java/kafka
    - /usr/share/java/kafka-serde-tools
- FTP files to OpenVMS Server
  - JAR files are Binary
  - License files are Text
- On OpenVMS, reset file characteristics

```
$ set file/attr=(rfm:stmlf,rat:cr,lrl:0,mr:0) -
_$ DISK$STATIC:[kafka.CONFLUENT-4-1-1...]*.jar
```

Kafka Library Location on OpenVMS

- Best location depends on your environment
- Multiple JCC LogMiner Loader versions
  - Separate Directory tree
- Relatively static environment
  - Kafka libraries in the directory tree jcc_tool_root:[java.lib...]
  - Example:

```
$ dire/size=all/date=create/prot jcc_tool_root:[java.lib.kafka.CONFLUENT-4-1-1]
Directory JCC_TOOL_ROOT:[JAVA.LIB.kafka.CONFLUENT-4-1-1]

confluent-common.DIR;1        1/16          15-NOV-2018 10:05:26.42  (RWED,RWED,RWE,E)
doc.DIR;1                     1/16          15-NOV-2018 10:05:27.27  (RWED,RWED,RWE,E)
kafka-serde-tools.DIR;1       2/16          15-NOV-2018 10:05:28.07  (RWED,RWED,RWE,E)
kafka.DIR;1                   8/16          15-NOV-2018 10:05:28.96  (RWED,RWED,RWE,E)

Total of 4 files, 12/64 blocks.
$
```

Kafka Example – Avro

```
!
! Define the Kafka Avro Specific stuff
!
output~kafka~synch~connect~record~Avro
kafka~connect~cnflnt4-04:9093,cnflnt4-03:9093,cnflnt4-02:9093
kafka~avro~SchemaRegistry~http://cnflnt4-02:8081
kafka~avro~namespace~MFP.avro
!
!  routine to expand wildcards does not work on unix-style names.
!
kafka~classpath~disk$static:[kafka.CONFLUENT-4-1-1.kafka-serde-tools]*.jar
kafka~classpath~disk$static:[kafka.CONFLUENT-4-1-1.confluent-common]*.jar
kafka~classpath~disk$static:[kafka.CONFLUENT-4-1-1.kafka]*.jar
!
kafka~Topic~'Personnel.',Table_Name
kafka~Model~ExactlyOnce
!
! External Properties File
!
java~properties~/control_files/kafka_avro.properties
!
! The remainder of this file is standard Loader "stuff"
!
```

Kafka Example – Output

- Output target is Kafka with a record format of Avro
  `output~kafka~synch~connect~record~Avro`
- List of Kafka nodes and ports
  `kafka~connect~cnflnt4-04:9093,cnflnt4-03:9093,cnflnt4-02:9093`
  - Three nodes in this Kafka cluster – supports failover
  - Port 9093 is SSL port
  - Kafka nodes and port numbers can be configured in Kafka server

Kafka Example – Schema Registry

- Schema Registry Service for Avro

  `kafka~avro~SchemaRegistry~http://cnflnt4-02:8081`

  - Registry traffic is not encrypted – HTTP
  - Port number is 8081
  - HTTP or HTTPS and port number can be configured in Schema Registry service
- Namespace used for Schema Registry

  `kafka~avro~namespace~MFP.avro`

  - Namespace is an arbitrary string
  - Namespace should be unique across publishers
  - Negotiate with Kafka consumer team

Kafka Example – Classpath

- Classpath for Kafka Java Libraries

```
kafka~classpath~disk$static:[kafka.CONFLUENT-4-1-1.kafka-serde-tools]*.jar
kafka~classpath~disk$static:[kafka.CONFLUENT-4-1-1.confluent-common]*.jar
kafka~classpath~disk$static:[kafka.CONFLUENT-4-1-1.kafka]*.jar
```

- VMS directory syntax and wildcard are required
- Directory is location where Confluent Kafka libraries installed

