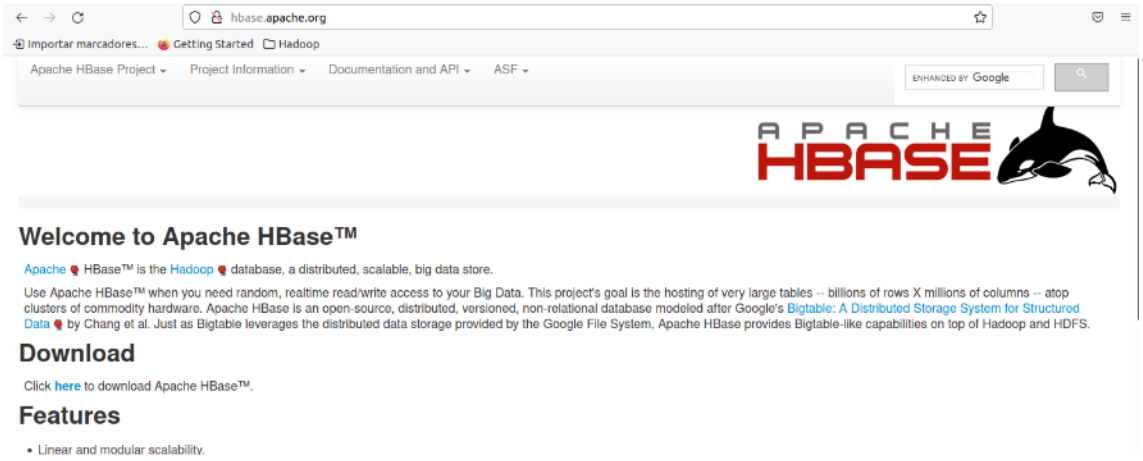
7. HBASE

Instalar y probar le sistema de base de datos HBASE

7.1. Instalar HBASE

7.1.1. Descargar HBASE

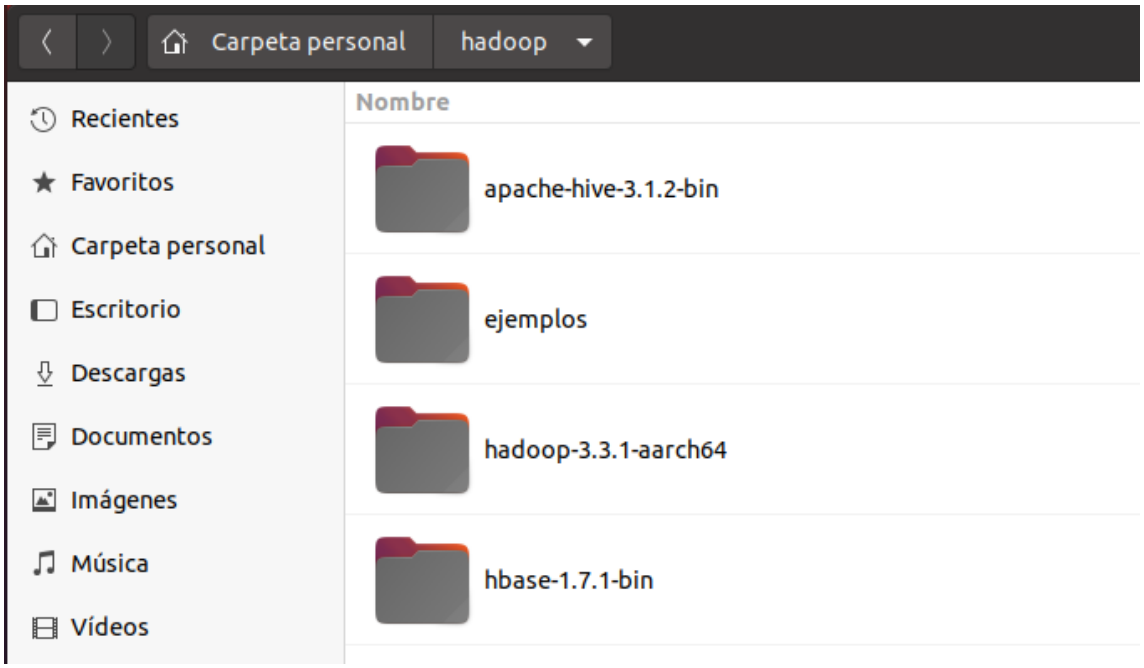
Descargar la versión 1.7.1 HBASE desde https://hbase.apache.org/



