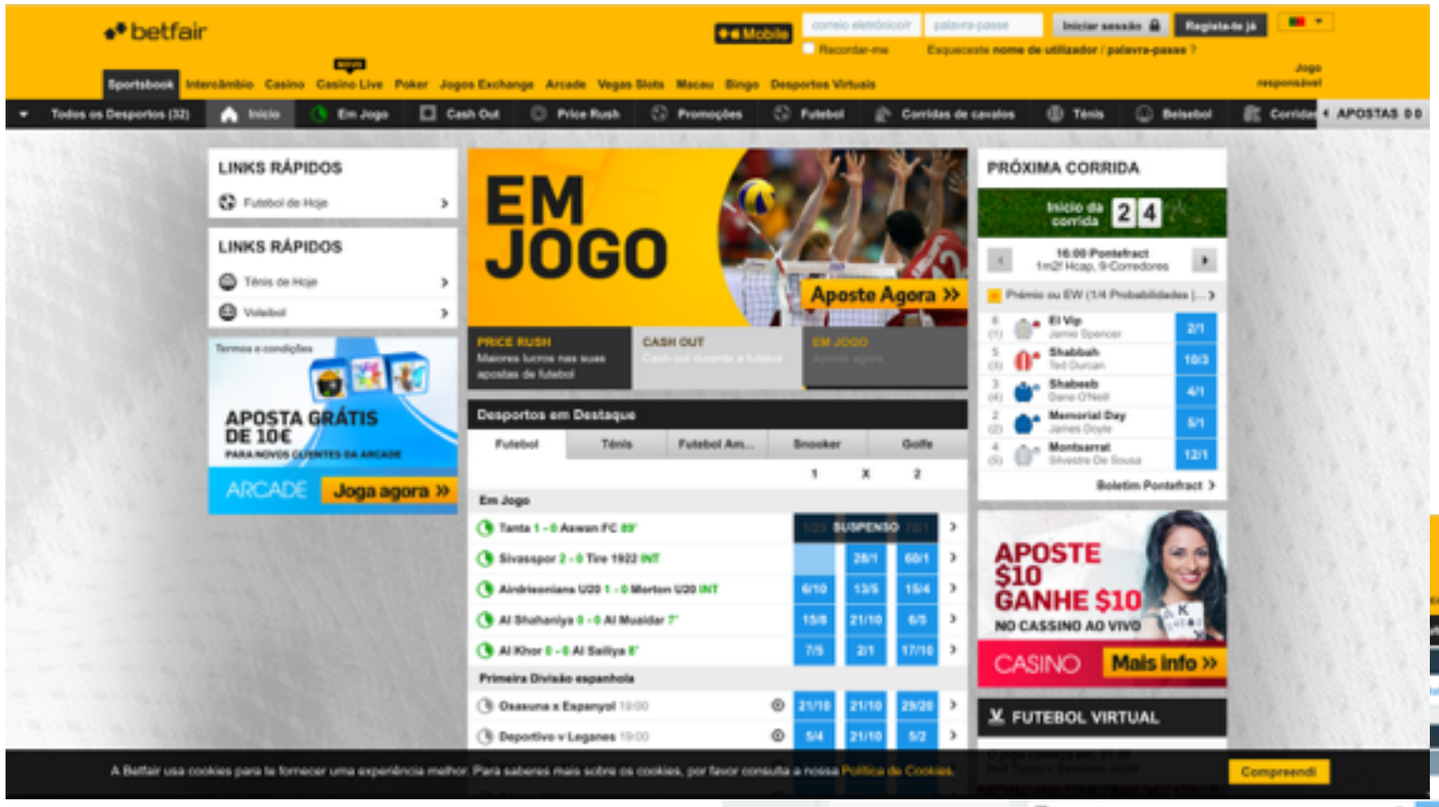
Stream Processing at Betfair

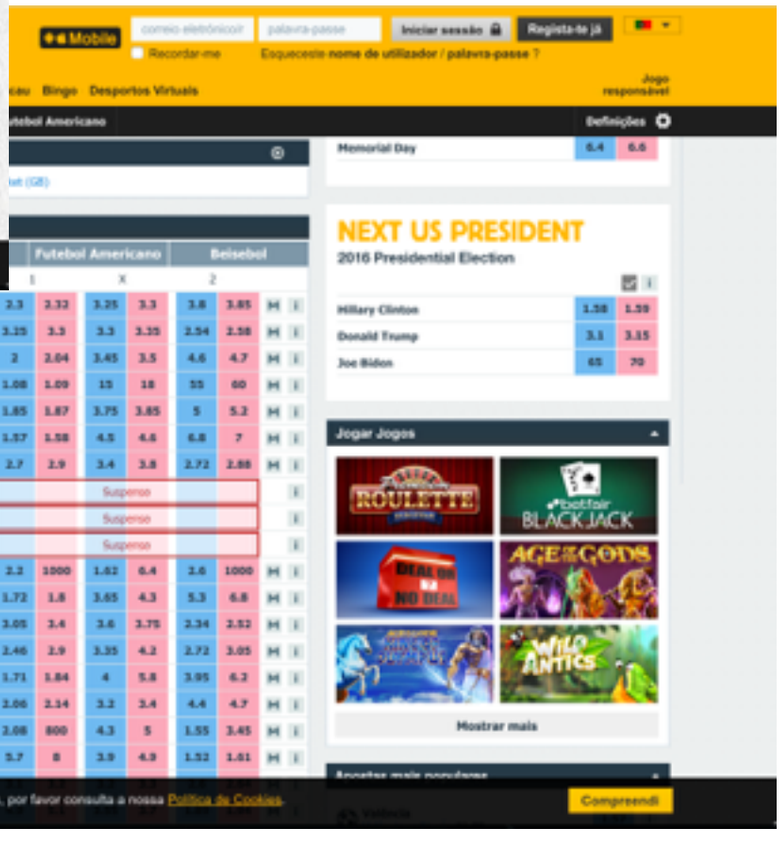
with Storm & Kafka

About me:

