**Documentación del entorno de Hadoop**

**Introducción**

La imagen del contenedor creado para el entorno de Hadoop está basada en la oficial de Centos, con una serie de herramientas instaladas y configuradas, además de modificaciones en el entorno que se listarán en este documento.

La imagen de Centos utilizada está descargada del repositorio de dockerhub, con la etiqueta **centos7**.

Los subtítulos con el prefijo **Archivo** en este documento hacen referencia a los archivos utilizados en la construcción de la imagen del laboratorio. En estas secciones se va a describir para qué es utilizado el archivo, su estructura y como modificarlo si se desea hacer un cambio.

Algo importante para mencionar es que a pesar de que los contenedores que se basan en esta imagen tienen un rol diferente **todos tienen las mismas herramientas**. El rol lo adquieren en una etapa posterior que se describirá en este mismo documento.

**Archivo Dockerfile**

Es el archivo donde se lista la serie de instrucciones que darán paso a la construcción de la imagen del entorno hadoop. Está dividido por secciones:

* Utilidades varias: Aquí se instalan todas las utilidades y dependencias que el ecosistema de hadoop necesita para funcionar, como por ejemplo Java8, SSH y python.
* Hadoop: Se crea un usuario hadoop, que será el responsable de las ejecuciones que tengan que ver con todo el ecosistema hadoop. Se define la ruta para almacenar los datos del HDFS **/data/datanode/** y se copian los archivos de hadoop para que la instalación del mismo sea en **/opt/hadoop/**. Posteriormente se reemplazan varios archivos de configuración (que se detallan más adelante en este mismo documento) y por último, se colocan las variables de entorno necesarias para que hadoop funcione correctamente.
* Hive: Se copian los archivos de hive a **/opt/hadoop/hive/,** se instala el conector de Hive-Mysql, se cambia el archivo de configuración **hive-site.xml** y se inicializan las variables de entorno.
* Spark: Se copian los archivos de spark a **/opt/hadoop/spark/** y se colocan las variables de entorno correspondientes
* Livy: Se intala en **/opt/hadoop/livy/**
* SSH: Configuraciones necesarias para que el servicio de SSH (por el cual se comunican todos los contenedores de este ecosistema) funcione correctamente.
* Arranque: Se ejecuta el script **clsuter\_start.sh** del cual se hablará posteriormente en este mismo documetno.

**Archivo capacity-scheduler.xml**

Es el archivo dónde se le indica a YARN que no usará el resource-calculator que tiene por default, sino que en cambio usará el indicado. Esto cambiando la propiedad con nombre **yarn.scheduler.capacity.resource-calculator** poniéndole el valor de **org.apache.hadoop.yarn.util.resource.DominantResourceCalculator**

Básicamente, con esta propiedad se le dice a YARN que utilice la configuración que nosotros asignemos en el archivo **yarn-site.xml** del cual hablaremos posteriormente.

**Archivo yarn-site.xml**

Está dividido en 3 secciones de acuerdo a las configuraciones, las propiedades que no se mencionen es simplemente porque son dependencias y no es objetivo de este documento entrar en detalle:

* Configuraciones generales:
  + yarn.resourcemanager.hostname: Especifica el nombre del host donde correrá YARN, en este caso en el namenode.
  + yarn.resourcemanager.webapp.address: La dirección IP y el puerto donde correrá la interfaz web de YARN. (0.0.0.0 hace referencia al localhost, es decir al namenode)
* Configuraciones de CPU:
  + yarn.nodemanager.resource.cpu-vcores: La cantidad de CPUs que tiene el host.
  + yarn.scheduler.minimum-allocation-vcores: La cantidad mínima de CPUs en total por contenedor.
  + yarn.scheduler.maximum-allocation-vcores: La cantidad máxima de CPUs en total por contenedor.
* Configuraciones de memoria:
  + yarn.nodemanager.resource.memory-mb: Memoria destinada a utilizar YARN (memoria del host, en este caso 100GB)
  + yarn.scheduler.minimum-allocation-mb: Cantidad mínima de memoria por contenedor creado
  + yarn.scheduler.maximum-allocation-mb: Cantidad máxima de memoria por contenedor creado

