Storm是一个分布式实时计算系统，它设计了一种对流和计算的抽象，概念比较简单，实际编程开发起来相对容易。
下面，简单介绍编程实践过程中需要理解的Storm中的几个概念：

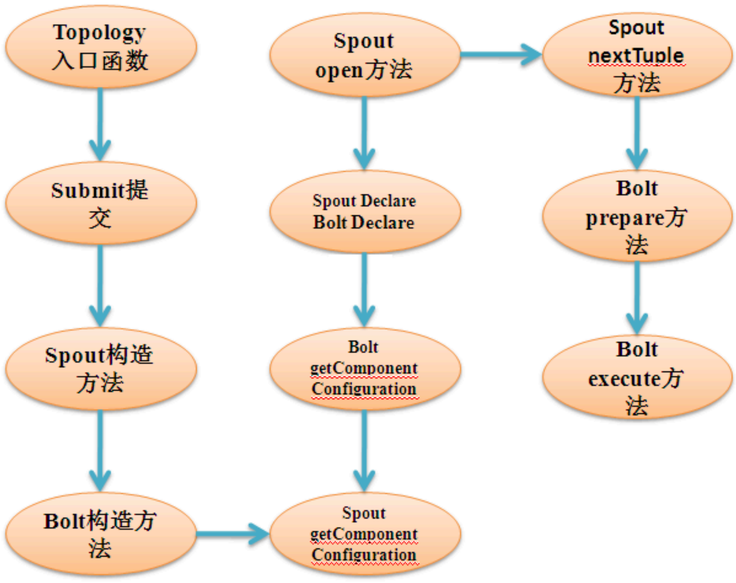
1. **Topology**
Storm中Topology的概念类似于Hadoop中的MapReduce Job，是一个用来编排、容纳一组计算逻辑组件（Spout、Bolt）的对象（Hadoop MapReduce中一个Job包含一组Map Task、Reduce Task），这一组计算组件可以按照DAG图的方式编排起来（通过选择Stream Groupings来控制数据流分发流向），从而组合成一个计算逻辑更加负责的对象，那就是Topology。一个Topology运行以后就不能停止，它会无限地运行下去，除非手动干预（显式执行bin/storm kill ）或意外故障（如停机、整个Storm集群挂掉）让它终止。

2. **Spout**
Storm中Spout是一个Topology的消息生产的源头，Spout应该是一个持续不断生产消息的组件，例如，它可以是一个Socket Server在监听外部Client连接并发送消息，可以是一个消息队列（MQ）的消费者、可以是用来接收Flume Agent的Sink所发送消息的服务，等等。Spout生产的消息在Storm中被抽象为Tuple，在整个Topology的多个计算组件之间都是根据需要抽象构建的Tuple消息来进行连接，从而形成流。

3. **Bolt**
Storm中消息的处理逻辑被封装到Bolt组件中，任何处理逻辑都可以在Bolt里面执行，处理过程和普通计算应用程序没什么区别，只是需要根据Storm的计算语义来合理设置一下组件之间消息流的声明、分发、连接即可。Bolt可以接收来自一个或多个Spout的Tuple消息，也可以来自多个其它Bolt的Tuple消息，也可能是Spout和其它Bolt组合发送的Tuple消息。

4. **Stream Grouping**
Storm中用来定义各个计算组件（Spout、Bolt）之间流的连接、分组、分发关系。
Storm定义了如下7种分发策略：
   - Shuffle Grouping（随机分组）
   - Fields Grouping（按字段分组）
   - All Grouping（广播分组）
   - Global Grouping（全局分组）
   - Non Grouping（不分组）
   - Direct Grouping（直接分组）
   - Local or Shuffle Grouping（本地/随机分组）

5. **Storm执行顺序**



6. Storm配置
在运行Topology之前，可以通过一些参数的配置来调节运行时的状态，参数的配置是通过Storm框架部署目录下的conf/storm.yaml文件来完成的。在此文件中可以配置运行时的Storm本地目录路径、运行时Worker的数目等。
在代码中，也可以设置Config的一些参数，但是优先级是不同的，不同位置配置Config参数的优先级顺序为：default.yaml < storm.yaml <Topology内部的configuration <内部组件的special configuration < 外部组件的special configuration
在storm.yaml中常用的几个选项为：

配置选项名称	配置选项作用
TOPOLOGY_MAX_TASK_PARALLELIS	每个Topology运行时最大的executor数目
TOPOLOGY_WORKERS	每个Topology运行时的worker的默认数目，若在代码中设置，则此选项值被覆盖
STORM_ZOOKEEPER_SERVERS	zookeeper集群的节点列表
STORM_LOCAL_DIR	Storm用于存储jar包和临时文件的本地存储目录
STORM_ZOOKEEPER_ROOT	Storm在zookeeper集群中的根目录，默认是“/”
UI_PORT	Storm集群的UI地址端口号，默认是8080
NIMBUS_HOST	Nimbus节点的host
SUPERVISOR_SLOTS_PORTS	Supervisor节点的worker占位槽，集群中的所有Topology共用这些槽位数，即使提交时设置了较大数值的槽位数，系统也会按照当前集群中实际剩余的槽位数来进行分配，当所有的槽位数都分配完时，新提交的Topology只能等待，系统会一直监测是否有空余的槽位空出来，如果有，就再次给新提交的Topology分配
SUPERVISOR_WORKER_START_TIMEOUT_SECS	Worker的超时时间，单位为秒，超时后，Storm认为当前worker进程死掉，会重新分配其运行着的task任务
DRPC_PORT	在使用drpc服务时，drpc server的服务器列表
DRPC_SERVERS	在使用drpc服务时，drpc server的服务端口