- Miguel Veiga
- Backend Developer
- At Blip since 2013
- Feeds Platform teams

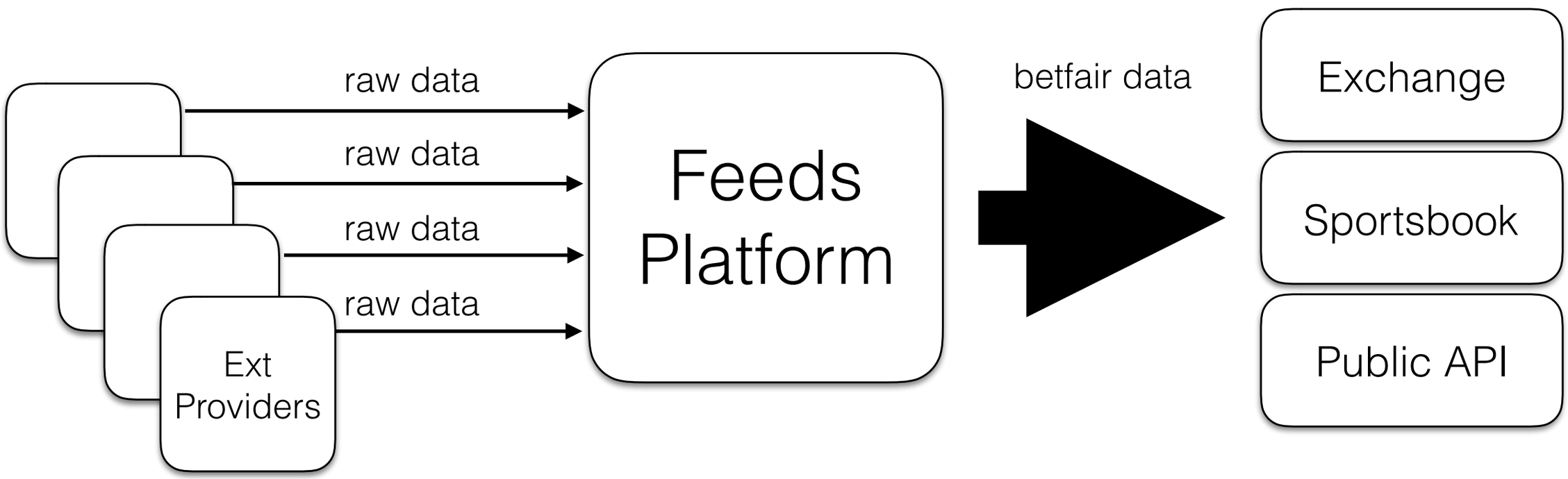
blip web engineers





Numbers:

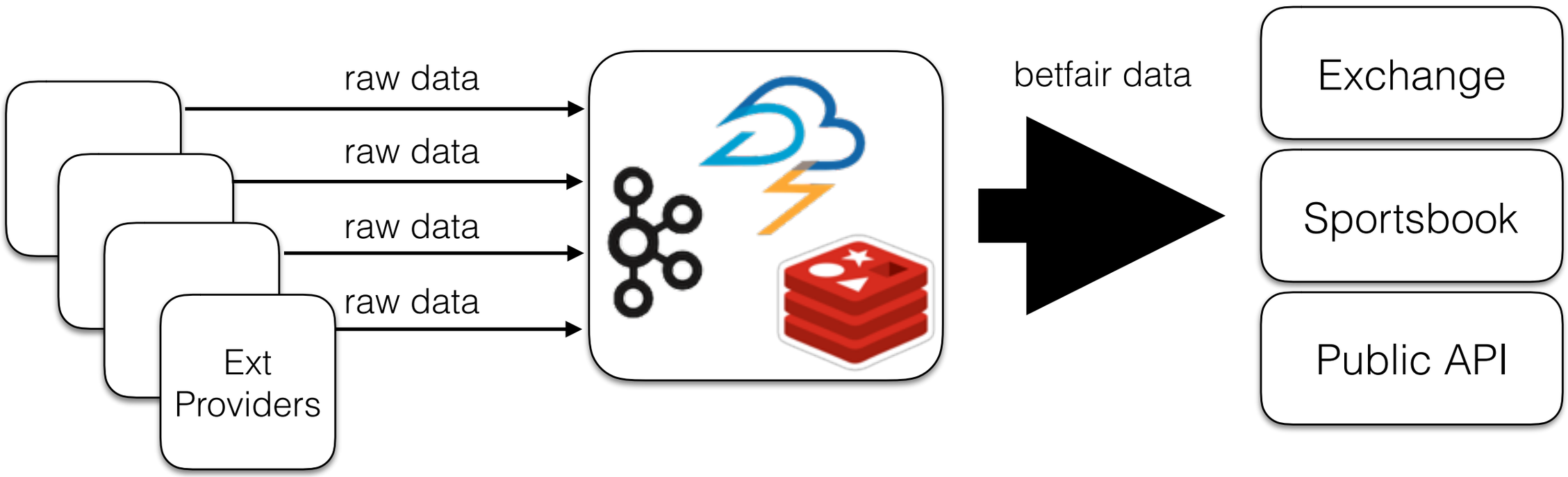
- More than **10** different external providers
- Providing data feeds for **20** different Sports
- Over **1 million** messages per day (and growing)


raw data
raw data
raw data
raw data
raw data
Ext
Providers
Feeds
Platform
betfair data
Exchange
Sportsbook
Public API

We wanted it to be:

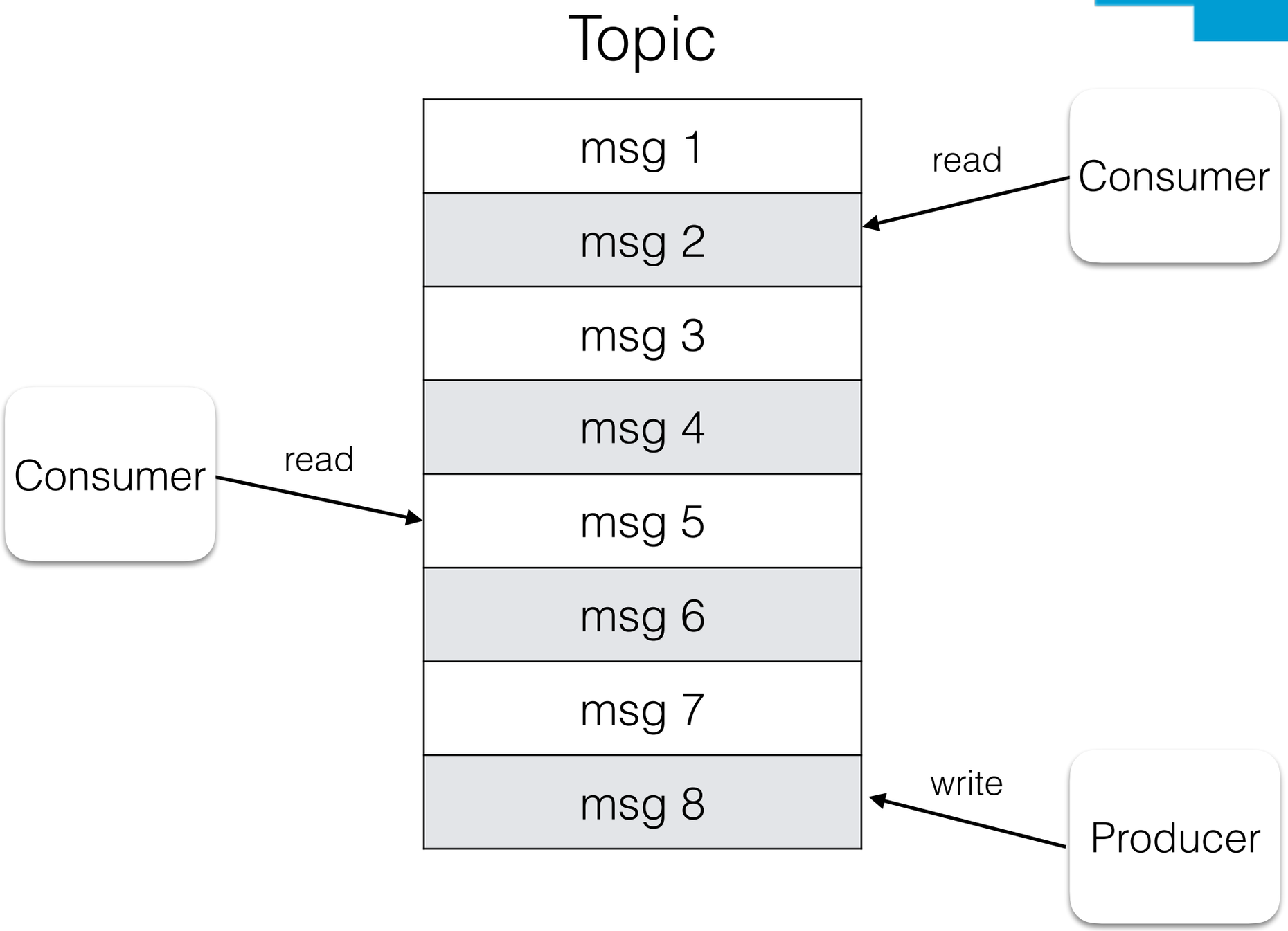
- Consistently Responsive
- Resilient
- Scalable
- Easy to Extend

