Checkpoints

Why do we need Checkpointing

- Streams can be running 24/7
- Don't want to lose everything in event of data failure
- Checkpointing the data in fault-tolerant storage is needed.

Types of Checkpointing

- *Metadata checkpointing* - Saving of the information defining the streaming computation to fault-tolerant storage like HDFS. This is used to recover from failure of the node running the driver of the streaming application.
- *Data checkpointing* - Saving of the generated RDDs to reliable storage. This is necessary in some *stateful* transformations that combine data across multiple batches.

Configuring Checkpointing

Checkpointing can be enabled by setting a directory in a fault-tolerant, reliable file system to which the checkpoint information will be saved. This is done by using `streamingContext.checkpoint(checkpointDirectory)`.

Streaming application should have the following behavior.

- When the program is being started for the first time, it will create a new `StreamingContext`, set up all the streams and then call `start()`.
- When the program is being restarted after failure, it will re-create a `StreamingContext` from the checkpoint data in the checkpoint directory.