Kafka Example – Topic and Model

- Topic Naming

  `kafka~Topic~'Personnel.',Table_Name`

  - Kafka Topics will be Personnel.EMPLOYEES, Personnel.SALARY_HISTORY, etc.
- Kafka Model

  `kafka~Model~ExactlyOnce`

  - JCC LogMiner Loader Kafka Option supports:
    - Transaction
    - ExactlyOnce – Only one copy of a published messages will exist in Kafka Server

Kafka Example – External Properties File

- Properties in Properties file passed to Java Engine
  `java~properties~/control_files/kafka_avro.properties`
- Control_files is an OpenVMS logical name
  `$ define control_files JCC_ROOT:[KEITH.SQL_CLASS.LML_KAFKA]`
- kafka_avro.properties contains:

```
#
# kafka_avro.properties
#
security.protocol=ssl
ssl.truststore.location=/mfp_ssl/kafka_truststore.jks
ssl.truststore.password=<password>
ssl.keystore.location=/mfp_ssl/kafka_keystore.jks
ssl.keystore.password=<password>
ssl.key.password=<password>
```

- Mfp_ssl is an OpenVMS logical name
  `$ define mfp_ssl JCC_ROOT:[KEITH.SQL_CLASS.LML_KAFKA.ssl]`

Kafka Example – Keystore and Truststore

- Properties File references Mfp_ssl which points to JCC_ROOT:[KEITH.SQL_CLASS.LML_KAFKA.ssl]

```
$ dire/noheader/size=all/prot JCC_ROOT:[KEITH.SQL_CLASS.LML_KAFKA.SSL]
JCC_ROOT:[KEITH.SQL_CLASS.LML_KAFKA.ssl]ca-key.;1
                                4/5             (RWED,RWED,RE,)
JCC_ROOT:[KEITH.SQL_CLASS.LML_KAFKA.ssl]kafka_keystore.jks;1
                                9/10            (RWED,RWED,RE,)
JCC_ROOT:[KEITH.SQL_CLASS.LML_KAFKA.ssl]kafka_truststore.jks;1
                                3/5             (RWED,RWED,RE,)
Total of 3 files, 16/20 blocks.
$
```

- These files were obtained from manager of the Kafka servers

Kafka Example – Rest of the Configuration File

- Remainder of the JCC LogMiner Loader configuration file identical to any other target
  - Table
  - MapTable
  - Etc.

List of Topics

```
$ java -cp /confluent_kafka_path/*:. kafka.admin.TopicCommand -
_$ --zookeeper cnflnt4-02.jcc.com:2181 -
_$ --list
Personnel.CANDIDATES
...
Personnel.EMPLOYEES
...
Personnel.SALARY_HISTORY
...
RegTest.DETAILS
RegTest.DETAILS_AUDIT_D
RegTest.DETAILS_AUDIT_M
RegTest.PEOPLE
__confluent.support.metrics
__consumer_offsets
__transaction_state
_schemas
$
```

Topic Personnel.EMPLOYEES

```
$ java -cp /confluent_kafka_path/*:. -
_$   kafka.admin.TopicCommand -
_$   --zookeeper cnflnt4-02.jcc.com:2181 -
_$   --topic Personnel.EMPLOYEES -
_$   --describe
Topic:Personnel.EMPLOYEES       PartitionCount:6        ReplicationFactor:2     Configs:
        Topic: Personnel.EMPLOYEES      Partition: 0    Leader: 1       Replicas: 1,2   Isr: 1,2
        Topic: Personnel.EMPLOYEES      Partition: 1    Leader: 2       Replicas: 2,3   Isr: 2,3
        Topic: Personnel.EMPLOYEES      Partition: 2    Leader: 3       Replicas: 3,1   Isr: 1,3
        Topic: Personnel.EMPLOYEES      Partition: 3    Leader: 1       Replicas: 1,3   Isr: 1,3
        Topic: Personnel.EMPLOYEES      Partition: 4    Leader: 2       Replicas: 2,1   Isr: 1,2
        Topic: Personnel.EMPLOYEES      Partition: 5    Leader: 3       Replicas: 3,2   Isr: 2,3
$
```