blip web engineers

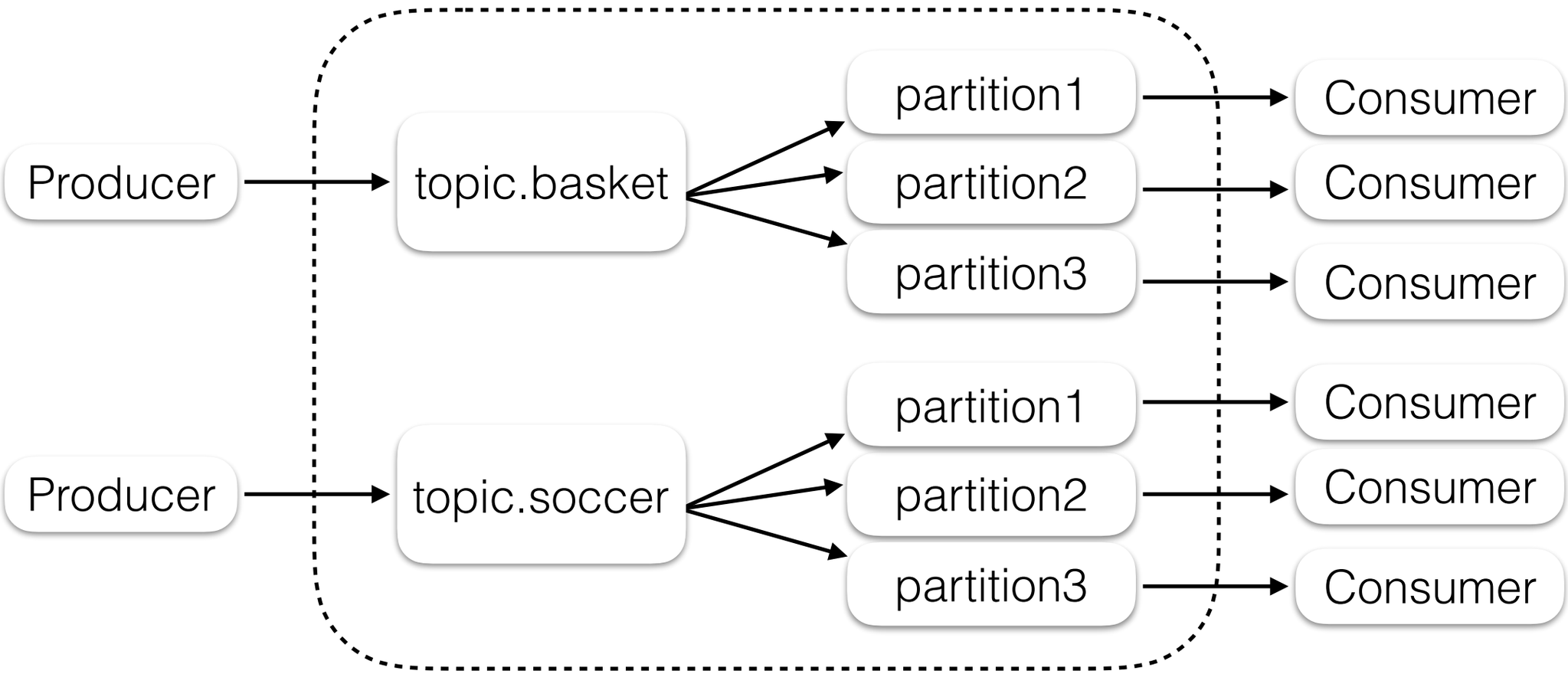


Kafka:

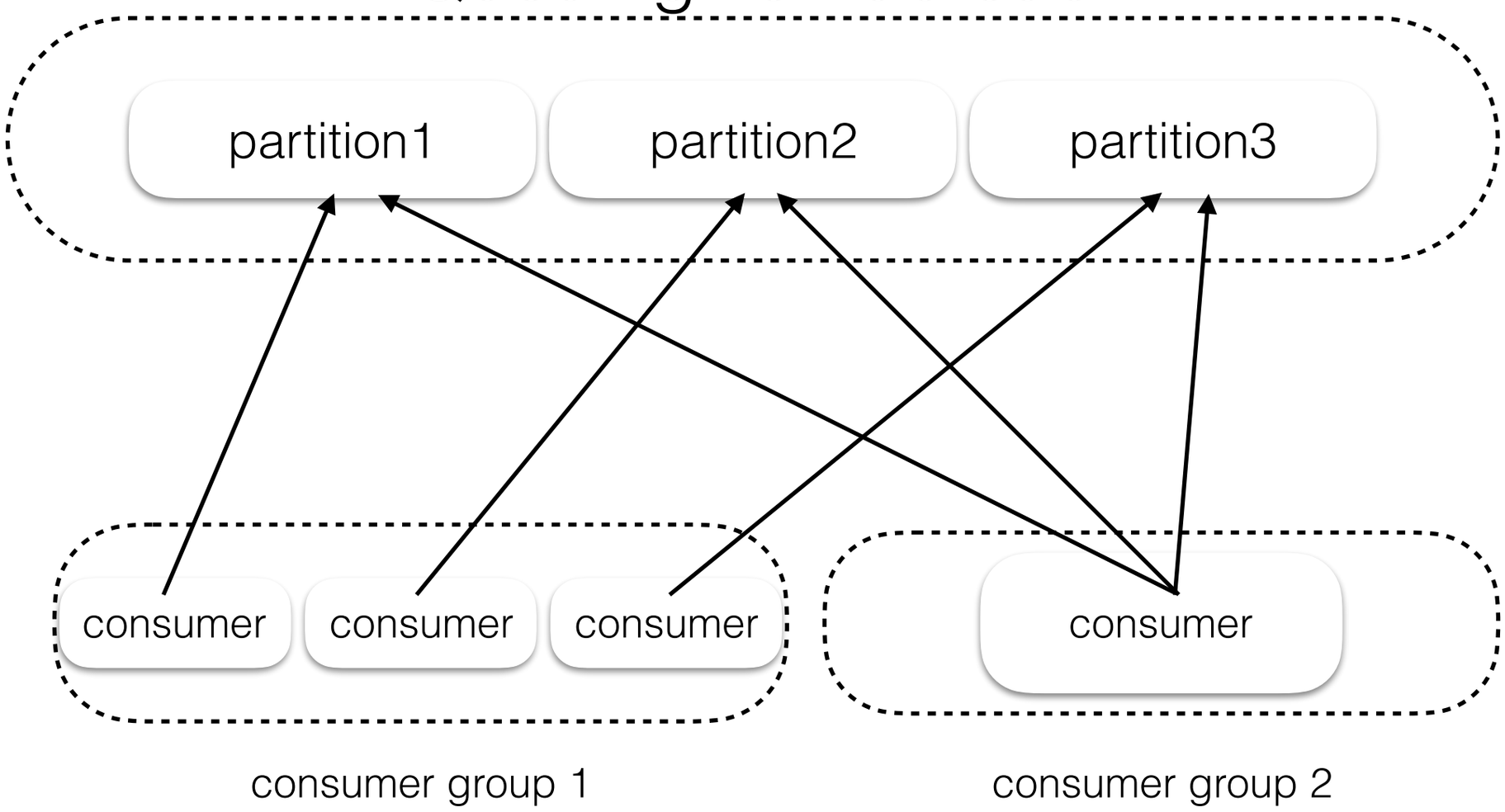
- Publish-subscribe messaging system
- Distributed as a Commit log
- Scalable, Resilient, High Throughput


