# *Capítulo 6*

# *Componentes de Hadoop*

**Al finalizar el capítulo, el alumno podrá:**

- Utilizar y crear Jobs MapReduce
- Utilizar Pig
- Utilizar Hive
- Utilizar Flume
- Utilizar Sqoop
- Utilizar Oozie

**Temas**

1. MapReduce
2. Pig y Hive
3. Flume y Sqoop
4. Oozie

# 1. MapReduce

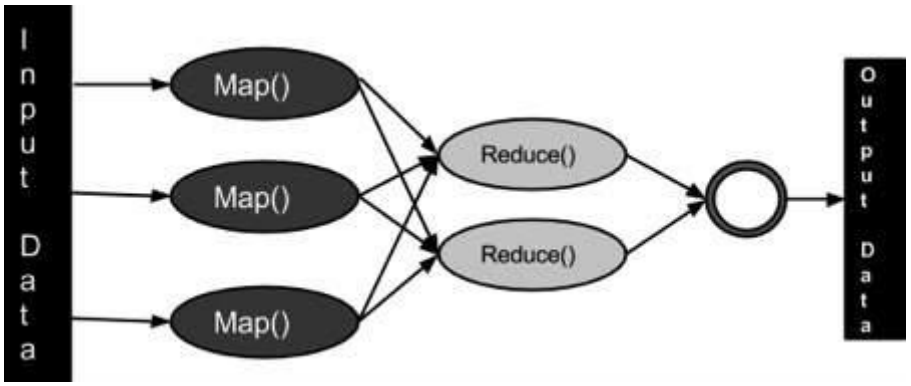
MapReduce es un marco con el que podemos escribir aplicaciones para procesar grandes cantidades de datos, en paralelo, en grandes grupos de hardware básico de una manera confiable.

**¿Qué es MapReduce?**
MapReduce es una técnica de procesamiento y un modelo de programa para computación distribuida basada en Java. El algoritmo MapReduce contiene dos tareas importantes, a saber, Map y Reduce. Map toma un conjunto de datos y los convierte en otro conjunto de datos, donde los elementos individuales se dividen en tuplas (pares clave / valor). En segundo lugar, reduzca la tarea, que toma la salida de un mapa como una entrada y combina esas tuplas de datos en un conjunto más pequeño de tuplas. Como lo implica la secuencia del nombre MapReduce, la tarea de reducción siempre se realiza después del trabajo del mapa.

La principal ventaja de MapReduce es que es fácil escalar el procesamiento de datos en múltiples nodos informáticos. Bajo el modelo MapReduce, las primitivas de procesamiento de datos se llaman Mappers y Reducers. Descomponer una aplicación de procesamiento de datos en Mappers y Reducers a veces no es trivial. Pero, una vez que escribimos una aplicación en el formulario MapReduce, escalar la aplicación para ejecutar cientos, miles o incluso decenas de miles de máquinas en un clúster es simplemente un cambio de configuración. Esta escalabilidad simple es lo que ha atraído a muchos programadores a utilizar el modelo MapReduce.

**El algoritmo**

- En general, el paradigma MapReduce se basa en enviar la computadora a donde residen los datos.
- El programa MapReduce se ejecuta en tres etapas, a saber, etapa de mapa, etapa de mezcla y reducción de etapa:
  - ✓ Etapa del Map: el trabajo del mapa o del mapeador es procesar los datos de entrada. En general, los datos de entrada están en forma de archivo o directorio y se almacenan en el sistema de archivos de Hadoop (HDFS). El archivo de entrada se pasa a la función del asignador línea por línea. El asignador procesa los datos y crea varios fragmentos pequeños de datos.
  - ✓ Etapa Reduce: esta etapa es la combinación de la etapa Shuffle y la etapa Reduce. El trabajo del Reducer es procesar los datos que provienen del mapeador. Después del procesamiento, produce un nuevo conjunto de resultados, que se almacenarán en el HDFS.
- Durante un trabajo de MapReduce, Hadoop envía las tareas de Asignar y Reducir a los servidores apropiados del clúster.
- El marco gestiona todos los detalles del paso de datos, como la emisión de tareas, la verificación de la finalización de tareas y la copia de datos en el clúster entre los nodos.
- La mayor parte de la computación se lleva a cabo en nodos con datos en discos locales que reducen el tráfico de la red.
- Después de completar las tareas dadas, el clúster recopila y reduce los datos para formar un resultado apropiado y lo envía de vuelta al servidor de Hadoop.




**Entradas y salidas (perspectiva de Java)**
El marco MapReduce opera en pares de <clave, valor>, es decir, el marco visualiza la entrada al trabajo como un conjunto de pares de <clave, valor> y produce un conjunto de pares de <clave, valor> como resultado del trabajo, concebiblemente de diferentes tipos.

La clave y las clases de valores deben estar en forma serializada por el marco y, por lo tanto, deben implementar la interfaz de escritura. Además, las clases clave deben implementar la interfaz Writable-Comparable para facilitar la clasificación por el framework. Tipos de entrada y salida de un trabajo de MapReduce: (Entrada) <k1, v1> -> mapa -> <k2, v2> -> reducir -> <k3, v3> (Salida).

| | **Input** | **Output** |
|---|---|---|
| **Map** | <k1, v1> | list (<k2, v2>) |
| **Reduce** | <k2, list(v2)> | list (<k3, v3>) |

**Terminología**

- PayLoad: las aplicaciones implementan las funciones Mapa y Reducir y forman el núcleo del trabajo.
- Mapper: Mapper asigna los pares clave / valor de entrada a un conjunto de par clave / valor intermedio.
- NamedNode: nodo que gestiona el Sistema de archivos distribuidos de Hadoop (HDFS).
- DataNode: nodo donde los datos se presentan por adelantado antes de que tenga lugar cualquier procesamiento.
- MasterNode - Nodo donde se ejecuta JobTracker y que acepta solicitudes de trabajo de los clientes.
- SlaveNode - Nodo donde se ejecuta el programa Map and Reduce.
- JobTracker: programa trabajos y hace un seguimiento de los trabajos de asignación al Task Tracker.
- Task Tracker: realiza un seguimiento de la tarea e informa el estado a JobTracker.

- Job: un programa es una ejecución de un Mapper y un Reducer en un conjunto de datos.

- Task: una ejecución de un Mapper o un Reducer en una porción de datos.
- Task Attempt: una instancia particular de un intento de ejecutar una tarea en un SlaveNode.

# 2. Pig y Hive

**Apache Pig:**

**¿Qué es Apache Pig?**

Apache Pig es una abstracción sobre MapReduce. Es una herramienta / plataforma que se utiliza para analizar conjuntos más grandes de datos que los representan como flujos de datos. Pig generalmente se usa con Hadoop; podemos realizar todas las operaciones de manipulación de datos en Hadoop usando Apache Pig.

Para escribir programas de análisis de datos, Pig proporciona un lenguaje de alto nivel conocido como Pig Latin. Este lenguaje proporciona varios operadores mediante los cuales los programadores pueden desarrollar sus propias funciones para leer, escribir y procesar datos.

