Guía para Instalar Hadoop en Ubuntu

Esta guía te proporcionará instrucciones paso a paso para instalar y configurar Hadoop en un sistema Ubuntu.

Paso 0: Actualizar los paquetes a la última versión y crear usuario

```
sudo apt-get update & apt-get upgrade -y
```

Es recomendable crear un usuario dedicado para ejecutar Hadoop, lo que ayuda a mantener el sistema organizado y seguro.

```
sudo adduser Hadoop
usermod -aG sudo hadoop
su - hadoop
```

- crea un nuevo usuario llamado "hadoop".
- añade el usuario hadoop al grupo sudo.
- cambia al nuevo usuario hadoop.

Paso 1: Instalar Java JDK 8

Hadoop requiere Java JDK 8 para funcionar correctamente. Instalaremos la versión de OpenJDK 8. Openjdk-8-jdk-headless es la versión sin interfaz gráfica, adecuada para servidores y entornos de desarrollo.

```
sudo apt install openjdk-8-jdk-headless
java -version
```

- Deberías ver una salida que indica que Java está en la versión 1.8.

Paso 2: Instalar SSH y PDSH

SSH es necesario para la comunicación entre nodos en Hadoop. PDSH es una herramienta para ejecutar comandos en múltiples hosts en paralelo.

```
sudo apt-get install ssh
sudo apt-get install pdsh
```

- ssh permite conexiones seguras entre máquinas.
- pdsh facilita la administración de clústeres al ejecutar comandos en varios nodos simultáneamente.

Paso 3: Añadir IPs de nodo master y workers

Consultar la IP asignada y cambiar en el fichero hosts si es necesario.

```
ip a
vi /etc/hosts
        192.168.0.45 nodomaster
        192.168.0.45 nodoworker1
```

Paso 4: Descargar e Instalar Hadoop

Descarga la versión más reciente de Hadoop desde el sitio oficial de Apache Hadoop y extrae el archivo.

```
cd
wget http://dlcdn.apache.org/hadoop/common/stable/hadoop-3.4.1.tar.gz
tar -xzf hadoop-3.4.1.tar.gz
mv hadoop-3.4.1 /home/hadoop
```

- obtiene la última versión estable de Hadoop
- descomprime el archivo tar.gz.
- mueve el directorio extraído al directorio /usr/local/hadoop

Paso 5: Configurar Variables de Entorno

Es necesario configurar las variables de entorno para que el sistema reconozca las ubicaciones de Java y Hadoop.

Editar /etc/environment

```
sudo vi /etc/environment
```

Agregar la siguiente línea:

```
JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64"
```

- JAVA_HOME apunta al directorio donde se instaló Java. Esto es necesario para que Hadoop pueda encontrar Java.

Crear un directorio para almacenar el registro:

```
mkdir /logs
```

Editar ~/.bashrc

```
vi ~/.bashrc
```

Agregar las siguientes líneas al final del archivo:

```
export HADOOP_HOME=/home/hadoop/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME

export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop

export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

Explicación de las Variables:

- HADOOP_HOME es el directorio donde instalaste Hadoop.
- Las otras variables establecen rutas necesarias para los componentes de Hadoop.
- PATH se actualiza para incluir los binarios de Hadoop, facilitando su ejecución desde cualquier lugar.
- HADOOP_OPTS define opciones adicionales para Hadoop, incluyendo la ruta a las bibliotecas nativas.

Aplicar los cambios:

```
source ~/.bashrc
```

- Esto recarga el archivo .bashrc y aplica las nuevas variables de entorno.

Paso 6: Configurar HDFS

Ahora configuraremos el sistema de archivos distribuido de Hadoop (HDFS) para especificar dónde almacenará los datos y otras configuraciones importantes.

Crear Directorios de Datos

```
mkdir -p ~/data/hdfs
```

- mkdir -p crea el directorio y todos sus directorios padre si no existen.
- ~/data/hdfs es el directorio donde HDFS almacenará los datos del NameNode y DataNode.

Crear Directorios para NameNode y DataNode:

```
mkdir -p ~/data/hdfs/namenode
mkdir -p ~/data/hdfs/datanode
```

Editar hdfs-site.xml

```
vi $HADOOP_HOME/etc/hadoop/hdfs-site.xml
```

Agregar o modificar el siguiente contenido:

```
<configuration>
  <property>
     <name>dfs.namenode.rpc-bind-host</name>
    <value>nodomaster</value>
  </property>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:///home/hadoop/data/hdfs/namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:///home/hadoop/data/hdfs/datanode</value>
  </property>
</configuration>
```

- <name>dfs.replication</name> establece el factor de replicación de HDFS. En un entorno de nodo único, se establece en 1 para evitar errores.
- <name>dfs.namenode.name.dir</name> especifica el directorio local donde el NameNode almacenará su información de metadatos.
- <name>dfs.datanode.data.dir</name> especifica el directorio local donde el DataNode almacenará los bloques de datos.

Configurar Workers

El fichero workers es usado por el script de arranque para levantar los demonios en todos los datanodes.