Topic
msg 1
msg 2
msg 3
msg 4
msg 5
msg 6
msg 7
msg 8
read
Consumer
Consumer
read
write
Producer

Partitioning

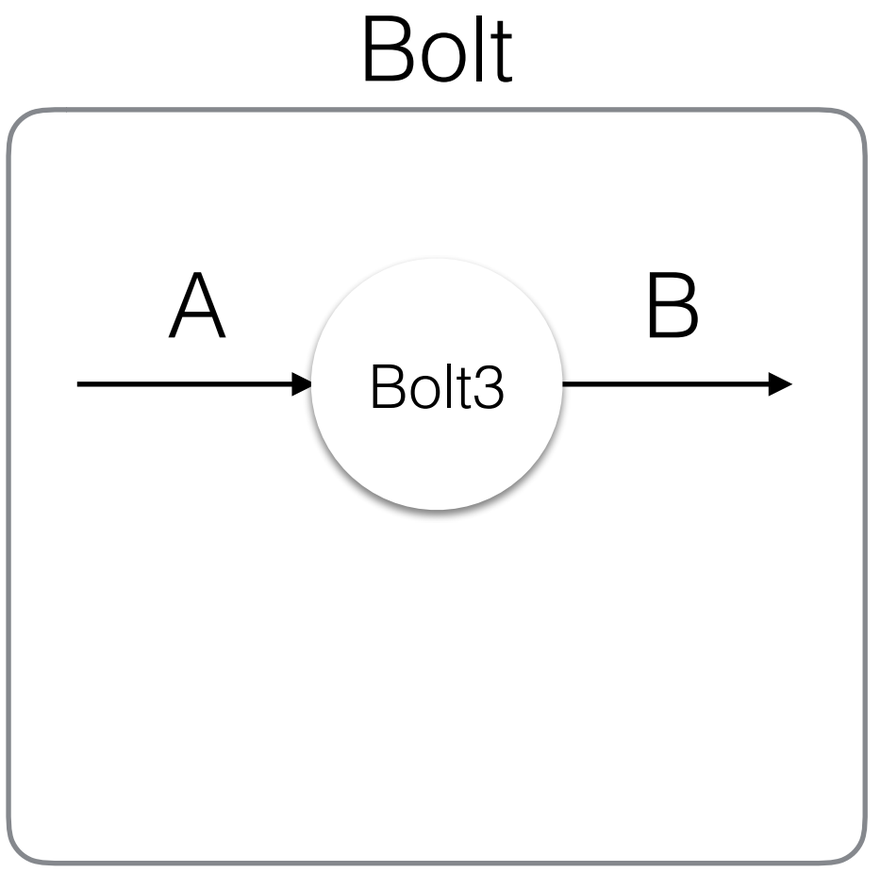


Queuing vs Pub-sub

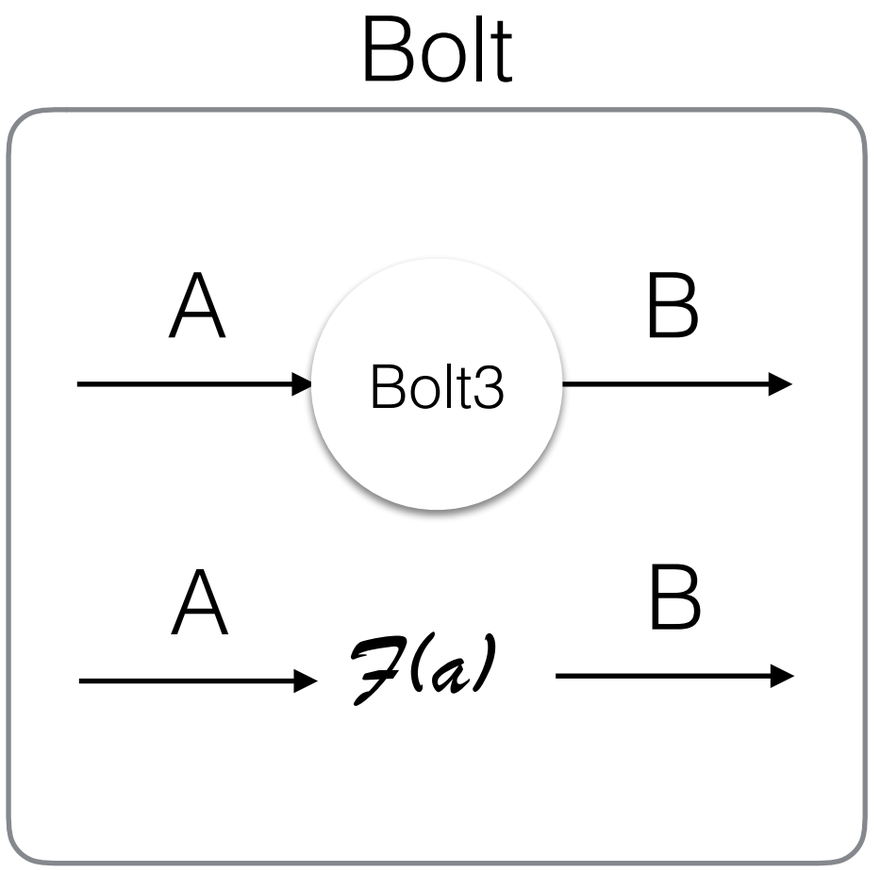


Storm:

- Highly distributed realtime stream processing system
- Scalable & fault-tolerant
- Data processing guaranties

