

App deployment in PySpark

- The Spark submit script provides many options to specify how the application should run
 - Most are the same as for `pyspark` and `spark-shell`
- General submit flags include
 - `master`: `local`, `yarn`, or a Mesos or Spark Standalone cluster manager URI
 - `jars`: Additional JAR files
 - `pyfiles`: Additional Python files (Python only)
 - `driver-java-options`: Parameters to pass to the driver JVM
- YARN-specific flags include
 - `num-executors`: Number of executors to start application with
 - `driver-cores`: Number cores to allocate for the Spark driver
 - `queue`: YARN queue to run in
- Show all available options
 - `help`

```
$ spark-submit NameList.py people.json namelist/
```

Spark Streaming

Use Cases:

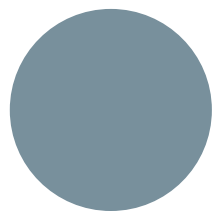
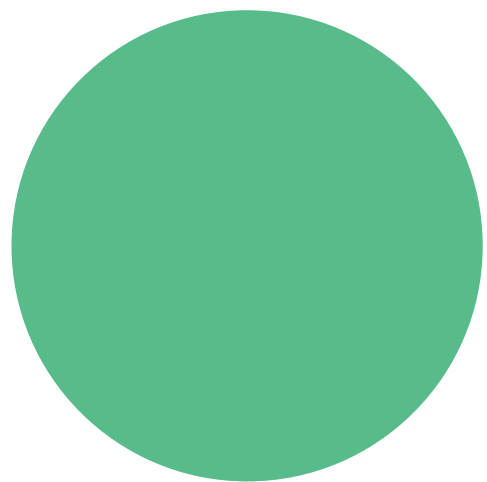
- Many big data applications need to process large data streams in real time, such as
 - Continuous ETL
 - Website monitoring
 - Fraud detection
 - Advertisement monetization
 - Social media analysis
 - Financial market trends
 - Event-based data



Spark Streaming v/s DStreams

Structured Streaming	DStreams API
<ul style="list-style-type: none">■ DataFrame/Dataset-based■ Higher level API■ Best for structured or semi-structured data■ Provides SQL-like semantics■ Guarantees consistency between streaming and static queries■ Queries optimized by the Catalyst optimizer	<ul style="list-style-type: none">■ RDD-based■ Lower level API■ Best for unstructured data

- **Designed for end-to-end, continuous, real-time data processing**
- **Ensures consistency**
 - Guarantees exactly-once handling
 - Query results are the same on static or streaming data
- **Built-in support for time-series data**
 - Handles out-of-order and late events



Thank you!



Saurav Agarwal