Para analizar datos usando Apache Pig, los programadores necesitan escribir scripts usando el lenguaje Pig Latin. Todos estos scripts se convierten internamente en tareas de mapa y reducción. Apache Pig tiene un componente conocido como Pig Engine que acepta los scripts de Pig Latin como entrada y convierte esos scripts en trabajos de MapReduce.

**¿Por qué necesitamos Apache Pig?**

Los programadores que no son tan buenos en Java normalmente solían tener dificultades para trabajar con Hadoop, especialmente al realizar cualquier tarea de MapReduce. Apache Pig es una gran ayuda para todos esos programadores.

Usando Pig Latin, los programadores pueden realizar tareas de MapReduce fácilmente sin tener que escribir códigos complejos en Java.

Apache Pig utiliza un enfoque de consultas múltiples, lo que reduce la longitud de los códigos. Por ejemplo, una operación que requeriría que escriba 200 líneas de código en Java se puede hacer fácilmente escribiendo tan solo 10 líneas de código en Apache Pig. En última instancia, Apache Pig reduce el tiempo de desarrollo en casi 16 veces.

Pig Latin es un lenguaje similar a SQL y es fácil aprender Apache Pig cuando está familiarizado con SQL.

Apache Pig proporciona muchos operadores integrados para admitir operaciones de datos como uniones, filtros, pedidos, etc. Además, también proporciona tipos de datos anidados como tuplas, bolsas y mapas que faltan en MapReduce.

**Características de Pig**

Apache Pig viene con las siguientes características:

- Amplio conjunto de operadores: proporciona muchos operadores para realizar operaciones como join, sort, filer, etc.

- Facilidad de programación: Pig Latin es similar a SQL y es fácil escribir un script de Pig si eres bueno en SQL.

- Oportunidades de optimización: las tareas en Apache Pig optimizan su ejecución de forma automática, por lo que los programadores deben centrarse únicamente en la semántica del lenguaje.

- Extensibilidad: utilizando los operadores existentes, los usuarios pueden desarrollar sus propias funciones para leer, procesar y escribir datos.

- UDF's - Pig proporciona la posibilidad de crear funciones definidas por el usuario en otros lenguajes de programación como Java e invocarlos o incrustarlos en Pig Scripts.

- Maneja todo tipo de datos: Apache Pig analiza todo tipo de datos, tanto estructurados como no estructurados. Almacena los resultados en HDFS.

**Aplicaciones de Apache Pig**

Apache Pig generalmente es utilizado por los científicos de datos para realizar tareas que implican procesamiento ad-hoc y prototipado rápido.
Apache Pig es usado:

- Para procesar grandes fuentes de datos, como registros web.
- Para realizar el procesamiento de datos para plataformas de búsqueda.
- Para procesar cargas de datos sensibles al tiempo.

**Historia de Apache Pig**

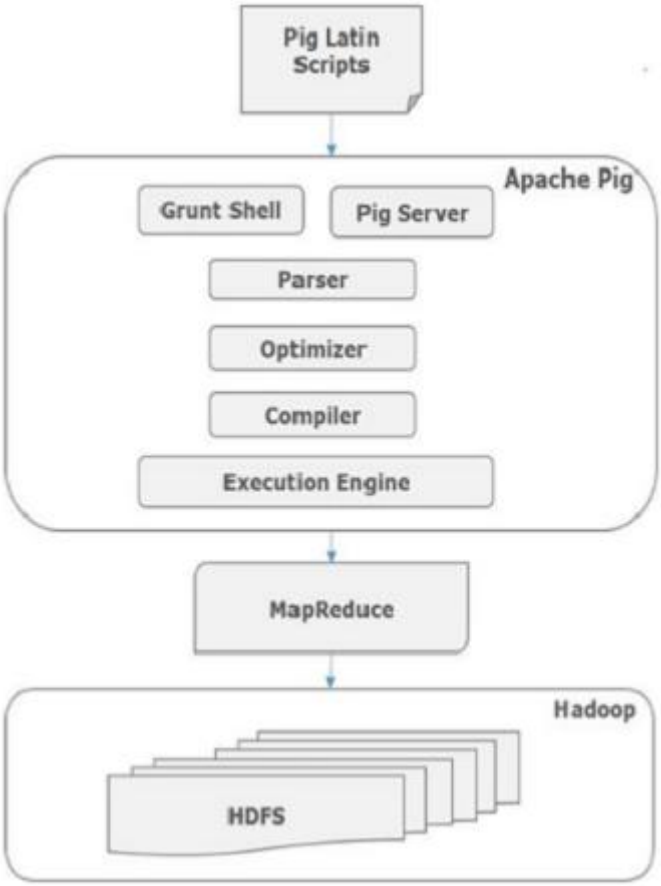
En 2006, Apache Pig se desarrolló como un proyecto de investigación en Yahoo, especialmente para crear y ejecutar trabajos de MapReduce en cada conjunto de datos. En 2007, Apache Pig fue abierto a través de la incubadora Apache. En 2008, salió el primer lanzamiento de Apache Pig. En 2010, Apache Pig se graduó como un proyecto de alto nivel de Apache.

**Arquitectura de Apache Pig**

El lenguaje utilizado para analizar datos en Hadoop usando Pig se conoce como Pig Latin. Es un lenguaje de procesamiento de datos de alto nivel que proporciona un amplio conjunto de tipos de datos y operadores para realizar diversas operaciones en los datos.

Para realizar una tarea en particular Los programadores que usan Pig, los programadores necesitan escribir un script Pig utilizando el lenguaje Pig Latin, y ejecutarlos usando cualquiera de los mecanismos de ejecución (Grunt Shell, UDFs, Embedded). Después de la ejecución, estos scripts pasarán por una serie de transformaciones aplicadas por Pig Framework, para producir el resultado deseado.

Internamente, Apache Pig convierte estos scripts en una serie de trabajos de MapReduce y, por lo tanto, facilita el trabajo del programador. La arquitectura de Apache Pig se muestra a continuación.




**Componentes de Apache Pig**

Como se muestra en la figura, hay varios componentes en el marco de Apache Pig. Echemos un vistazo a los principales componentes.

**Analizador**
Inicialmente, los Scripts Pig son manejados por el Analizador. Comprueba la sintaxis de la secuencia de comandos, realiza verificación de tipos y otras comprobaciones varias. La salida del analizador será un DAG (gráfico acíclico dirigido), que representa las sentencias Pig Latin y los operadores lógicos.

En el DAG, los operadores lógicos del script se representan como los nodos y los flujos de datos se representan como bordes.