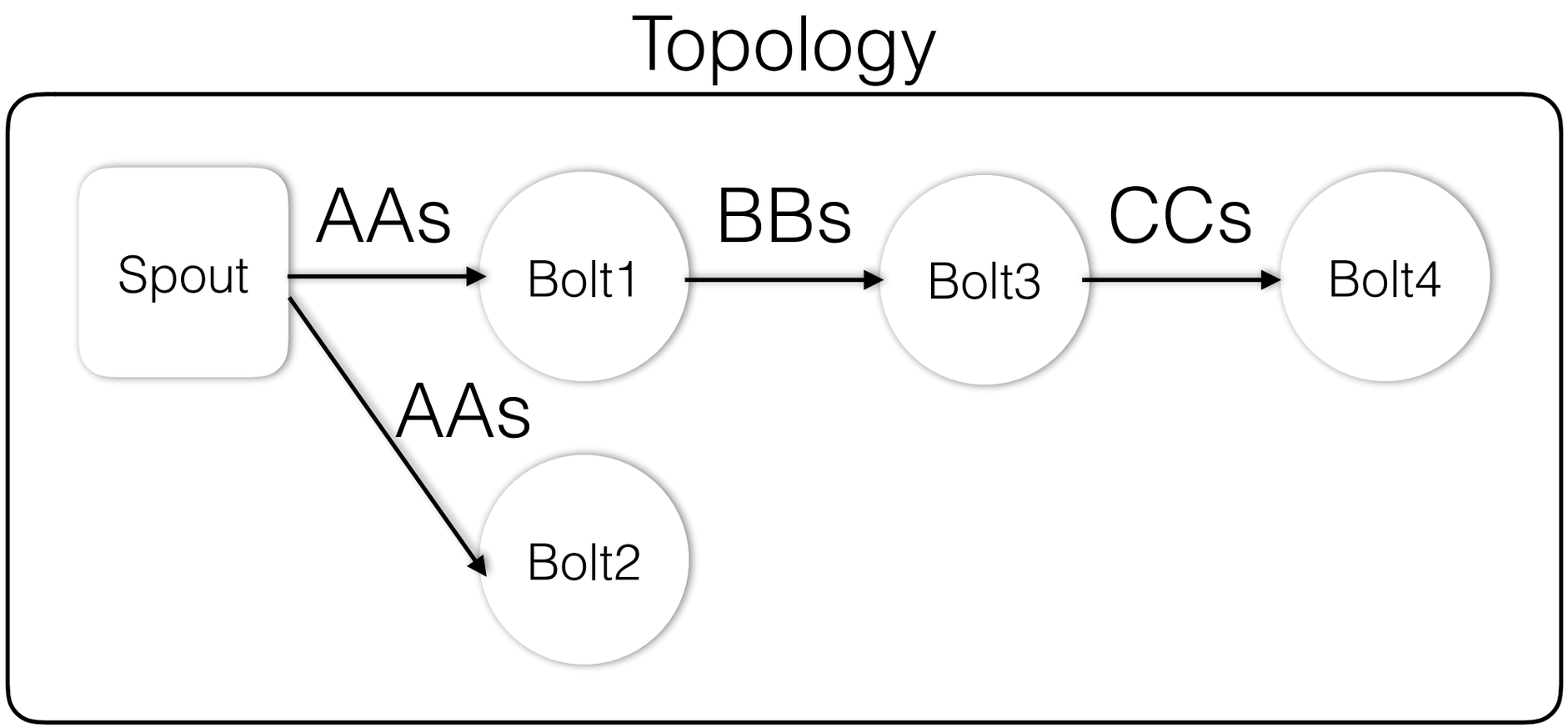


- Single Responsibility
- Loose Coupling
- Preserve Referential Transparency

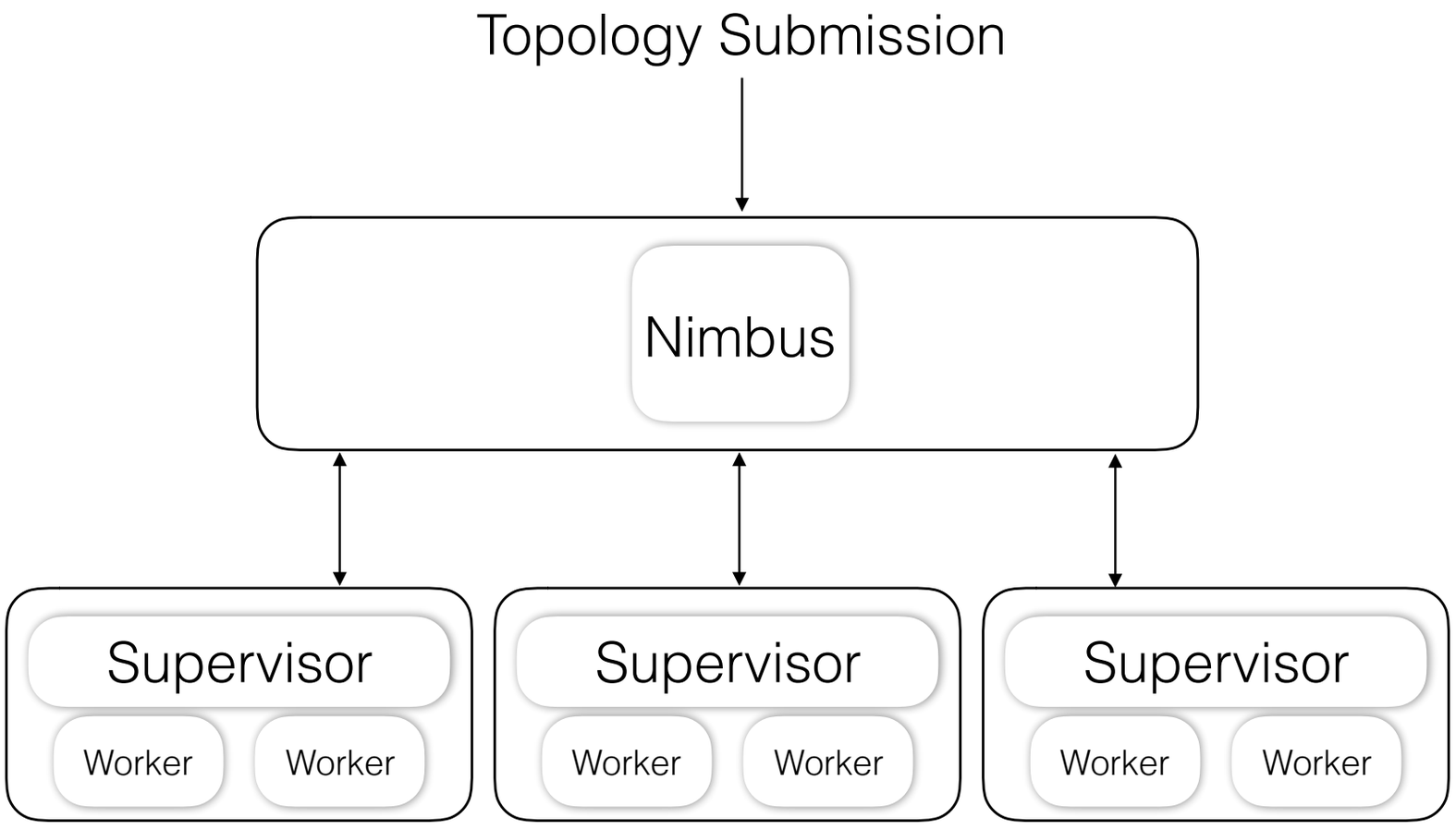


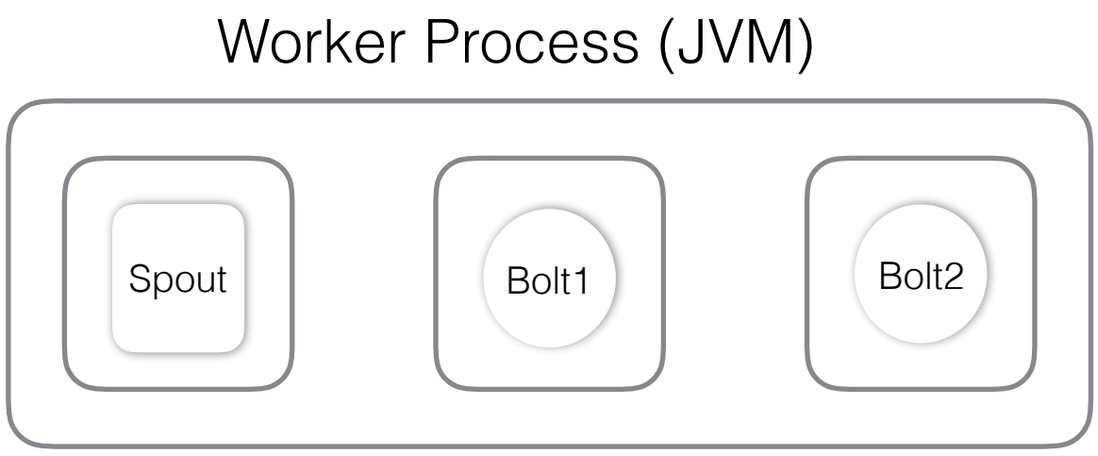
- Single Responsibility
- Loose Coupling
- Preserve Referential Transparency

```java
public class SampleBolt extends BaseRichBolt {

    private static final Logger LOGGER = LoggerFactory.getLogger(SampleBolt.class);

    private OutputCollector outputCollector;

    @Override
    public void prepare(Map map, TopologyContext topologyContext, OutputCollector outputCollector) {
        LOGGER.info("Bolt dependencies initialisation goes here!");
        this.outputCollector = outputCollector;
    }

    @Override
    public void execute(Tuple tuple) {

        LOGGER.info("Some business logic execution goes here!");
        String message = tuple.getString(0);

        getOutputCollector().emit(tuple, new Values(message + " : message consumed"));
        getOutputCollector().ack(tuple);
    }

    @Override
    public void declareOutputFields(OutputFieldsDeclarer outputFieldsDeclarer) {
        outputFieldsDeclarer.declare(new Fields(Keys.FEED.getKey()));
    }

    @Override
    public void cleanup() {
        LOGGER.info("Bolt dependencies cleanup goes here!");
    }

    public OutputCollector getOutputCollector() {
