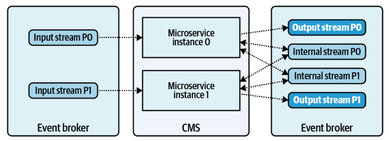
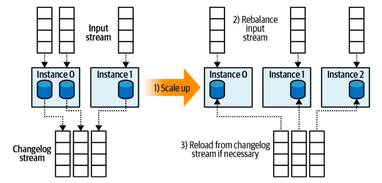
Adam Bellemare - Building Event-Driven Microservices

12. Lightweight Framework Microservices

- Benefits and Limitations
  - Provide similar functionality to Heavyweight Frameworks, but in a way that heavily leverages
    - the event broker
    - the container management system (CMS)
  - In many cases exceed Heavyweight Frameworks
    - 

      The lightweight framework model, showcasing the usage of internal event streams for repartitioning data
- Lightweight Processing
  - mirrors the processing methodology of the heavyweight framework
  - Data of the same key must be local for any key-based operations, such as a join, or a groupByKey operation followed by a subsequent reduce/ aggregation
    - Copartitioning
      - Using event broker and CMS
- Handling State and Using Changelogs
  - Apps can use different resources (state stores, etc) from the cluster depending on their needs
  - Using event broker to store changelogs
- Scaling Applications and Recovering from Failures
  - 

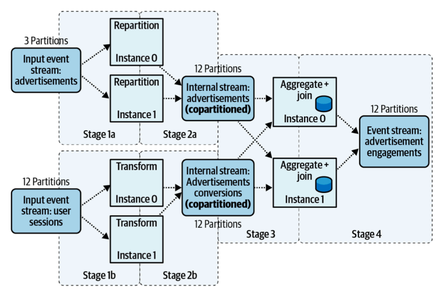
    Scaling up a lightweight microservice
  - Apps can be dynamically scaled
  - Event Shuffling
    - Any dynamic scaling requires only that the consumers be reassigned to the internal event stream, regardless of the producers
  - State Assignment
    - The operator state (the mappings of <partitionId, offset>)
      - is stored within the consumer group for the individual application
    - The keyed state (pairs of <key, state>)
      - is stored within the changelog for each state store in the application
  - State Replication and Hot Replicas
    - Hot Replicas
      - allow you to maintain high availability during scaling and
      - but they do come at the cost of additional disk and CPU usage
    - State Replication
      - populate a replica of the state on the new instance
      - wait until it's caught up to the head of the changelog
      - and then rebalance to assign it ownership of the input partitions
- Choosing a Lightweight Framework
  - currently there are only two options
    - Both use Apache Kafka event broker
  - Apache Kafka Streams
  - Apache Samza: Embedded Mode
    - embedded mode allows you to embed this functionality within individual applications, just like any other Java library
    - embedded mode may not provide all of the functionality that it has in cluster mode
- Languages and Syntax
  - JVM/Java
  - SQL and KSQL (by Confluent)
- Example: Stream-Table-Table Join: Enrichment Pattern

```
KStream<WindowKey,Actions> userSessions = ...

//Transform 1 userSession into 1 to N conversion events, rekey on AdvertisementId
KTable<AdvertisementId,Long> conversions = userSessions
  .transform(...)  //transform userSessions into conversion events
  .groupByKey()
  .aggregate(...)  //Creates an aggregate KTable

//Materialize the advertisement entities
KTable<AdvertisementId,Advertisement> advertisements = ...

//The tables are automatically co-partitioned by including
//the join operation in the topology.
conversions
  .join(advertisements, joinFunc) //See stage 4 for more details.
  .to("AdvertisementEngagements")
```



Processing topology for advertising engagement-sessions