Descargar hbase-1.7.1-bin.tar.gz compatible con la versión de java instalada

7.1.2. Descomprimir y configurar variables de entorno

Una vez descargado, descomprimir los archivos en la carpeta correspondiente. Es recomendable agregar la ruta a $PATH para facilitar la usabilidad. En el ejemplo descomprimo el archivo en la carpeta “hadoop” junto con el resto de herramientas



Agrego la carpeta bin a $PATH y creo la variable HBASE_HOME. Para abrir el archivo de configuración:

```
$ sudo gedit ~/.profile
```

Agregar las rutas en función de los directorios donde se hayan descomprimido los archivos

```
PATH="/home/hadoop/hadoop/hbase-1.7.1-bin/hbase-1.7.1/bin:$PATH"

export HBASE_HOME="/home/hadoop/hadoop/hbase-1.7.1-bin/hbase-1.7.1"
```

Agregar la variable JAVA_HOME en el caso de que no esté previamente. Con la instalación básica está en la ruta “usr/lib/jvm/java-8-openjdk-amd64”

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

Resultado de las modificaciones:

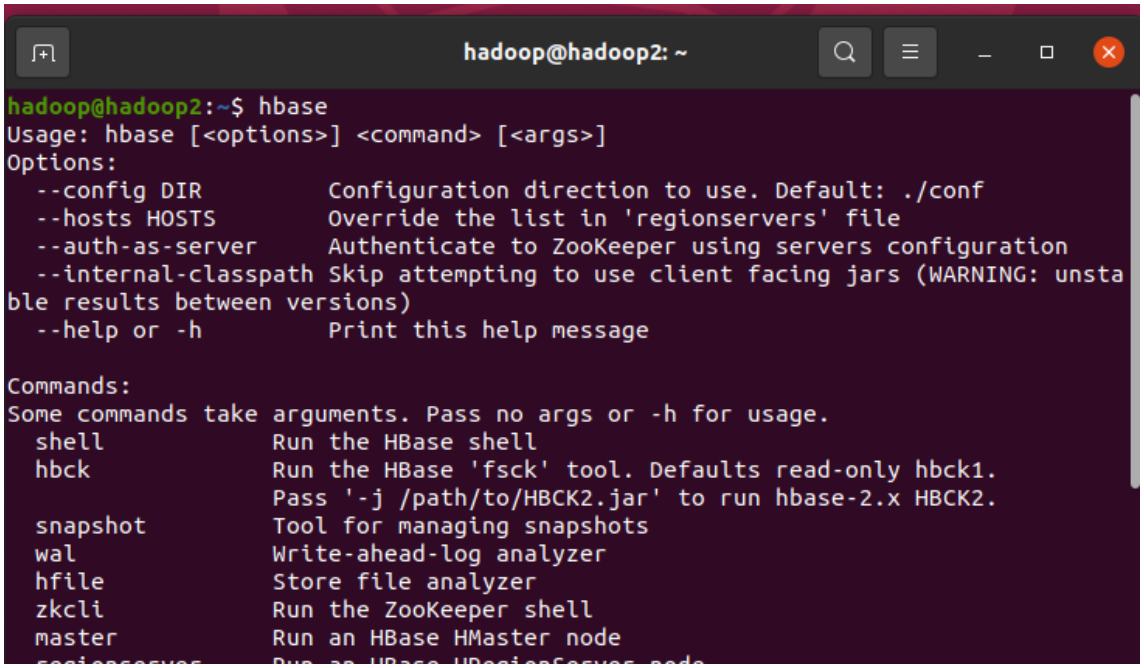
```
28
29 #agregro las variables para hadoop y modifico el path
30 PATH="/home/hadoop/hadoop/hadoop-3.3.1-aarch64/hadoop-3.3.1/bin:$PATH"
31 PATH="/home/hadoop/hadoop/hadoop-3.3.1-aarch64/hadoop-3.3.1/sbin:$PATH"
32 PATH="/home/hadoop/hadoop/apache-hive-3.1.2-bin/bin:$PATH"
33 PATH="/home/hadoop/hadoop/hbase-1.7.1-bin/hbase-1.7.1/bin:$PATH"
34
35
36 export HADOOP_HOME="/home/hadoop/hadoop/hadoop-3.3.1-aarch64/hadoop-3.3.1"
37 export HADOOP_MAPRED_HOME=$HADOOP_HOME
38 export HADOOP_COMMON_HOME=$HADOOP_HOME
39 export HADOOP_HDFS_HOME=$HADOOP_HOME
40 export YARN_HOME=$HADOOP_HOME
41 export HIVE_HOME="/home/hadoop/hadoop/apache-hive-3.1.2-bin"
42 export HBASE_HOME="/home/hadoop/hadoop/hbase-1.7.1-bin/hbase-1.7.1"
43 export JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64"
44
```