        return outputCollector;
    }
}
```


Topology
Spout
AAs
Bolt1
BBs
Bolt3
CCs
Bolt4
AAs
Bolt2

```
public class SampleTopology {

    public static StormTopology createTopology() {

        Injector injector = createInjector(new TopologyModule());
        BaseRichSpout Kafka = injector.getInstance(BaseRichSpout.class);
        BaseRichBolt sample = injector.getInstance(SampleBolt.class);
        BaseRichBolt publisher = injector.getInstance(PublisherBolt.class);

        TopologyBuilder builder = new TopologyBuilder();
        builder.setSpout("Kafka", Kafka, 3);
        builder.setBolt("sample", sample, 3).shuffleGrouping("Kafka");
        builder.setBolt("publisher", publisher, 3).shuffleGrouping("sample");

        return builder.createTopology();
    }

    public static void main(String[] args) throws Exception {

        Config conf = new Config();
        conf.setDebug(false);
        conf.setNumWorkers(1);
        StormSubmitter.submitTopologyWithProgressBar(args[0], conf, createTopology());
    }
}
```

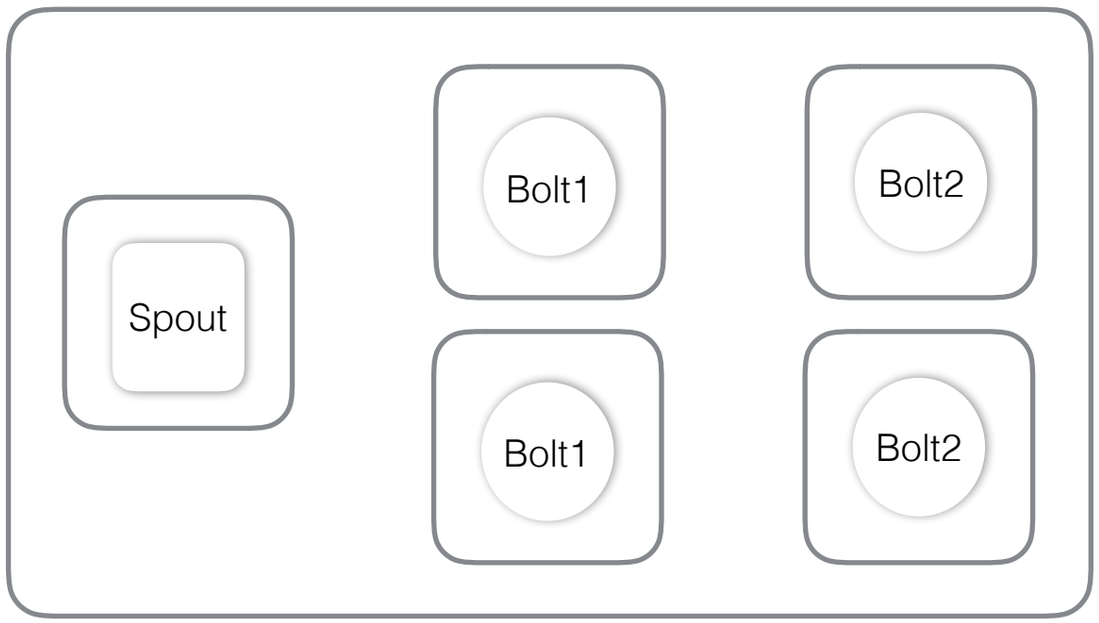

Topology Submission
Nimbus
Supervisor
Worker
Worker
Supervisor
Worker
Worker
Supervisor
Worker
Worker



parallelism hint == number of executors

```
builder.setSpout("Spout", spout, 1)
builder.setBolt("Bolt1", bolt1, 1)
builder.setBolt("Bolt2", bolt2, 1)
…
conf.setNumWorkers(1);
```