Para conocer más de las configuraciones de YARN, por favor dirígete a:

* <https://docs.cloudera.com/HDPDocuments/HDP2/HDP-2.0.6.0/bk_installing_manually_book/content/rpm-chap1-11.html>
* <https://docs.cloudera.com/documentation/enterprise/6/6.3/topics/cm_mc_yarn_service.html>

**Archivo mapred-site.xml**

Es el archivo de configuraciones para Map Reduce. Se divide en 3 secciones de acuerdo a las configuraciones, las propiedades que no se mencionen es simplemente porque son dependencias y no es objetivo de este documento entrar en detalle:

* Configuraciones generales:
  + mapreduce.framework.name: Aquí se especifica el servicio que se usará para los jobs de MapReduce, en este caso YARN.
* Variables de entorno: Se especificarán las variables de entorno que se necesitan para que el entorno hadoop funcione correctamente.
* Configuraciones de memoria:
  + mapreduce.{map/reduce}.memory.mb: Se especifica el tamaño que tendrá cada uno de los mappers/reducers
  + mapreduce.{map/reduce}.java.opts: Tamaño de la memoria heap desitnada a los mappers/reducers. Lo recomendable es que sea aproximadamente el 80% de lo asignado en la propiedad anteriormente descrita.

**Archivo hdfs-site.xml**

Archivo de configuración del HDFS de hadoop. Las configuraciones más importantes aquí son:

* dfs.replication: La cantidad de veces que estarán replicados los archivos en el hdfs (aquí solo es 1 porque no queremos duplicar los valores en el nodo)
* dfs.namenode.name.dir: Donde se almacenarán los metadatos del namenode
* dfs.datanode.data.dir: Donde se almacenarán los archivos del hdfs.

**Archivo hive-site.xml**

Archivo de configuración de hive. Las configuraciones más importantes son:

* javax.jdo.option.ConnectionURL: La cadena de conexión para la base de datos de hive. La metastore. En este caso el hostname del contenedor que tiene esta base de datos es metastore y el nombre de la base de igual forma es metastore.
* javax.jdo.option.ConnectionUserName: Nombre de usuario de la base de datos (tiene que tener los permisos de creación y modificación de la metastore)
* javax.jdo.option.ConnectionPassword: Contraseña del usuario de la metastore.
* mapreduce.framework.name: Le indica a hive que motor de framework utilizará. En este caso YARN.
* hive.exec.mode.local.auto: Si está seteado en false (como es nuestro caso), hive no iniciará en modo local, sino que lo hará en modo cluster.

**Archivo core-site.xml**

Archivo de configuración del namenode. En la propiedad df.defaultFS se indica la ruta HDFS donde estará corriendo el servicio de hdfs. En este caso en namenode:9000

**Archivo workers**

Aquí se lista el hostname de los contenedores que servirán como workers en hadoop, separados por un salto de línea entra cada uno de ellos. En este caso datanode, datanode2 y datanode3

**Archivo hue.ini**

Aquí se describen las configuraciones para hue.

* [[database]]: Base de datos para almacenar metadatos de hue. Se especifica el motor (engine), host, puerto, usuario y password. (para nuestro caso el contenedor que tiene esta base de datos es mysql\_general)
* fs\_defaultfs: El mismo que se especificó en el archivo **core-site.xml**
* webhdfs\_url: La dirección URL del servicio web del hdfs
* resourcemanager\_api\_url: URL y purto donde corre el resource manager.
* [beeswax]: Aquí se especifica el hostname del servidor de hive (en este caso hiveServer) así como el puerto y el puerto para el servicio http. Por último el hostname de la metastore y su puerto
* livy\_server\_url: URL del servidor livy. Para poder ejecutar queries de Spark desde hue.

**Archivo cluster\_start.sh**

Este archivo corre en cada uno de los contenedores, es el que dicta qué rol tendrán cada uno de ellos en base al nombre del host.

Genera e intercambia las llaves SSH para los contenedores y por último inicia todos los servicios necesarios del ecosistema hadoop.

Para agregar un contenedor adicional a este ecosistema, se debería especificar aquí su intercambio de llaves y asignar el rol que tendrá.