SQL Update

- With a JCC LogMiner Loader session started, a source row is updated:

```
SQL> update employees set last_name = last_name where employee_id = '00165';
1 row updated
SQL> commit;
```

Consuming an Avro Message

```
$ java -cp /confluent_kafka_path/*:. kafka.tools.ConsoleConsumer -
_$    --bootstrap-server cnflnt4-04.jcc.com:9093 -
_$    --formatter io.confluent.kafka.formatter.AvroMessageFormatter -
_$    --property schema.registry.url="http://cnflnt4-04.jcc.com:8081" -
_$    --topic Personnel.EMPLOYEES -
_$    --consumer.config KAFKA_AVRO.PROPERTIES
{"EMPLOYEE_ID":"00165","LAST_NAME":{"string":"Smith
"},"FIRST_NAME":null,"MIDDLE_INITIAL":{"string":"D"},"ADDRESS_DATA_1":{"
string":"120 Tenby Dr.                "},"ADDRESS_DATA_2":{"string":"
"},"CITY":{"string":"Chocorua
"},"STATE":{"string":"NH"},"POSTAL_CODE":{"string":"03817"},"SEX":{"string
":"M"},"BIRTHDAY":{"long":-
493344000000000},"STATUS_CODE":{"string":"2"},"CDC_FLAG":{"string":"M"},"L
ML_COMMIT_TIME":{"long":1545323810282604},"JCCLML_LSN":{"long":82},"TRANSA
CTION_COMMIT_TIME":{"long":1545323810282604},"JCCLML_READ_TIME":{"long":15
45323810480604},"TRANSMISSION_DATE_TIME":{"long":1545323810483604},"LOADER
_SEQUENCE_NUMBER":{"long":82},"ORIGINATING_DBKEY":{"long":22236523163025409},"CDC_TS":{"string":"2018-12-20 16:36:50.28"}}
```

Formatted Avro Message – JSON


```
{
  "EMPLOYEE_ID": "00165",
  "LAST_NAME": "Smith         ",
  "FIRST_NAME": null,
  "MIDDLE_INITIAL": "D",
  "ADDRESS_DATA_1": "120 Tenby Dr.            ",
  "ADDRESS_DATA_2": "                      ",
  "CITY": "Chocorua            ",
  "STATE": "NH",
  "POSTAL_CODE": "03817",
  "SEX": "M",
  "BIRTHDAY": -493344000000000,
  "STATUS_CODE": "2",
  "CDC_FLAG": "M",
  "LML_COMMIT_TIME": 1545324105883310,
  "JCCLML_LSN": 84,
  "TRANSACTION_COMMIT_TIME": 1545324105883310,
  "JCCLML_READ_TIME": 1545324106254298,
  "TRANSMISSION_DATE_TIME": 1545324106257297,
  "LOADER_SEQUENCE_NUMBER": 84,
  "ORIGINATING_DBKEY": 22236523163025409,
  "CDC_TS": "2018-12-20 16:41:45.88"
}
```

Java Console is not sufficient

- Need application to consume Kafka messages and do something with them
- JCC LogMiner Loader Regression Test
  - Consumer Messages
  - Parse them into columns
  - Write them to a target database
  - Supports comparing source and target data
- Real application
  - Many open source tools available to consume Kafka messages
  - Write data to a data lake?