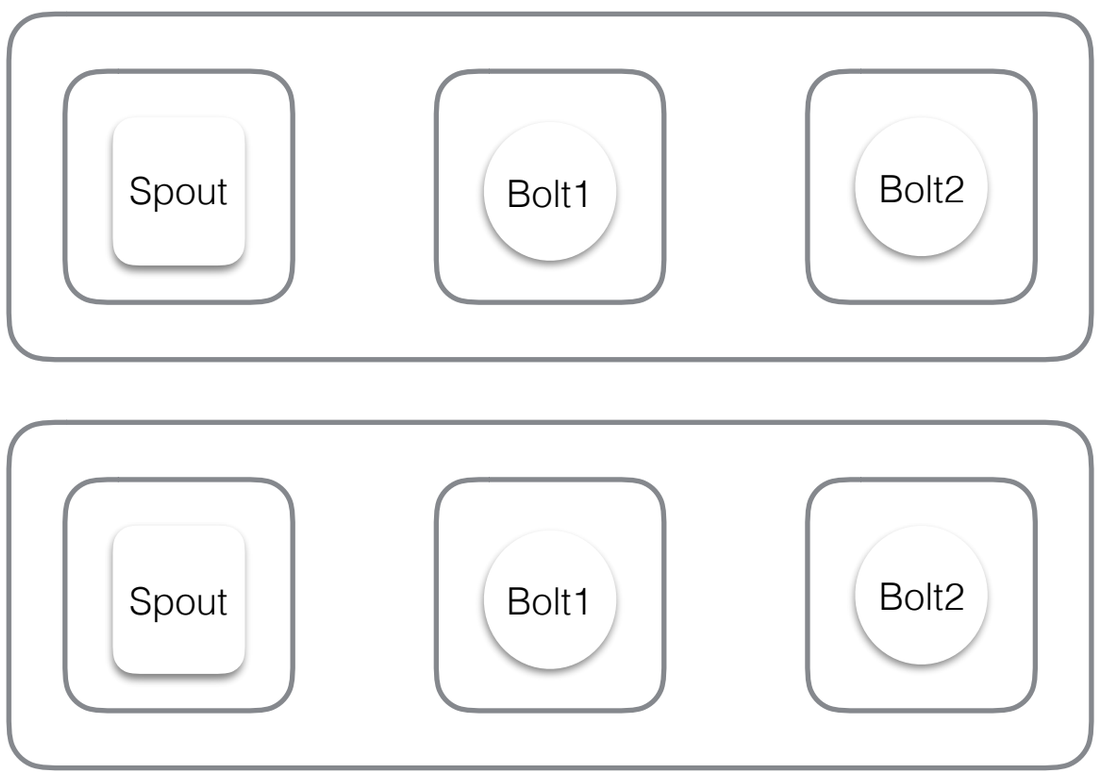
number of worker processes



parallelism hint == number of executors

```
builder.setSpout("Spout", spout, 1)
builder.setBolt("Bolt1", bolt1, 2)
builder.setBolt("Bolt2", bolt2, 2)
…
conf.setNumWorkers(1);
```

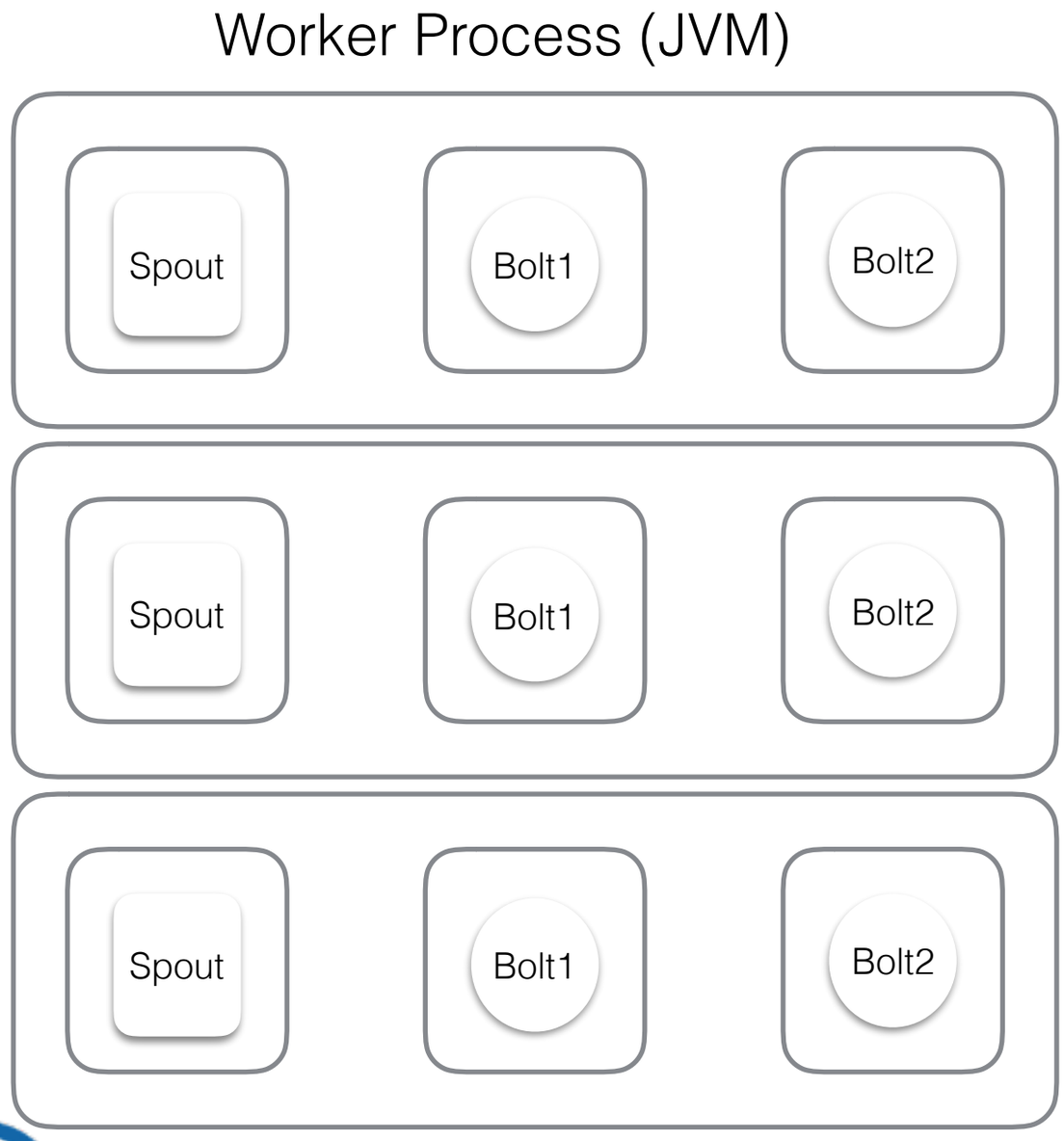
number of worker processes



parallelism hint == number of executors

```
builder.setSpout("Spout", spout, 2)
builder.setBolt("Bolt1", bolt1, 2)
builder.setBolt("Bolt2", bolt2, 2)
…
conf.setNumWorkers(2);
```