```
vi ~/hadoop/etc/hadoop/workers
        nodoworker1
        nodoworker2
        …
```

Paso 7: Configurar Core Site

Este archivo define las propiedades básicas del sistema Hadoop, incluyendo la URI del sistema de archivos.

Editar core-site.xml

```
vi $HADOOP_HOME/etc/hadoop/core-site.xml
```

Agregar o modificar el siguiente contenido:

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://nodomaster:9000</value>
  </property>
  <property>
      <name>hadoop.proxyuser.hadoop.hosts</name>
      <value>*</value>
      <description>Permitir al usuario 'hadoop' realizar impersonación
desde cualquier host</description>
   </property>
   <property>
        <name>hadoop.proxyuser.hadoop.groups</name>
        <value>*</value>
        <description>Permitir al usuario 'hadoop' realizar impersonación
para cualquier grupo</description>
    </property>
</configuration>
```

- <name>fs.defaultFS</name> define la URI del sistema de archivos por defecto. hdfs://localhost:9000 indica que HDFS está corriendo en localhost en el puerto 9000.

Paso 8: Configurar Acceso SSH sin Contraseña

Hadoop utiliza SSH para gestionar sus demonios, incluso en un clúster de un solo nodo. Configuraremos SSH para permitir el acceso sin contraseña.

```
ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 0600 ~/.ssh/authorized_keys
```

- ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa genera un par de claves RSA sin contraseña.

- cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys agrega la clave pública al archivo de claves autorizadas.
- chmod 0600 ~/.ssh/authorized_keys establece los permisos adecuados para el archivo de claves autorizadas.

Verificación:

Prueba la conexión SSH a localhost, no debería pedir contraseña.

```
ssh localhost
```

Paso 9: Formatear el NameNode

Antes de iniciar Hadoop por primera vez, es necesario formatear el NameNode para inicializar el sistema de archivos HDFS.

```
hdfs namenode -format
```

- Este comando prepara el NameNode para su uso, creando los directorios necesarios y configurando el sistema de archivos.
- Solo debes formatear el NameNode una vez. Si lo formateas nuevamente, se perderán todos los datos almacenados en HDFS.

Paso 10: Iniciar los Servicios de Hadoop

Ahora podemos iniciar los servicios de Hadoop para poner en marcha el clúster.

```
start-all.sh
```

- start-all.sh es un script que inicia todos los demonios de Hadoop (HDFS y YARN).

Paso 11: Verificar la Instalación

Para asegurarte de que Hadoop está funcionando correctamente, puedes acceder a las interfaces web que proporciona.

Interfaz Web de HDFS NameNode

http://nodomaster:9870/

- Esta es la interfaz de usuario del NameNode, donde puedes ver el estado del sistema de archivos, el espacio utilizado, los bloques y otros detalles.

Interfaz Web de YARN ResourceManager

http://localhost:8088/

- Aquí puedes monitorizar las aplicaciones en ejecución, el uso de recursos y el estado de los nodos en el clúster.
- Si estás accediendo desde otra máquina, reemplaza localhost con la dirección IP de tu servidor Hadoop.

Paso 12: Ejecutar una Prueba de MapReduce

Para confirmar que todo funciona correctamente, puedes ejecutar un trabajo de ejemplo de MapReduce.

Crear Directorio de Entrada en HDFS:

```
hdfs dfs -mkdir -p /user/hadoop/input
```

Copiar Archivos de Prueba al Directorio de Entrada:

```
hdfs dfs -put $HADOOP_HOME/etc/hadoop/*.xml /user/hadoop/input
```

Ejecutar el Trabajo de Ejemplo:

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar wordcount /user/hadoop/input /user/hadoop/output
```

- Este comando ejecuta el ejemplo wordcount, que cuenta las palabras en los archivos de entrada y almacena el resultado en el directorio de salida.

Verificar los Resultados:

```
hdfs dfs -cat /user/hadoop/output/part-r-00000
```

- Este comando muestra el contenido del archivo de salida, donde puedes ver las palabras y su número de apariciones.

Conclusión

¡Felicidades! Has instalado y configurado exitosamente Hadoop en tu sistema Ubuntu. Ahora estás listo para explorar las capacidades de Hadoop para el procesamiento de grandes volúmenes de datos.

Pasos Adicionales:

- **Configuración Avanzada:** Puedes ajustar más configuraciones en los archivos mapred-site.xml y yarn-site.xml según tus necesidades.
- **Seguridad:** Considera configurar Kerberos y otros mecanismos de seguridad para entornos de producción.
- **Monitorización:** Utiliza herramientas como Apache Ambari o Cloudera Manager para monitorear y administrar tu clúster.

Recursos Adicionales

- Documentación Oficial de Hadoop
- Tutoriales de Hadoop

Programa ejemplo Wordcount

```
public class WordCount {
    public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
            String line = value.toString();
            StringTokenizer tokenizer = new StringTokenizer(line);
            while (tokenizer.hasMoreTokens()) {
                word.set(tokenizer.nextToken());
                context.write(word, one);
            }
        }
    }

    public static class Reduce extends Reducer<Text, IntWritable, Text,
IntWritable> {
        public void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            context.write(key, new IntWritable(sum));
        }
    }
}
```