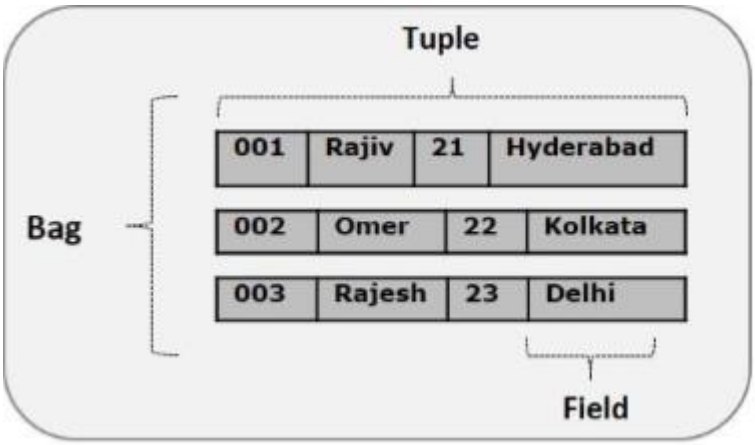
**Optimizador**
El plan lógico (DAG) se pasa al optimizador lógico, que lleva a cabo las optimizaciones lógicas, como proyección y pushdown.

**Compilador**
El compilador compila el plan lógico optimizado en una serie de trabajos de MapReduce.

**Motor de ejecución**
Finalmente, los trabajos de MapReduce se envían a Hadoop en un orden ordenado. Finalmente, estos trabajos de MapReduce se ejecutan en Hadoop y producen los resultados deseados.

**Modelo de datos de Pig Latin**

El modelo de datos de Pig Latin está completamente anidado y permite tipos de datos complejos no atómicos como el mapa y la tupla. A continuación se muestra la representación diagramática del modelo de datos de Pig Latin.




**Modelo de datos**

**Átomo**

Cualquier valor individual en Pig Latin, independientemente de sus datos, se conoce como un átomo. Se almacena como una cadena y se puede utilizar como cadena y número. int, long, float, double, chararray y bytearray son los valores atómicos de Pig. Una pieza de datos o un valor atómico simple se conoce como un campo.

Ejemplo: 'raja' o '30'

**Tuple**

Un registro formado por un conjunto ordenado de campos se conoce como tupla, los campos pueden ser de cualquier tipo. Una tupla es similar a una fila en una tabla de RDBMS.

Ejemplo - (Raja, 30)

**Bag**

Un Bag es un conjunto desordenado de tuplas. En otras palabras, una colección de tuplas (no únicas) se conoce como un Bag. Cada tupla puede tener cualquier cantidad de campos (esquema flexible). Una Bag está representada por '{}'. Es similar a una tabla en RDBMS, pero a diferencia de una tabla en RDBMS, no es necesario que cada tupla contenga el mismo número de campos o que los campos en la misma posición (columna) tengan el mismo tipo.

Ejemplo - {(Raja, 30), (Mohammad, 45)}

Una Bag puede ser un campo en una relación; en ese contexto, se conoce como Bag interior.

Ejemplo - {Raja, 30, {9848022338, raja@gmail.com,}}

**Map**

Un Map (o mapa de datos) es un conjunto de pares clave-valor. La clave debe ser de tipo chararray y debe ser única. El valor puede ser de cualquier tipo. Está representado por '[]'

Ejemplo - [nombre # Raja, edad # 30]

**Relation**
Una Relation es una bag de tuples. Las relaciones en Pig Latin no están ordenadas (no hay garantía de que las tuples se procesen en un orden particular).

**Apache Pig - Instalación**

**Requerimientos:**

- Hadoop 2.X - http://hadoop.apache.org/common/releases.html (Puede ejecutar Pig con diferentes versiones de Hadoop configurando HADOOP_HOME para que apunte al directorio donde ha instalado Hadoop. Si no configura HADOOP_HOME , por defecto Pig se ejecutará con la versión incorporada, actualmente Hadoop 2.7.3.)
- Java 1.7 - http://java.sun.com/javase/downloads/index.jsp (establezca JAVA_HOME en la raíz de su instalación de Java)

**Opcional:**

- Python 2.7 - https://www.python.org (al usar las UDF de Streaming Python)
- Ant 1.8 - http://ant.apache.org/ (para compilaciones)

**Download Apache Pig**

En primer lugar, descargue la última versión de Apache Pig desde el siguiente sitio web: https://pig.apache.org/

**Apache Hive:**

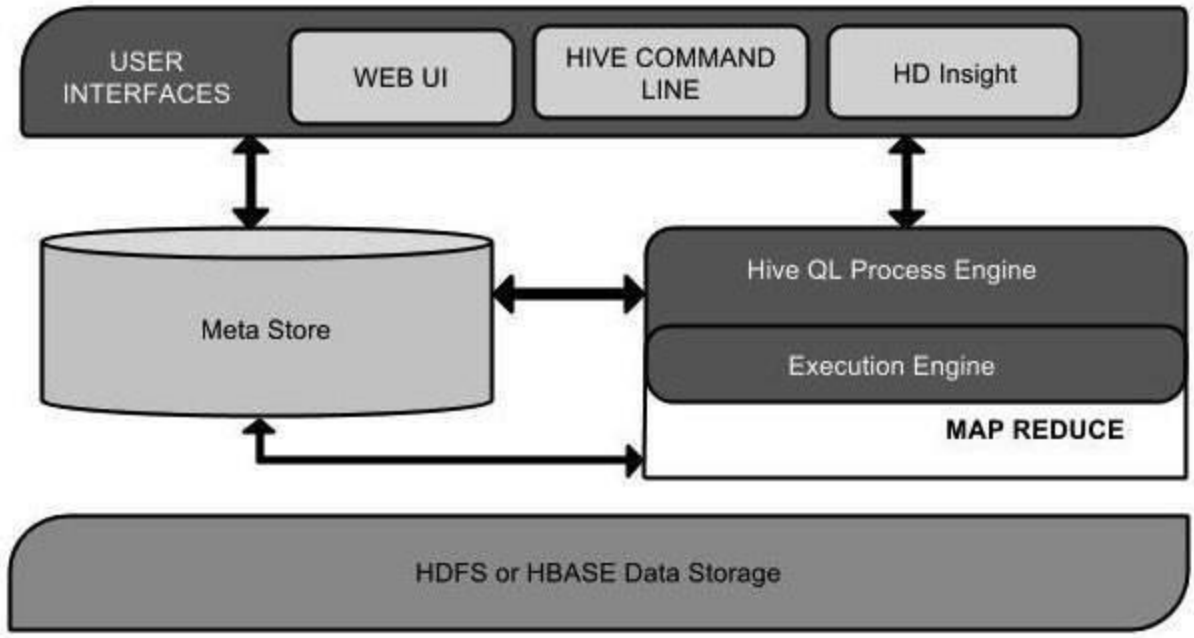
**¿Qué es Hive?**
Hive es una herramienta de infraestructura de almacenamiento de datos para procesar datos estructurados en Hadoop.
Inicialmente Hive fue desarrollado por Facebook, más tarde Apache Software Foundation lo desarrolló y desarrolló como fuente abierta bajo el nombre de Apache Hive. Es utilizado por diferentes compañías. Por ejemplo, Amazon lo usa en Amazon Elastic MapReduce.

**Hive no es:**
- Una base de datos relacional
- Un diseño para procesamiento de transacciones en línea (OLTP)
- Un lenguaje para consultas en tiempo real y actualizaciones a nivel de fila

**Características de Hive:**
- Almacena el esquema en una base de datos y procesa datos en HDFS.
- Está diseñado para OLAP.
- Proporciona lenguaje de tipo SQL para consultas llamado HiveQL o HQL.
- Es familiar, rápido, escalable y extensible.