Kafka Option and Partitions

- JCC LogMiner Loader Kafka Option is aware of partitions
- Distributes data across partitions based on hash of DBKey
- All Messages for a particular DBKey will be written to the same partition
  - Updates of the same row in consecutive transactions will be processed in the correct order
- Rows from the same source transaction might be written to different partitions

JCC LogMiner Loader Failure and Restart

- Some failure scenarios where JCC LogMiner Loader might re-publish a small number of source transactions
- JCC LogMiner Loader uses Kafka Transactions
  - Short window between Kafka commit and recording restart point in checkpoint file
- Multiple JCC LogMiner Loader Threads
  - Restart is from the oldest checkpoint across the threads
  - Updates from other threads resent
- Add Virtual Column Loader Sequence Number (LSN) so downstream processing can identify of message has already been seen

JCC LogMiner Loader Kafka Option Summary

- Most of configuration identical to other targets
- Operation identical to other targets

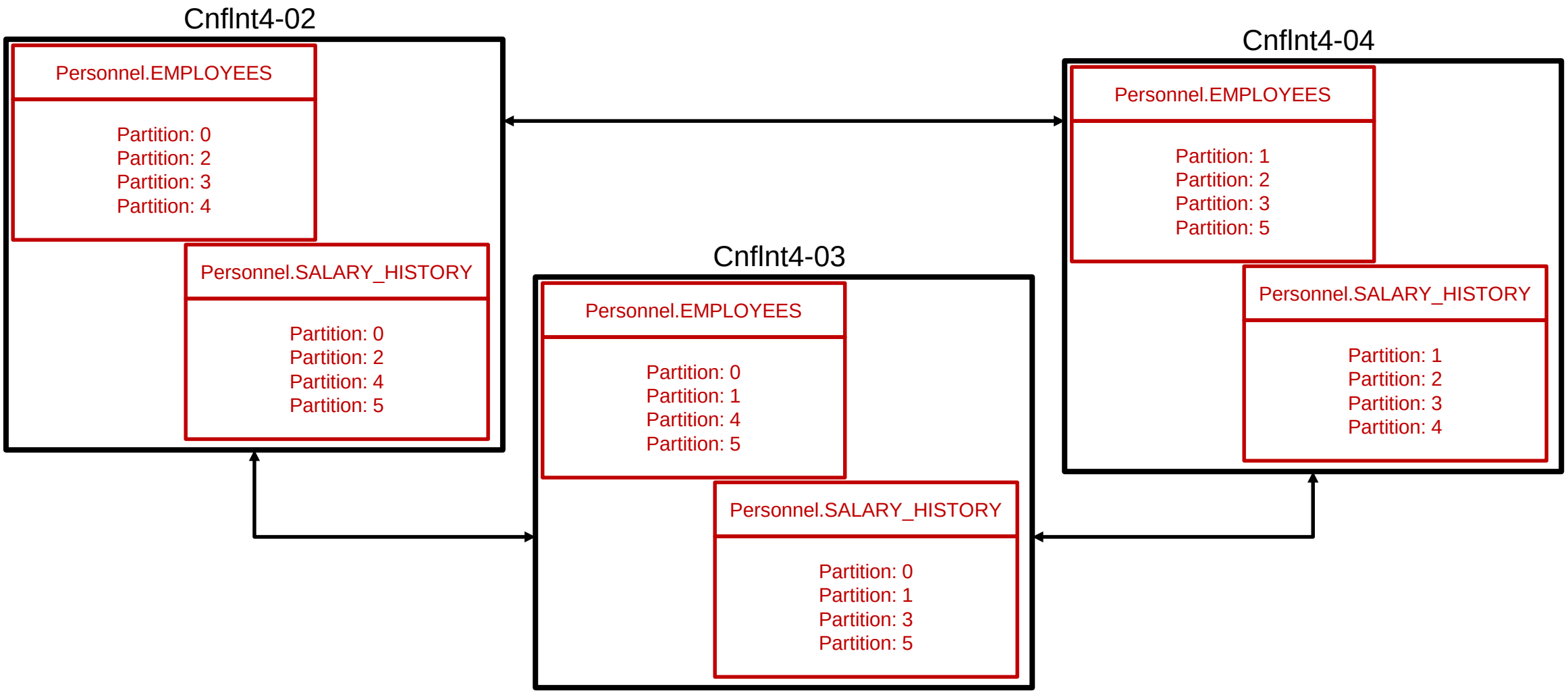
Kafka Server – A Peak Inside The Black Box

