Integrations:

Kafka

Objective

Scalable stream processing toolkit

Structured Streaming + DStreams integration

Structured Streaming & Kafka

Read like any other data source

```
val kafkaDF = spark.readStream
    .format("kafka")
    .option("kafka.bootstrap.servers", "localhost:9092")
    .option("subscribe", "mySuperTopic")
    .load()
```

need to specify bootstrap server and topic

Write like any other data source

```
kafkaDF
    .writeStream
    .format("kafka")
    .option("kafka.bootstrap.servers", "localhost:9092")
    .option("topic", "mySuperTopic")
    .option("checkpointLocation", "checkpoints")
    .start()
    .awaitTermination()
```

need to specify bootstrap server, topic and checkpoint location

Can also configure offsets, etc:

https://spark.apache.org/docs/latest/structured-streaming-kafka-integration.html

DStreams & Kafka: Read

More complex reading steps

```
val kafkaParams = Map[String, Object](
    "bootstrap.servers" -> "oneHost:9092, someOtherHost:9092",
    "key.serializer" -> classOf[StringSerializer],
    "value.serializer" -> classOf[StringSerializer],
    "key.deserializer" -> classOf[StringDeserializer],
    "value.deserializer" -> classOf[StringDeserializer],
    "auto.offset.reset" -> "latest",
    "enable.auto.commit" -> (false: java.lang.Boolean)
)
```

for Java compatibility

```
val stream = KafkaUtils.createDirectStream(
    ssc,
    PreferConsistent,
    Subscribe[String, String](topics, kafkaParams + ("group.id" -> "group1"))
)
```

for Kafka cache requirements

DStreams & Kafka: Write

More complex writing, per RDD, per partition

```
stream.foreachRDD { rdd =>
    rdd.foreachPartition { partition =>
        // need to create a Kafka producer here, on this executor
        val producer = new KafkaProducer[String, String](kafkaHashMap)

        // create records to send to Kafka
        partition.foreach { record =>
            val message = new ProducerRecord[String, String](kafkaTopic, null, record)
            producer.send(message)
        }

        producer.close()
    }
}
```

Remember to

- create the producer per partition as producers are not serializable
- create a Kafka message per each record
- close the producer when you're done processing the partition

Spark rocks