**Arquitectura de Hive:**

El siguiente diagrama de componentes representa la arquitectura de Hive:



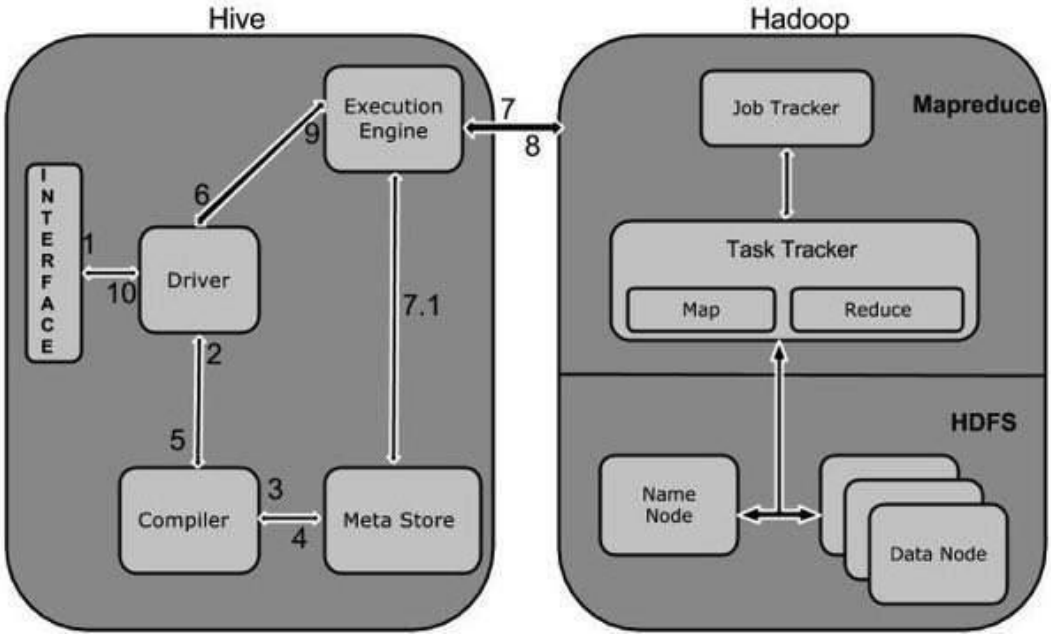

Este diagrama de componentes contiene diferentes unidades. La siguiente tabla describe cada unidad:

| Nombre de Unidad | Operación |
|---|---|
| User Interface | Hive es un software de infraestructura de almacenamiento de datos que puede crear interacción entre el usuario y HDFS. Las interfaces de usuario compatibles con Hive son Hive Web UI, Hive command line y Hive |

| | |
|---|---|
| | HD Insight (en Windows server). |
| Meta Store | Hive elige los servidores de bases de datos respectivos para almacenar el esquema o Metadatos de tablas, bases de datos, columnas en una tabla, sus tipos de datos y mapeo HDFS. |
| HiveQL Process Engine | HiveQL es similar a SQL para consultar información de esquema en el Metastore. Es uno de los reemplazos del enfoque tradicional para el programa MapReduce. En lugar de escribir el programa MapReduce en Java, podemos escribir una consulta para el trabajo de MapReduce y procesarlo. |
| Execution Engine | La parte de conjunción de HiveQL process Engine y MapReduce es Hive Execution Engine. El motor de ejecución procesa la consulta y genera resultados de la misma manera que los resultados de MapReduce. Utiliza el sabor de MapReduce. |
| HDFS or HBASE | El sistema de archivos distribuidos de Hadoop o HBASE son las técnicas de almacenamiento de datos para almacenar datos en el sistema de archivos. |

**Trabajando con Hive:**

El siguiente diagrama muestra el flujo de trabajo entre Hive y Hadoop.




La siguiente tabla define cómo Hive interactúa con el marco de Hadoop:

| Numero de Paso | Operación |
|---|---|
| **1.Execute Query** | La interfaz de Hive como Command Line o Web UI envía una consulta al controlador (cualquier controlador de base de datos |

| | |
|---|---|
| | como JDBC, ODBC, etc.) para ejecutar. |
| **2.Get Plan** | El controlador toma la ayuda del compilador de consultas que analiza la consulta para verificar la sintaxis y el plan de consulta o el requisito de consulta. |
| **3.Get Metadata** | El compilador envía una solicitud de metadatos al Metastore (cualquier base de datos). |
| **4.Send Metada** | Metastore envía metadatos como respuesta al compilador. |
| **5.Send Plan** | El compilador verifica el requisito y vuelve a enviar el plan al controlador. Hasta aquí, el análisis sintáctico y la compilación de una consulta están completos. |
| **6.Execute Plan** | El controlador envía el plan de ejecución al motor de ejecución. |
| **7.Execute Job** | Internamente, el proceso del trabajo de ejecución es un trabajo de MapReduce. El motor de ejecución envía el trabajo a JobTracker, que está en el nodo Nombre y asigna este trabajo a TaskTracker, que está en el nodo Datos. Aquí, la consulta ejecuta el trabajo MapReduce. |
| **7.1 Metadata Ops** | Mientras tanto, en ejecución, el motor de ejecución puede ejecutar operaciones de metadatos con Metastore. |
| **8.Fetch Result** | El motor de ejecución recibe los resultados de los nodos Datos. |
| **9.Send Results** | El motor de ejecución envía esos valores resultantes al controlador. |
| **10.Send Results** | El controlador envía los resultados a Hive Interfaces. |

**Requerimientos:**

- Java 1.7 Nota: Las versiones de Hive 1.2 en adelante requieren Java 1.7 o posterior. Las versiones de Hive 0.14 a 1.1 también funcionan con Java 1.6. Se recomienda encarecidamente a los usuarios que comiencen a pasar a Java 1.8 (ver HIVE-8607).
- Hadoop 2.x (preferido), 1.x (no compatible con Hive 2.0.0 en adelante). Las versiones de Hive hasta 0.13 también admiten Hadoop 0.20.x, 0.23.x.
- Hive se usa comúnmente en entornos Linux y Windows de producción. Mac es un entorno de desarrollo de uso común. Las instrucciones en este documento son aplicables a Linux y Mac. Usarlo en Windows requeriría pasos ligeramente diferentes.

