Lesson 6 Hands-On
Directions
In this lesson, you've learned all about how to work with data in real-time. Now it's time to make sense of all that knowledge with a Hands-On project! This Hands-On will be graded, so make sure you complete each part.
Caution!
Do not submit your project until you have completed all requirements, as you will not be able to resubmit.

Description
Please compare and contrast the various programs available for you to use in real-time. You should at least be able to answer the following questions in your written response:

- What programs behave similarly?
- How do these programs differ from each other?
- If you could only pick one program, which one would you choose and why?

The programs are available for me to use in real-time are Spark Streaming, Apache Storm, and Flink. Flink is the fastest data streaming framework listed and the Spark Streaming is the slowest. Apache Storm and Flink work in real-time, while Spark Streaming works in micro-batches. If I could only pick one program, I would go with Flink. Flink is fully scalable, has strong fault tolerance, uses event-based real-time data, and it is fastest.

Apache Flink vs Apache Spark

Features	Apache Flink	Apache Spark
Computation Model	Flink is based on the operator-based computational model.	Spark is based on the micro-batch modal.
Streaming engine	Apache Flink uses streams for all workloads: streaming, SQL, micro-batch and batch. Batch is a finite set of streamed data.	Apache Spark uses micro-batches for all workloads. But it is not sufficient for use cases where we need to process large streams of live data and provide results in real time.
Iterative processing	Flink API provides two dedicated iterations operation Iterate and Delta Iterate.	Spark is based on non-native iteration which is implemented as regular for – loops outside the system.

Optimization	Apache Flink comes with an optimizer that is independent with the actual programming interface.	In Apache Spark jobs has to be manually optimized.
Latency	With minimum efforts in configuration Apache Flink's data streaming run-time achieves low latency and high throughput.	Apache Spark has high latency as compared to Apache Flink.
Performance	Overall performance of Apache Flink is excellent as compared to any other data processing system. Apache Flink uses native closed loop iterations operators which makes machine learning and graph processing more faster.	Though Apache Spark has an excellent community background and now It is considered as most matured community. But Its stream processing is not much efficient than Apache Flink as it uses micro-batch processing.
Fault tolerance	The fault tolerance mechanism followed by Apache Flink is based on Chandy-Lamport distributed snapshots. The mechanism is lightweight, which results in maintaining high throughput rates and provide strong consistency guarantees at the same time.	Spark Streaming recovers lost work and delivers exactly-once semantics out of the box with no extra code or configuration.(refer Spark fault tolerant feature guide)
Duplicate elimination	Apache Flink process every records exactly one time hence eliminates duplication.	Spark also process every record exactly one time hence eliminates duplication.
Window Criteria	Flink has a record-based or any custom user-defined Window criteria.	Spark has a time-based Window criteria
Memory -Management	Flink provides automatic memory management.	Spark provides configurable memory management. Spark 1.6, Spark has moved towards automating memory management as well.
Speed	Flink processes data at lightening fast speed	Spark's processing model is slower than Flink

Apache Flink vs Apache Spark - A comparison guide - DataFlair (data-flair.training)

Storm vs. Spark: Comparison

Both Storm and Spark are free-to-use and open-source Apache projects with a similar intent. The table below outlines the main difference between the two technologies:

	Storm	Spark
Programming Languages	Multi-language integration	Support for Python, R, Java, Scala
Processing Model	Stream processing with micro-batching available through Trident	Batch processing with micro-batching available through Streaming
Primitives	Tuple stream Tuple batch Partition	DStream
Reliability	Exactly once (Trident) At least once At most once	Exactly once
Fault Tolerance	Automatic restart by the supervisor process	Worker restart through resource managers
State Management	Supported through Trident	Supported through Streaming
Ease of Use	Harder to operate and deploy	Easier to manage and deploy

[Apache Storm vs. Spark: Side-by-Side Comparison (phoenixnap.com)]

Other programs not mentioned in the book:
[Spark Streaming vs Flink vs Storm vs Kafka Streams vs Samza : Choose Your Stream Processing Framework | by chandan prakash | Medium]