- Test Environment for JCC LogMiner Loader Kafka Option
  - Centos V7.x
  - Virtual Machines
  - Several different Kafka versions
    - Kafka V1.x single node
    - Confluent V4.x three-node cluster
    - Confluent V5.x single node
- We will look at the three-node cluster in more detail

JCC Test Environment – Confluent V4.x Cluster

- Three mostly identical nodes:
  - cnflnt4-02
  - cnflnt4-03
  - cnflnt4-04
- Each node runs three services:
  - Zookeeper
  - Kafka
  - Schema Registry

Three Node Kafka Cluster



Zookeeper Configuration

- /etc/kafka/zookeeper.properties

```
# the directory where the snapshot is stored.
dataDir=/var/lib/zookeeper
# the port at which the clients will connect
clientPort=2181
# disable the per-ip limit on the number of connections since this is a non-
production config
maxClientCnxns=0
#
# Zookeeper Ensemble (cluster)
# 2018-08-23 KWH
#
initLimit=5
syncLimit=2
server.1=cnflnt4-02.jcc.com:2888:3888
server.2=cnflnt4-03.jcc.com:2888:3888
server.3=cnflnt4-04.jcc.com:2888:3888
```

Zookeeper Service

```
[keith@cnflnt4-02 ~]$ sudo systemctl status confluent-zookeeper
? confluent-zookeeper.service - Apache Kafka - ZooKeeper
   Loaded: loaded (/usr/lib/systemd/system/confluent-zookeeper.service; enabled; vendor preset: disabled)
   Active: active (running) since Mon 2018-10-15 10:13:15 EDT; 2 months 5 days ago
     Docs: http://docs.confluent.io/
 Main PID: 1112 (java)
    Tasks: 31
   CGroup: /system.slice/confluent-zookeeper.service
           mq1112 java -Xmx512M -Xms512M -server -XX:+UseG1GC -XX:MaxGCPauseMillis=20 -XX:InitiatingHeapOccupancyPercent=35 -XX:+...

Dec 17 11:57:14 cnflnt4-02.jcc.com zookeeper-server-start[1112]: [2018-12-17 11:57:14,148] INFO Accepted socket connection fr...ory)
Dec 17 11:57:14 cnflnt4-02.jcc.com zookeeper-server-start[1112]: [2018-12-17 11:57:14,151] WARN Connection request from old c...ver)
Dec 17 11:57:14 cnflnt4-02.jcc.com zookeeper-server-start[1112]: [2018-12-17 11:57:14,151] INFO Client attempting to establis...ver)
Dec 17 11:57:14 cnflnt4-02.jcc.com zookeeper-server-start[1112]: [2018-12-17 11:57:14,156] INFO Established session 0x166781d...ver)
Dec 17 11:57:14 cnflnt4-02.jcc.com zookeeper-server-start[1112]: [2018-12-17 11:57:14,169] INFO Closed socket connection for ...nxn)
Dec 17 17:56:05 cnflnt4-02.jcc.com zookeeper-server-start[1112]: [2018-12-17 17:56:05,581] INFO Accepted socket connection fr...ory)
Dec 17 17:56:05 cnflnt4-02.jcc.com zookeeper-server-start[1112]: [2018-12-17 17:56:05,904] WARN Connection request from old c...ver)
Dec 17 17:56:05 cnflnt4-02.jcc.com zookeeper-server-start[1112]: [2018-12-17 17:56:05,904] INFO Client attempting to establis...ver)
Dec 17 17:56:05 cnflnt4-02.jcc.com zookeeper-server-start[1112]: [2018-12-17 17:56:05,909] INFO Established session 0x166781d...ver)
Dec 17 17:56:05 cnflnt4-02.jcc.com zookeeper-server-start[1112]: [2018-12-17 17:56:05,929] INFO Closed socket connection for ...nxn)
Hint: Some lines were ellipsized, use -l to show in full.
[keith@cnflnt4-02 ~]$
```