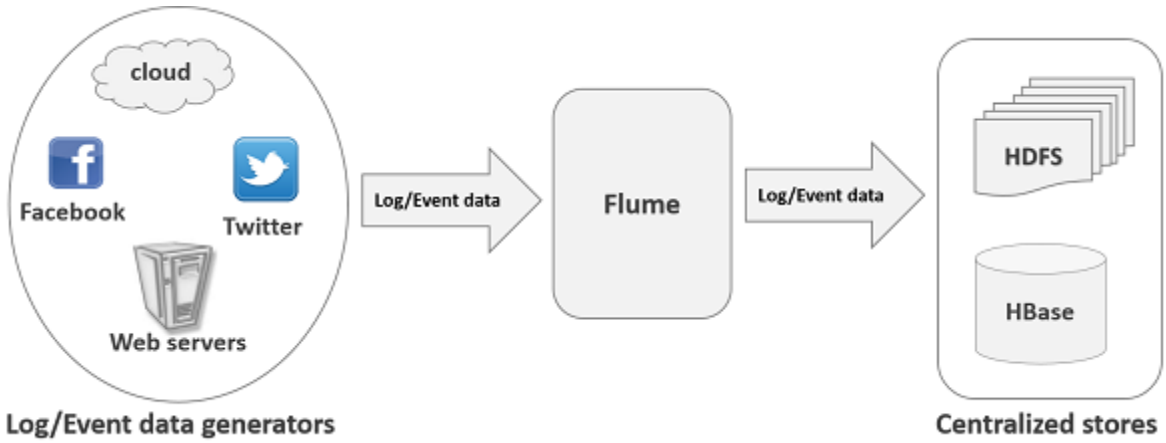
**Instalación de Hive:**

Los siguientes pasos son necesarios para instalar Hive en su sistema. Ir a la documentación oficial de Hive.

http://hive.apache.org/

# 3. Flume y Sqoop

**Apache Flume**
**¿Qué es Flume?**
Apache Flume es un mecanismo de ingestión de herramienta/servicio/datos para recolectar agregados y transportar grandes cantidades de datos de transmisión como archivos de registro, eventos (etc …) de diversas fuentes a un almacén de datos centralizado.
Flume es una herramienta altamente confiable, distribuida y configurable. Está diseñado principalmente para copiar datos de transmisión (datos de registro) desde varios servidores web a HDFS.




**Aplicaciones de Flume**
Supongamos que una aplicación web de comercio electrónico desea analizar el comportamiento del cliente de una región en particular. Para hacerlo, tendrían que mover los datos de registro disponibles a Hadoop para su análisis. Aquí, Apache Flume viene a nuestro rescate.
Flume se usa para mover los datos de registro generados por los servidores de aplicaciones a HDFS a una velocidad mayor.

**Ventajas de Flume**
Aquí están las ventajas de usar Flume:

- Usando Apache Flume podemos almacenar los datos en cualquiera de las tiendas centralizadas (HBase, HDFS).
- Cuando la tasa de datos entrantes excede la velocidad a la que los datos se pueden escribir en el destino, Flume actúa como un mediador entre los productores de datos y las tiendas centralizadas y proporciona un flujo constante de datos entre ellos.
- Flume proporciona la característica de enrutamiento contextual.
- Las transacciones en Flume se basan en canales donde se mantienen dos transacciones (un emisor y un receptor) para cada mensaje. Garantiza la entrega confiable de mensajes.
- Flume es confiable, tolerante a fallas, escalable, manejable y personalizable.

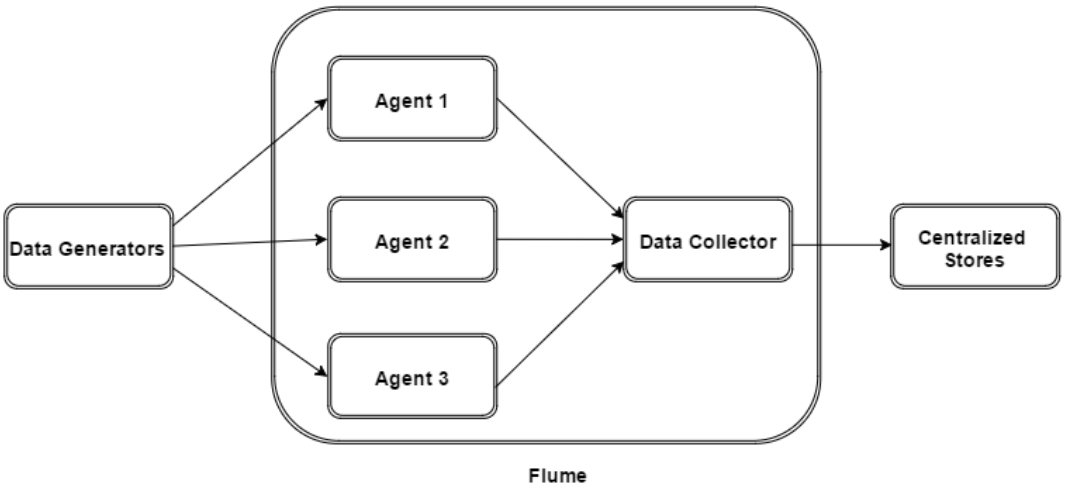
**Características de Flume**
Algunas de las características notables de Flume son las siguientes:

- Flume ingiere datos de registro de múltiples servidores web en una tienda centralizada (HDFS, HBase) de manera eficiente.
- Usando Flume, podemos obtener los datos de múltiples servidores de inmediato en Hadoop.

- Junto con los archivos de registro, Flume también se utiliza para importar grandes volúmenes de datos de eventos producidos por sitios de redes sociales como Facebook y Twitter, y sitios web de comercio electrónico como Amazon y Flipkart.
- Flume admite un gran conjunto de fuentes y tipos de destinos.
- Flume admite flujos de varios saltos, flujos de abanico de ventilador en abanico, enrutamiento contextual, etc.
- El flujo se puede escalar horizontalmente.

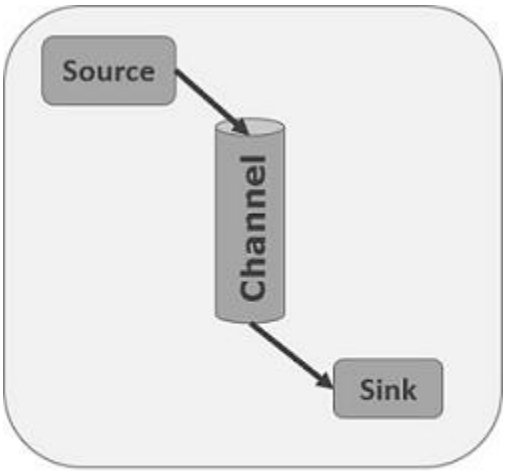
**Apache Flume - Arquitectura**

La siguiente ilustración muestra la arquitectura básica de Flume. Como se muestra en la ilustración, los generadores de datos (como Facebook, Twitter) generan datos que son recopilados por agentes de Flume individuales que se ejecutan en ellos. A partir de entonces, un recopilador de datos (que también es un agente) recopila los datos de los agentes que se agregan y envían a un almacén centralizado como HDFS o HBase.




- **Flume Event**
  Un evento es la unidad básica de los datos transportados dentro de Flume. Contiene una carga útil de una matriz de bytes que se transportará desde el origen hasta el destino acompañado de encabezados opcionales. Un evento típico de Flume tendría la siguiente estructura:

- **Flume Agent**
  Un agente es un proceso de daemon independiente (JVM) en Flume. Recibe los datos (eventos) de clientes u otros agentes y los reenvía a su próximo destino (receptor o agente). Flume puede tener más de un agente. El siguiente diagrama representa un agente de Flume




Flume Agent

Como se muestra en el diagrama, un agente de Flume contiene tres componentes principales, a saber, source, channel y sink.

