**Apache Storm Overview**

* Apache Storm is a distributed real-time big data-processing system. Storm is designed to process vast amount of data in a fault-tolerant and horizontal scalable method.

**Key features of Apache Storm**

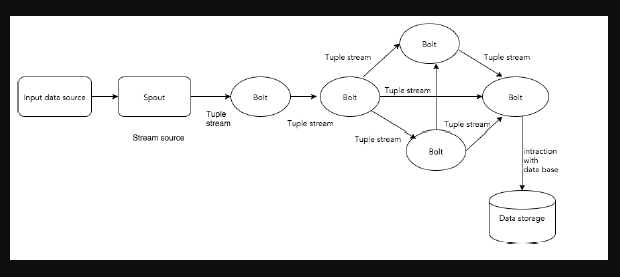
* Real-time stream processing
* Stateless
* Master/Slave architecture with Zookeeper based coordination. The master node is called as nimbus and slaves are supervisors.
* Storm topology runs until shutdown by the user or an unexpected unrecoverable failure.
* If nimbus / supervisor dies, restarting makes it continue from where it stopped, hence nothing gets affected.
* Storm is fault tolerant, flexible, reliable, and supports any programming language

**Use-Cases of Apache Storm**

Apache Storm is very famous for real-time big data stream processing. For this reason, most of the companies are using Storm as an integral part of their system. Some notable examples are as follows −

* **Twitter** − Twitter is using Apache Storm for its range of “Publisher Analytics products”. “Publisher Analytics Products” process each and every tweets and clicks in the Twitter Platform. Apache Storm is deeply integrated with Twitter infrastructure.
* **Wego** − Wego is a travel metasearch engine located in Singapore. Travel related data comes from many sources all over the world with different timing. Storm helps Wego to search real-time data, resolves concurrency issues and find the best match for the end-user.

**Storm Architectural Diagram:**



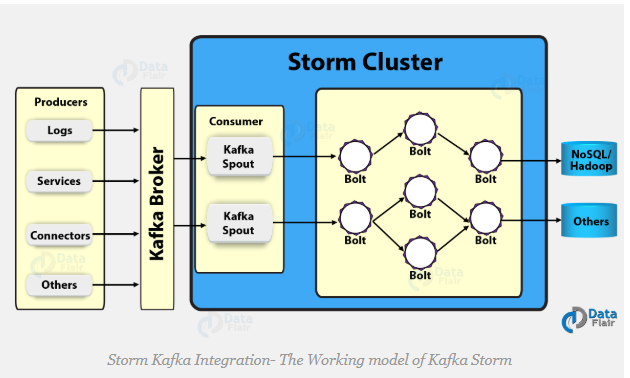
**Following are components of Storm:**

|  |  |
| --- | --- |
| Component | Description |
| Tuple | Tuple is the main data structure in Storm. It is a list of ordered elements. By default, a Tuple supports all data types. Generally, it is modelled as a set of comma separated values and passed to a Storm cluster. |
| Stream | Stream is an unordered sequence of tuples. |
| Spouts | Source of stream. Generally, Storm accepts input data from raw data sources like Twitter Streaming API, Apache Kafka queue, Kestrel queue, etc. Otherwise you can write spouts to read data from data sources. “ISpout" is the core interface for implementing spouts. Some of the specific interfaces are IRichSpout, BaseRichSpout, KafkaSpout, etc. |
| Bolts | Bolts are logical processing units. Spouts pass data to bolts and bolts process and produce a new output stream. Bolts can perform the operations of filtering, aggregation, joining, interacting with data sources and databases. Bolt receives data and emits to one or more bolts. “IBolt” is the core interface for implementing bolts. Some of the common interfaces are IRichBolt, IBasicBolt, etc. |