Kafka Configuration – /etc/kafka/server.properties

```
############################## Server Basics ##############################

# The id of the broker. This must be set to a unique integer for each broker.
broker.id=1
#broker.id.generation.enable=true

### SSL Settings ###
ssl.truststore.location=/var/private/ssl/kafka.truststore.jks
ssl.truststore.password=<password>
ssl.keystore.location=/var/private/ssl/kafka.keystore.jks
ssl.keystore.password=<password>
ssl.key.password=<password>

security.inter.broker.protocol=SSL
ssl.client.auth=required
```

Kafka Configuration Continued

```
############################# Socket Server Settings #############################

# The address the socket server listens on. It will get the value returned from
# java.net.InetAddress.getCanonicalHostName() if not configured.
#   FORMAT:
#     listeners = listener_name://host_name:port
#   EXAMPLE:
#     listeners = PLAINTEXT://your.host.name:9092
#listeners=PLAINTEXT://:9092
listeners=SSL://cnflnt4-02.jcc.com:9093,PLAINTEXT://cnflnt4-02.jcc.com:9092

# Hostname and port the broker will advertise to producers and consumers. If not set,
# it uses the value for "listeners" if configured.  Otherwise, it will use the value
# returned from java.net.InetAddress.getCanonicalHostName().
#advertised.listeners=PLAINTEXT://your.host.name:9092
advertised.listeners=SSL://cnflnt4-02.jcc.com:9093,PLAINTEXT://cnflnt4-02.jcc.com:9092
```

Kafka Configuration Continued

```
############################## Zookeeper ##############################

# Zookeeper connection string (see zookeeper docs for details).
# This is a comma separated host:port pairs, each corresponding to a zk
# server. e.g. "127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002".
# You can also append an optional chroot string to the urls to specify the
# root directory for all kafka znodes.
#zookeeper.connect=localhost:2181
zookeeper.connect=cnflnt4-02.jcc.com:2181,cnflnt4-03.jcc.com:2181,cnflnt4-04.jcc.com:2181
```

Kafka Service

```
[keith@cnflnt4-02 ~]$ sudo systemctl status confluent-kafka
? confluent-kafka.service - Apache Kafka - broker
   Loaded: loaded (/usr/lib/systemd/system/confluent-kafka.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2018-12-16 19:30:56 EST; 3 days ago
     Docs: http://docs.confluent.io/
 Main PID: 16232 (java)
    Tasks: 69
   CGroup: /system.slice/confluent-kafka.service
           mq16232 java -Xmx1G -Xms1G -server -XX:+UseG1GC -XX:MaxGCPauseMillis=20 -XX:InitiatingHeapOccupancyPercent=35 -XX:+Exp...

Dec 20 10:08:47 cnflnt4-02.jcc.com kafka-server-start[16232]: at org.apache.kafka.common.network.KafkaChannel.close(KafkaChan...:70)
Dec 20 10:08:47 cnflnt4-02.jcc.com kafka-server-start[16232]: at org.apache.kafka.common.network.Selector.doClose(Selector.java:746)
Dec 20 10:08:47 cnflnt4-02.jcc.com kafka-server-start[16232]: at org.apache.kafka.common.network.Selector.close(Selector.java:734)
Dec 20 10:08:47 cnflnt4-02.jcc.com kafka-server-start[16232]: at org.apache.kafka.common.network.Selector.pollSelectionKeys(S...532)
Dec 20 10:08:47 cnflnt4-02.jcc.com kafka-server-start[16232]: at org.apache.kafka.common.network.Selector.poll(Selector.java:424)
Dec 20 10:08:47 cnflnt4-02.jcc.com kafka-server-start[16232]: at kafka.network.Processor.poll(SocketServer.scala:665)
Dec 20 10:08:47 cnflnt4-02.jcc.com kafka-server-start[16232]: at kafka.network.Processor.run(SocketServer.scala:582)
Dec 20 10:08:47 cnflnt4-02.jcc.com kafka-server-start[16232]: at java.lang.Thread.run(Thread.java:748)
Dec 20 10:11:02 cnflnt4-02.jcc.com kafka-server-start[16232]: [2018-12-20 10:11:02,330] INFO [GroupMetadataManager brokerId=1...ger)
Dec 20 10:21:02 cnflnt4-02.jcc.com kafka-server-start[16232]: [2018-12-20 10:21:02,331] INFO [GroupMetadataManager brokerId=1...ger)
Hint: Some lines were ellipsized, use -l to show in full.
[keith@cnflnt4-02 ~]$
```