- **Source**
  Una fuente es el componente de un Agente que recibe datos de los generadores de datos y los transfiere a uno o más canales en forma de eventos Flume. Apache Flume admite varios tipos de fuentes y cada fuente recibe eventos de un generador de datos específico. Ejemplo: fuente de Avro, fuente de ahorro, fuente de Twitter 1%, etc.

- **Channel**
  Un canal es una tienda transitoria que recibe los eventos de la fuente y los almacena en búfer hasta que son consumidos por los sumideros. Actúa como un puente entre las fuentes y los sumideros.
  Estos canales son totalmente transaccionales y pueden funcionar con cualquier cantidad de fuentes y receptores. Ejemplo: canal JDBC, canal del sistema de archivos, canal de memoria, etc.

- **Sink**
  Un Sink almacena los datos en tiendas centralizadas como HBase y HDFS. Consume los datos (eventos) de los canales y los entrega al destino. El destino del sink podría ser otro agente o las tiendas centrales. Ejemplo – HDFS Sink

Nota: un agente de canalización puede tener múltiples fuentes, sinks y channels.

**Componentes adicionales del Agent Flume**
Lo que hemos discutido anteriormente son los componentes primitivos del agente. Además de esto, tenemos algunos componentes más que juegan un papel vital en la transferencia de eventos del generador de datos a las tiendas centralizadas.

- **Interceptors**
  Los interceptores se utilizan para alterar / inspeccionar eventos de canalización que se transfieren entre la fuente y el canal.

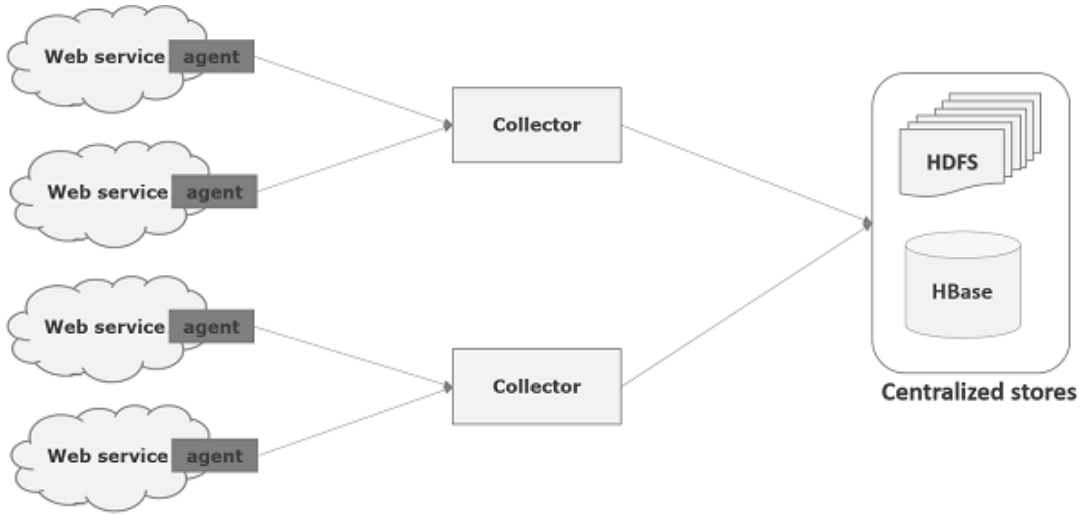
- **Channel Selectors**
  Estos se utilizan para determinar qué canal se debe optar por transferir los datos en el caso de canales múltiples. Hay dos tipos de selectores de canales:

- **Default cannel selectors**: estos también se conocen como selectores de canales de replicación y replican todos los eventos en cada canal.

- **Multiplexing channel selectors:** Estos deciden el canal para enviar un evento basado en la dirección en el encabezado de ese evento.

- **Sink Processors:**
  Estos se utilizan para invocar un sink particular del grupo seleccionado de sinks. Estos se utilizan para crear rutas de conmutación por error para sus receptores o para equilibrar los eventos de carga en múltiples receptores de un canal.

**Apache Flume - DataFlow**
Flume es un marco que se usa para mover datos de registro a HDFS. En general, los eventos y los datos de registro los generan los servidores de registro y estos servidores tienen agentes de Flume ejecutándose en ellos. Estos agentes reciben los datos de los generadores de datos.
Los datos en estos agentes serán recolectados por un nodo intermedio conocido como Collector. Al igual que los agentes, puede haber varios coleccionistas en Flume.
Finalmente, los datos de todos estos recopiladores se agregarán y enviarán a una tienda centralizada como HBase o HDFS. El siguiente diagrama explica el flujo de datos en Flume.




- **Multi-hop Flow**
  Dentro de Flume, puede haber múltiples agentes y antes de llegar al destino final, un evento puede viajar a través de más de un agente. Esto se conoce como flujo multi-hop.

- **Fan-out Flow**
  El flujo de datos de una fuente a múltiples canales se conoce como fan-out flow de ventilador. Es de dos tipos:
  - Replicación: el flujo de datos donde los datos se replicarán en todos los canales configurados.
  - Multiplexación: el flujo de datos donde los datos se enviarán a un canal seleccionado que se menciona en el encabezado del evento.

- **Fan-in Flow**
  El flujo de datos en el que se transferirán los datos de muchas fuentes a un canal se conoce como flujo de entrada.

**Manejo de fallas**
En Flume, para cada evento, se llevan a cabo dos transacciones: una en el emisor y otra en el receptor. El emisor envía eventos al receptor. Poco después de recibir los datos, el receptor confirma su propia transacción y envía una señal de "recepción" al remitente. Después de recibir la señal, el emisor confirma su transacción. (El emisor no realizará su transacción hasta que reciba una señal del receptor).

**Instalar Flume**

En primer lugar, descargue la última versión del software Apache Flume del sitio web

https://flume.apache.org/.

**Apache Sqoop**

**¿Qué es Sqoop?**

El sistema de gestión de aplicaciones tradicional, es decir, la interacción de aplicaciones con bases de datos relacionales que utilizan RDBMS, es una de las fuentes que generan Big Data. Tales Big Data, generados por RDBMS, se almacenan en servidores de bases de datos relacionales en la estructura de la base de datos relacional.