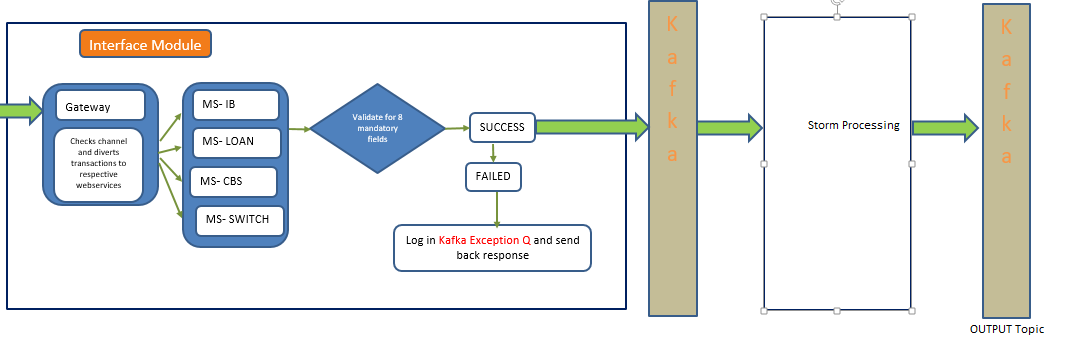
**Topology**

* Spouts and bolts are connected together and they form a topology. Real-time application logic is specified inside Storm topology. In simple words, a topology is a directed graph where vertices are computation and edges are stream of data.
* A simple topology starts with spouts. Spout emits the data to one or more bolts. Bolt represents a node in the topology having the smallest processing logic and the output of a bolt can be emitted into another bolt as input.

**Storm Kafka Integration**



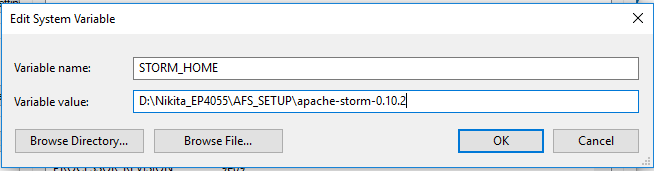
This block is equivalent to the storm processing unit in the below diagram.



Apache Storm set up path:

[**\\sezcpu063\Shared\Nikita Handover setup and documents\apache-storm-0.10.2**](file:///\\sezcpu063\Shared\Nikita%20Handover%20setup%20and%20documents\apache-storm-0.10.2)

**Set environment variable:**



**Following entries need to be added in Omnicongfig.properties files:**

**Note: Change IP address accordingly.**

1. topology=storm-kafka-topology
2. LOGGER-spout=LOGGER-spout
3. UPI-spout=UPI-spout
4. CBS-spout=CBS-spout
5. ATM-spout=ATM-spout
6. Loan-spout=Loan-spout
7. IB-spout=IB-spout
8. POS-spout=POS-spout
9. kafka.zookeeper=172.25.2.211:2181
10. kafa.topic=test1
11. kafka.zkRoot=/kafka
12. kafka.consumer.group=sample\_group
13. kafkaspout.count=1
14. KafkaConfig.forceFromStart=false
15. sink-type-bolt=sink-type
16. sinkbolt.count=1
17. solr-bolt=solr-bolt
18. solrbolt.count=1
19. solr.collection=collection2
20. solr.url=http://172.25.2.211:8983/solr/
21. solr.zookeeper.hosts=172.25.2.211:2181
22. nimbus.thriftport=6627
23. storm.zookeeperport=2181
24. kafka.broker0=172.25.2.211:9092
25. bootstrap.server=172.25.2.211:9092
26. TOPOLOGYDECIDER = true

List of java classes:

|  |  |
| --- | --- |
| **List of Java classes** | **Functionality** |
| AFSDispatcherServlet.java | This servlet is activated on load on start up in web.xml file. Inside init() method code to submit topology is written. |

**Simulator application set up**

**SVN URL**: [**http://sezsvn.infraseepz.lan/svn/FRMS/trunk/InfraSimulator**](http://sezsvn.infraseepz.lan/svn/FRMS/trunk/InfraSimulator)

**Sample tomcat is shared at:**

**\\sezcpu063\Shared\Nikita Handover setup and documents\SimulatorAppTC**

**You can check out code from this location and can make changes in following:**

1. Simulator.properties
   1. upload\_ws=http://**172.25.2.211**:6081/
   2. Note: Change IP accordingly
2. RequestorConfig.csv : Any new topology entries needs to be made in here.
3. UsecaseLog folder: topology specific transaction csv file needs to be placed here

List of tables involved:

|  |  |
| --- | --- |
| **Table Name** | **Functionality** |
| SECTORCHANNELALERTCODEMAP | New topology entries needs to be done using manual insert scripts Note: Verify all the parameters and then make inserts |
| TOPOLOGYMST | New topology entries needs to be done using manual insert scripts Note: Verify all the parameters and then make inserts |
| CEPSCANRESULT | Transaction ref id wise entries are saved in this table irrespective of fraud status |
| CFCSSAFSLEGACHKEYFIELDMAP | Here mapping of legacy fields with AFS specific keys is done |