Confluent Schema Registry Configuration

- /etc/schema-registry/schema-registry.properties

```
# The address the socket server listens on.
listeners=http://cnflnt4-02.jcc.com:8081,https://cnflnt4-02.jcc.com:8082

### SSL Settings ###
kafkastore.ssl.truststore.location=/var/private/ssl/kafka.truststore.jks
kafkastore.ssl.truststore.password=<password>
kafkastore.ssl.keystore.location=/var/private/ssl/kafka.keystore.jks
kafkastore.ssl.keystore.password=<password>
kafkastore.ssl.key.password=<password>
kafkastore.security.protocol=SSL
...
# Alternatively, Schema Registry can now operate without Zookeeper, handling all coordination via
# Kafka brokers. Use this setting to specify the bootstrap servers for your Kafka cluster and it
# will be used both for selecting the master schema registry instance and for storing the data for
# registered schemas.
kafkastore.bootstrap.servers=SSL://cnflnt4-02.jcc.com:9093,SSL://cnflnt4-
03.jcc.com:9093,SSL://cnflnt4-04.jcc.com:9093

# The name of the topic to store schemas in
kafkastore.topic=_schemas
```

Confluent Schema Registry Service

```
[keith@cnflnt4-02 ~]$ sudo systemctl status confluent-schema-registry
? confluent-schema-registry.service - RESTful Avro schema registry for Apache Kafka
   Loaded: loaded (/usr/lib/systemd/system/confluent-schema-registry.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2018-12-16 19:30:43 EST; 3 days ago
     Docs: http://docs.confluent.io/
 Main PID: 15903 (java)
    Tasks: 33
   CGroup: /system.slice/confluent-schema-registry.service
           mq15903 java -Xmx512M -server -XX:+UseG1GC -XX:MaxGCPauseMillis=20 -XX:InitiatingHeapOccupancyPercent=35 -XX:+Explicit...

Dec 20 08:00:57 cnflnt4-02.jcc.com schema-registry-start[15903]: [2018-12-20 08:00:56,999] INFO 172.16.1.24 - - [20/Dec/2018:...:77)
Dec 20 08:00:57 cnflnt4-02.jcc.com schema-registry-start[15903]: [2018-12-20 08:00:57,393] INFO 172.16.1.24 - - [20/Dec/2018:...:77)
Dec 20 08:00:57 cnflnt4-02.jcc.com schema-registry-start[15903]: [2018-12-20 08:00:57,946] INFO 172.16.1.24 - - [20/Dec/2018:...:77)
Dec 20 08:00:59 cnflnt4-02.jcc.com schema-registry-start[15903]: [2018-12-20 08:00:59,084] INFO 172.16.1.24 - - [20/Dec/2018:...:77)
Dec 20 08:01:00 cnflnt4-02.jcc.com schema-registry-start[15903]: [2018-12-20 08:01:00,398] INFO 172.16.1.24 - - [20/Dec/2018:...:77)
Dec 20 08:01:11 cnflnt4-02.jcc.com schema-registry-start[15903]: [2018-12-20 08:01:11,587] INFO 172.16.1.24 - - [20/Dec/2018:...:77)
Dec 20 08:01:14 cnflnt4-02.jcc.com schema-registry-start[15903]: [2018-12-20 08:01:14,034] INFO 172.16.1.24 - - [20/Dec/2018:...:77)
Dec 20 08:01:14 cnflnt4-02.jcc.com schema-registry-start[15903]: [2018-12-20 08:01:14,431] INFO 172.16.1.24 - - [20/Dec/2018:...:77)
Dec 20 08:01:21 cnflnt4-02.jcc.com schema-registry-start[15903]: [2018-12-20 08:01:21,112] INFO 172.16.1.24 - - [20/Dec/2018:...:77)
Dec 20 08:01:32 cnflnt4-02.jcc.com schema-registry-start[15903]: [2018-12-20 08:01:32,476] INFO 172.16.1.24 - - [20/Dec/2018:...:77)
Hint: Some lines were ellipsized, use -l to show in full.
[keith@cnflnt4-02 ~]$
```