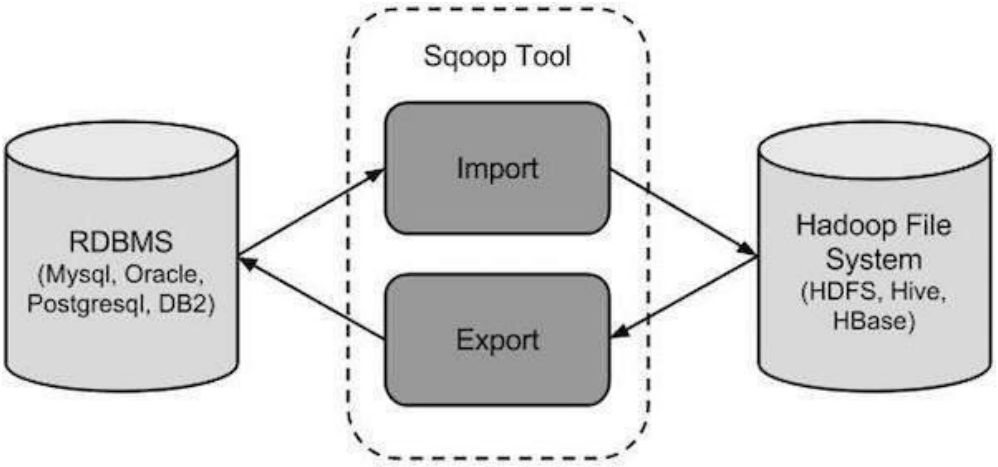
Cuando los almacenamientos Big Data y analizadores como MapReduce, Hive, HBase, Cassandra, Pig, etc. del ecosistema Hadoop entraron en escena, requirieron una herramienta para interactuar con los servidores de bases de datos relacionales para importar y exportar los Big Data que residen en ellos. Aquí, Sqoop ocupa un lugar en el ecosistema de Hadoop para proporcionar una interacción viable entre el servidor de base de datos relacional y el HDFS de Hadoop.

**Sqoop - "SQL a Hadoop y Hadoop a SQL"**

Sqoop es una herramienta diseñada para transferir datos entre Hadoop y servidores de bases de datos relacionales. Se utiliza para importar datos de bases de datos relacionales como MySQL, Oracle a Hadoop HDFS y exportar desde el sistema de archivos Hadoop a bases de datos relacionales. Lo proporciona la Apache Software Foundation.

**¿Cómo funciona Sqoop?**

La siguiente imagen describe el flujo de trabajo de Sqoop.




**Sqoop Import**

La herramienta de importación importa tablas individuales desde RDBMS a HDFS. Cada fila en una tabla se trata como un registro en HDFS. Todos los registros se almacenan como datos de texto en archivos de texto o como datos binarios en archivos Avro y Sequence.

**Export Sqoop**

La herramienta de exportación exporta un conjunto de archivos de HDFS a un RDBMS. Los archivos proporcionados como entrada a Sqoop contienen registros, que se llaman filas en la tabla. Esos son leídos y analizados en un conjunto de registros y delimitados con delimitador especificado por el usuario.

**Uso Básico**

Con Sqoop, puede importar datos desde un sistema de base de datos relacional o un mainframe a HDFS. La entrada al proceso de importación es una tabla de base de

datos o conjuntos de datos de mainframe. Para las bases de datos, Sqoop leerá la tabla fila por fila en HDFS. Para los conjuntos de datos de mainframe, Sqoop leerá los registros de cada conjunto de datos de mainframe en HDFS. El resultado de este proceso de importación es un conjunto de archivos que contiene una copia de la tabla o datasets importados. El proceso de importación se realiza en paralelo. Por esta razón, la salida estará en múltiples archivos. Estos archivos pueden ser archivos de texto delimitados (por ejemplo, con comas o pestañas que separan cada campo) o Avro o SequenceFiles binarios que contienen datos de registro serializados.

Un subproducto del proceso de importación es una clase Java generada que puede encapsular una fila de la tabla importada. Esta clase es utilizada durante el proceso de importación por Sqoop. También se le proporciona el código fuente de Java para esta clase, para su uso en el procesamiento posterior de MapReduce de los datos. Esta clase puede serializar y deserializar datos hacia y desde el formato SequenceFile. También puede analizar la forma de texto delimitado de un registro. Estas habilidades le permiten desarrollar rápidamente las aplicaciones de MapReduce que usan los registros almacenados en HDFS en su canal de procesamiento. También puede analizar los datos de registro delimitados usted mismo, utilizando cualquier otra herramienta que prefiera.

Después de manipular los registros importados (por ejemplo, con MapReduce o Hive) puede tener un conjunto de datos de resultados que luego puede exportar a la base de datos relacional. El proceso de exportación de Sqoop leerá un conjunto de archivos de texto delimitados de HDFS en paralelo, los analizará en registros y los insertará como nuevas filas en una tabla de base de datos de destino, para consumo de aplicaciones externas o usuarios.

Sqoop incluye algunos otros comandos que le permiten inspeccionar la base de datos con la que está trabajando. Por ejemplo, puede enumerar los esquemas de base de datos disponibles (con la herramienta sqoop-list-databases) y las tablas dentro de un esquema (con la herramienta sqoop-list-tables). Sqoop también incluye un shell de ejecución de SQL primitivo (la herramienta sqoop-eval).

La mayoría de los aspectos de importación, generación de código y procesos de exportación se pueden personalizar. Para las bases de datos, puede controlar el rango de fila específico o las columnas importadas. Puede especificar delimitadores particulares y caracteres de escape para la representación de datos basada en archivos, así como también el formato de archivo utilizado. También puede controlar los nombres de clase o paquete utilizados en el código generado. Las secciones posteriores de este documento explican cómo especificar estos y otros argumentos a Sqoop.

**Instalar Sqoop**

En primer lugar, descargue la última versión del software Apache Sqoop del sitio web

http://sqoop.apache.org/

# 4. Oozie

¿Qué es Apache Oozie?
Apache Oozie es un sistema de programación para ejecutar y administrar trabajos de Hadoop en un entorno distribuido. Permite combinar múltiples trabajos complejos para ejecutarse en un orden secuencial para lograr una tarea más grande. Dentro de una secuencia de tareas, también se pueden programar dos o más trabajos para que se ejecuten en paralelo entre sí.

Una de las principales ventajas de Oozie es que está estrechamente integrada con la pila Hadoop que admite varios trabajos de Hadoop como Hive, Pig, Sqoop y también trabajos específicos del sistema como Java y Shell.

Oozie es una aplicación web Java de código abierto disponible bajo la licencia Apache 2.0. Es responsable de activar las acciones del flujo de trabajo, que a su vez usa el motor de ejecución de Hadoop para ejecutar la tarea. Por lo tanto, Oozie puede aprovechar la maquinaria Hadoop existente para el equilibrio de carga, el fail-over, etc.