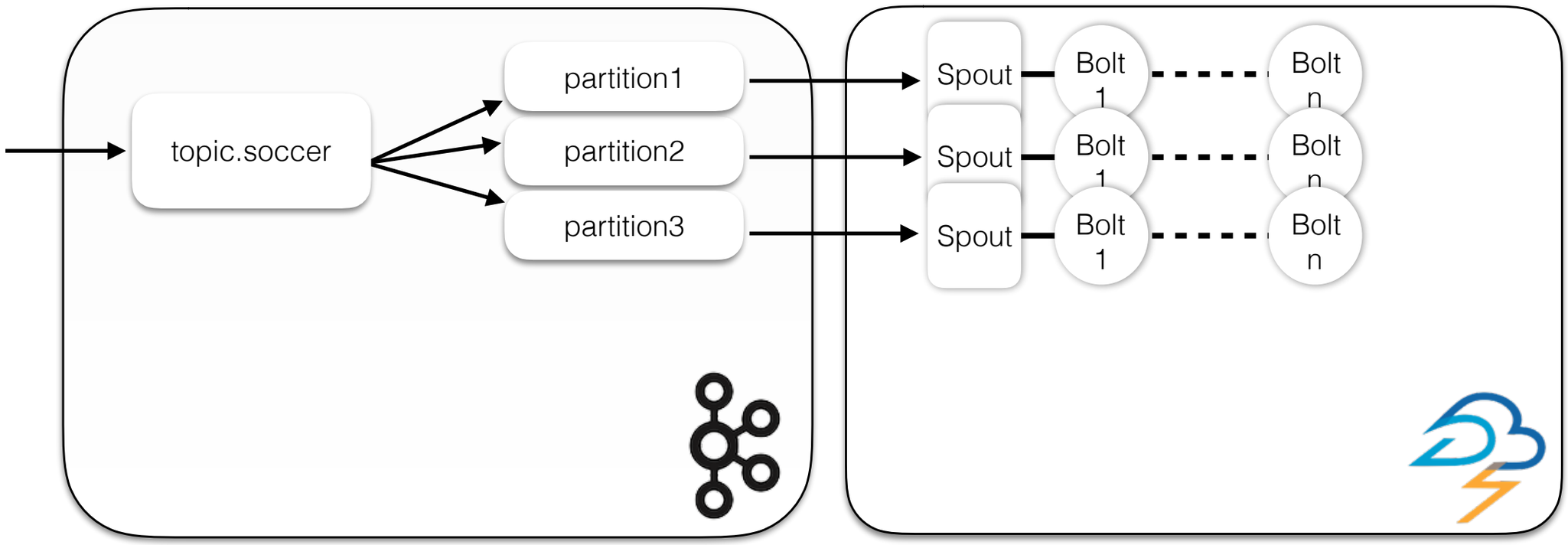
number of worker processes

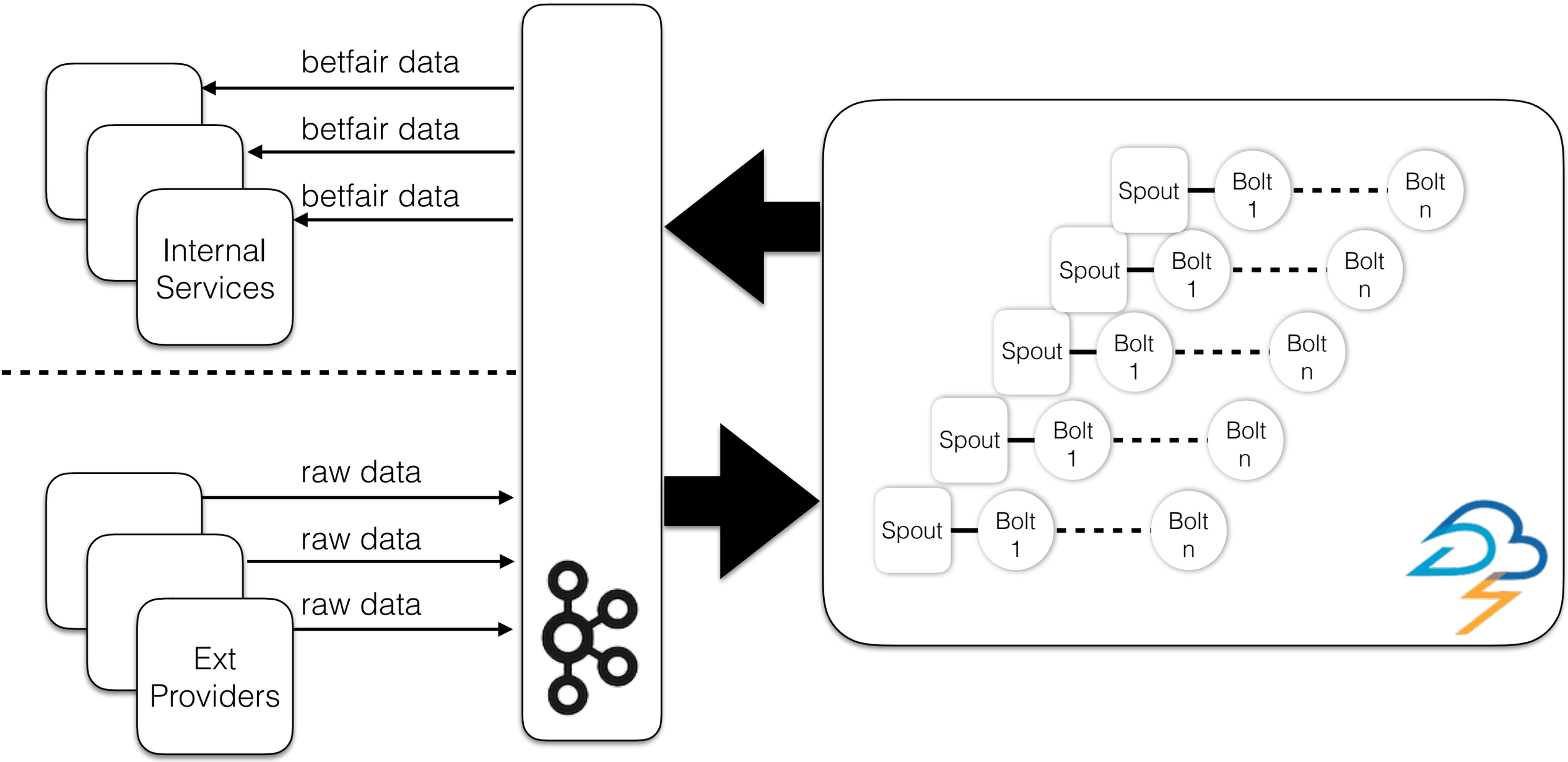


parallelism hint == number of executors

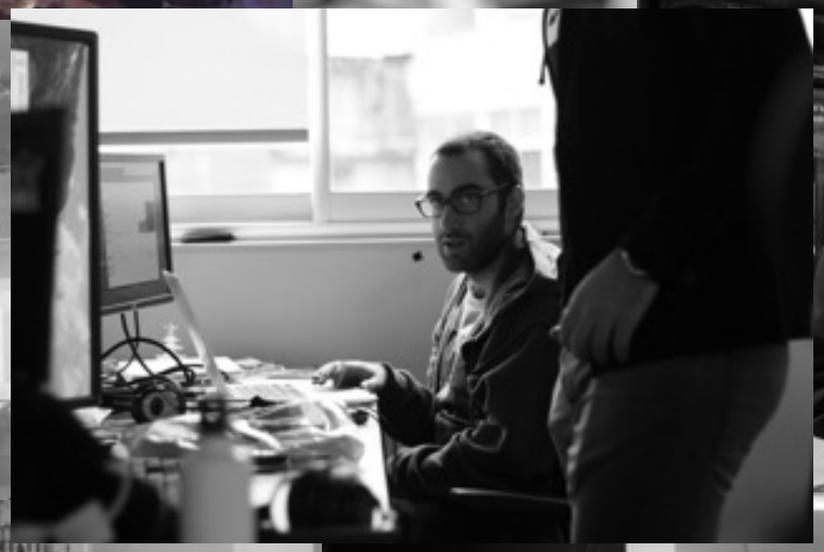
```
builder.setSpout("Spout", spout, 3)
builder.setBolt("Bolt1", bolt1, 3)
builder.setBolt("Bolt2", bolt2, 3)
…
conf.setNumWorkers(3);
```

number of worker processes


topic.soccer
partition1
partition2
partition3
Spout
Spout
Spout
Bolt 1
Bolt n
Bolt 1
Bolt n
Bolt 1
Bolt n


betfair data
betfair data
betfair data
Internal Services
raw data
raw data
raw data
Ext Providers
Spout
Bolt 1
Bolt n
Spout
Bolt 1
Bolt n
Spout
Bolt 1
Bolt n
Spout
Bolt 1
Bolt n
Spout
Bolt 1
Bolt n

Feeds Platform:

- 3 nodes Kafka Cluster
- 52 Topics
- 53 nodes Storm Cluster
- 26 Topologies
- Event/Market Management & Live-scores.

blip
web
engineers