Confluent Schema Registry

- Schema information is published in the topic _schema

```
$ java -cp /confluent_kafka_path/*:. -
_$   kafka.admin.TopicCommand -
_$   --zookeeper cnflnt4-02.jcc.com:2181 -
_$   --topic _schemas -
_$   --describe
Topic:_schemas  PartitionCount:1        ReplicationFactor:3     Configs:cleanup.policy=compact
        Topic: _schemas Partition: 0    Leader: 1       Replicas: 1,3,2 Isr: 1,2,3
```

- _schema has a single partition, replicated across all nodes
- Includes version information, id, record description
- Record description is JSON

Schema Registry Example

- First time a COLLEGES record is processed

```
$ java -cp /confluent_kafka_path/*:. kafka.tools.ConsoleConsumer -
_$    --bootstrap-server cnflnt4-04.jcc.com:9093 -
_$    --property schema.registry.url="http://cnflnt4-04.jcc.com:8081" -
_$    --topic _schemas -
_$    --consumer.config KAFKA_AVRO.PROPERTIES
{"subject":"Personnel.COLLEGES-
value","version":1,"id":81,"schema":"{\"type\":\"record\",\"name\":\"COLLEGES\",\"namespace\":\"MFP.a
vro\",\"fields\":[{\"name\":\"COLLEGE_CODE\",\"type\":\"string\"},{\"name\":\"COLLEGE_NAME\",\"type\"
:[\"null\",\"string\"],\"default\":null},{\"name\":\"CITY\",\"type\":[\"null\",\"string\"],\"default\
":null},{\"name\":\"STATE\",\"type\":[\"null\",\"string\"],\"default\":null},{\"name\":\"POSTAL_CODE\
",\"type\":[\"null\",\"string\"],\"default\":null},{\"name\":\"CDC_FLAG\",\"type\":[\"null\",
\"string\"],\"default\":null},{\"name\":\"LML_COMMIT_TIME\",\"type\":[\"null\",{\"type\":\"long\",\"l
ogicalType\":\"timestamp-
micros\"}],\"default\":null},{\"name\":\"JCCLML_LSN\",\"type\":[\"null\",\"long\"],\"default\":null}]
}","deleted":false}
```

Kafka Configuration Summary

- Kafka is fairly easy to configure and manage
  - Once you find the correct white papers and blogs
  - Once you figure out the parts and terminology
- We have not (yet) tested resilience to node crashes
- We still have much to learn about performance configuration and monitoring

Summary

- Kafka provides a reliable message queuing service
- JCC LogMiner Loader Kafka Option supports publishing committed Rdb transactions to a Kafka server
- Data published to Kafka is available to variety of downstream tools
- Data can be published in formats:
  - XML
  - JSON
  - Avro
- Avro allows Kafka consumers to easily handle metadata changes