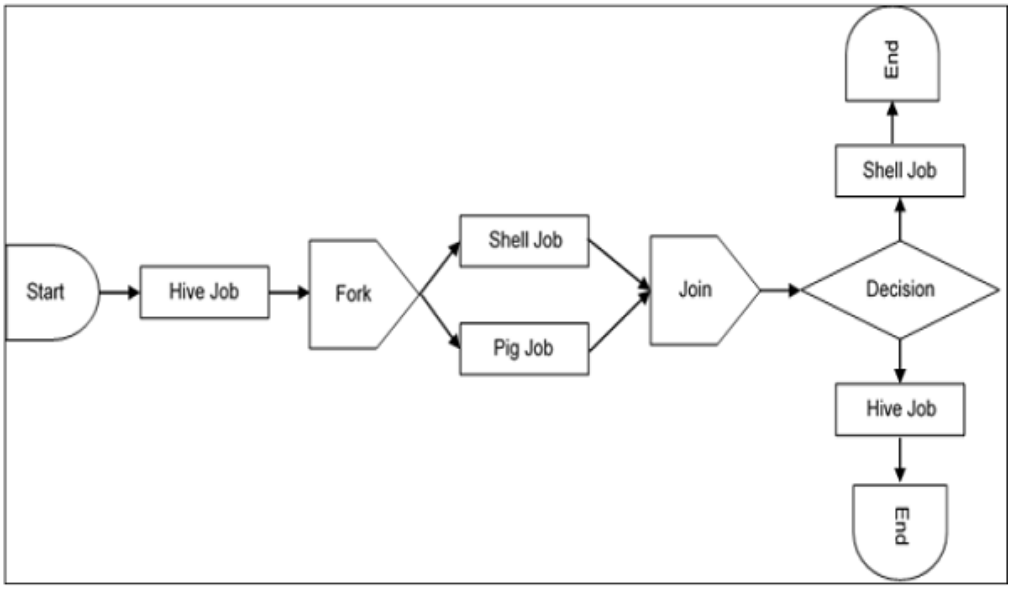
Oozie detecta la finalización de tareas mediante la devolución de llamada y el sondeo. Cuando Oozie inicia una tarea, proporciona una URL HTTP de devolución de llamada única para la tarea y notifica esa URL cuando se completa. Si la tarea no puede invocar la URL de devolución de llamada, Oozie puede sondear la tarea para completarla.

Seguir estos tres tipos de trabajos es común en Oozie:

- Oozie Workflow Jobs: se representan como Directed Acyclic Graphs (DAG) para especificar una secuencia de acciones que se ejecutarán.
- Oozie Coordinator Jobs: consisten en trabajos de flujo de trabajo desencadenados por el tiempo y la disponibilidad de datos.
- Oozie Bundle: se puede denominar paquete de múltiples trabajos de coordinador y flujo de trabajo.

Examinaremos cada uno de estos en detalle en los siguientes capítulos.

Un flujo de trabajo de muestra con Controles (Start, Decision, Fork, Join and End) y Acciones (Hive, Shell, Pig) se verá como el siguiente diagrama:




Flujo de trabajo de muestra
El flujo de trabajo siempre comenzará con una etiqueta de inicio y finalizará con una etiqueta de finalización.

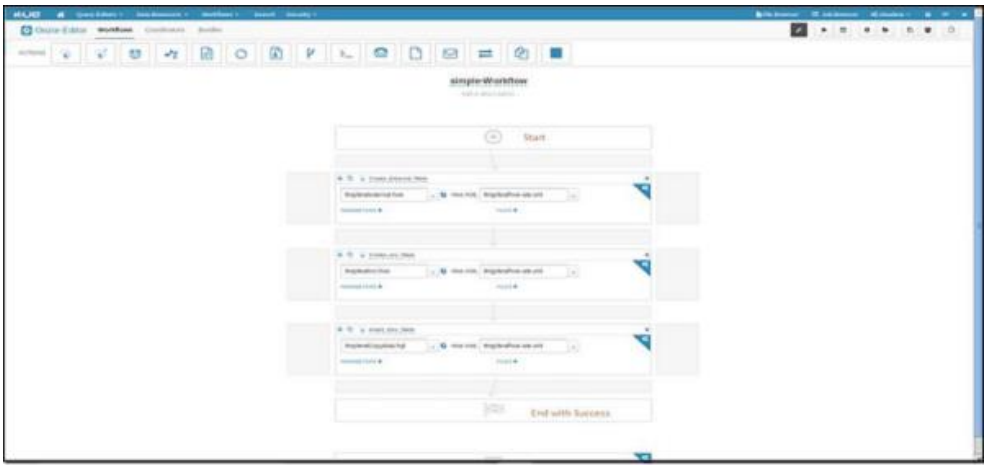
**Casos de uso de Apache Oozie**
Apache Oozie es utilizado por los administradores del sistema Hadoop para ejecutar análisis de registro complejos en HDFS. Los desarrolladores de Hadoop usan Oozie para realizar operaciones ETL en datos en un orden secuencial y guardan la salida en un formato específico (Avro, ORC, etc.) en HDFS.

En una empresa, los trabajos de Oozie están programados como coordinadores o paquetes.

**Editores de Oozie**
Antes de sumergirnos en Oozie, echemos un vistazo rápido a los editores disponibles para Oozie.
La mayoría de las veces, no necesitará un editor y escribirá los flujos de trabajo utilizando cualquier editor de texto popular (como Notepad ++, Sublime o Atom), como lo haremos en este tutorial.
Pero como principiante tiene sentido crear un flujo de trabajo mediante el método de arrastrar y soltar usando el editor y luego ver cómo se genera el flujo de trabajo. Además, para asignar GUI con el flujo de trabajo real.xml creado por el editor. Esta es la única sección donde discutiremos sobre los editores de Oozie y no la utilizaremos en nuestro tutorial.

El más popular entre los editores de Oozie es Hue.

**Hue Editor**
Este editor es muy útil de usar y está disponible con casi todas las soluciones de proveedores de Hadoop.
La siguiente captura de pantalla muestra un flujo de trabajo de ejemplo creado por este editor.



Puede arrastrar y soltar controles y acciones y agregar su trabajo dentro de estas acciones.

**Apache Oozie - Workflow**
El flujo de trabajo en Oozie es una secuencia de acciones dispuestas en un DAG (Direct Acyclic Graph) de dependencia de control. Las acciones están en dependencia controlada ya que la siguiente acción solo puede ejecutarse según el resultado de la acción actual. Las acciones posteriores dependen de su acción previa. Una acción de flujo de trabajo puede ser una acción de Hive, acción Pig, acción de Java, acción de Shell, etc. Puede haber árboles de decisión para decidir cómo y con qué condición debe ejecutarse un trabajo.
Un fork se usa para ejecutar múltiples trabajos en paralelo. Los flujos de trabajo de Oozie se pueden parametrizar (variables como $ {nameNode} se pueden pasar dentro de la definición del flujo de trabajo). Estos parámetros provienen de un archivo de configuración llamado archivo de propiedades. (.property)

**Apache Oozie - Coordinator**
Las aplicaciones coordinadoras permiten a los usuarios programar flujos de trabajo complejos, incluidos los flujos de trabajo que se programan regularmente. El coordinador de Oozie modela los desencadenantes de ejecución del flujo de trabajo en forma de predicados de tiempo, datos o eventos. El trabajo de flujo de trabajo mencionado dentro del Coordinador se inicia solo después de que se cumplan las condiciones dadas.

**Apache Oozie - Bundle**
El sistema Oozie Bundle permite al usuario definir y ejecutar un conjunto de aplicaciones coordinadoras a menudo denominadas canalizaciones de datos. No hay una dependencia explícita entre las aplicaciones del coordinador en un paquete. Sin embargo, un usuario podría usar la dependencia de datos de las aplicaciones del coordinador para crear una interconexión de aplicaciones de datos implícita.

El usuario podrá iniciar / detener / suspender / reanudar / volver a ejecutar en el nivel de paquete, lo que dará como resultado un control operativo mejor y más fácil.