Reiniciar la sesión para asegurar que carga correctamente las variables

Verificar que HDFS está operativo y que tanto el namenode como el datanode están activos

Para verificar que la instalación es correcta llamar a hbase y comprobar que muestra la lista de comandos

```
$ hbase
```

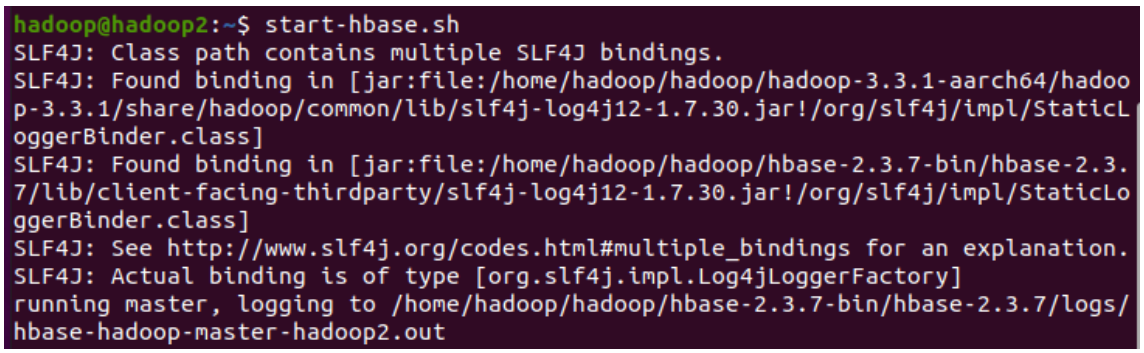


7.2. Funcionamiento básico

7.2.1. Iniciar HBASE

La configuración básica es en modo monopuesto. Utiliza un directorio temporal en el sistema de ficheros local. Tanto Hbase master, el regionserver y la instancia de ZooKeeper se ejecutan en la misma JVM.

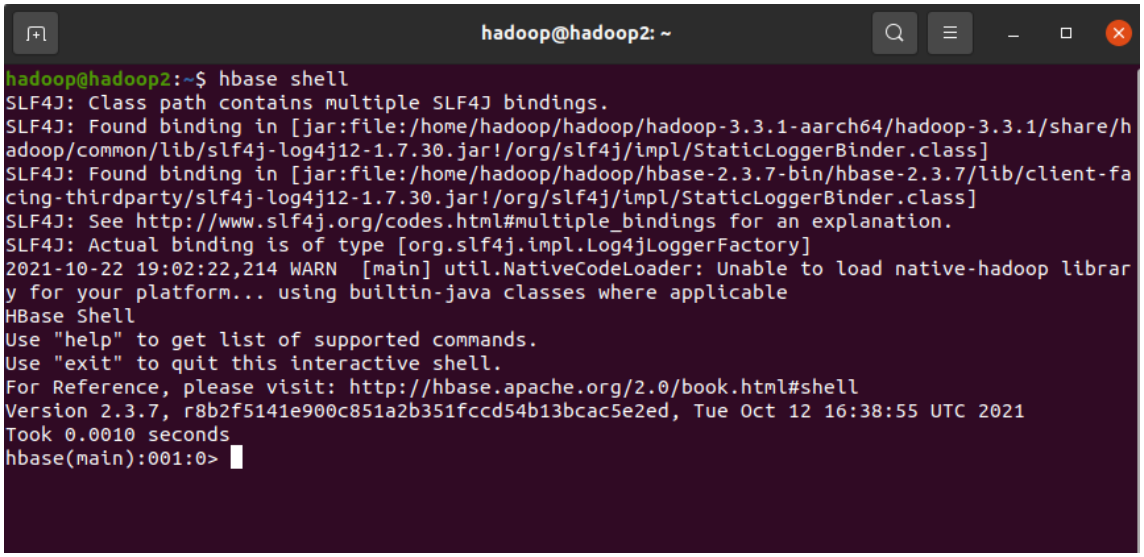
```
$ start-hbase.sh
```



7.2.2. Iniciar Hbase Shell

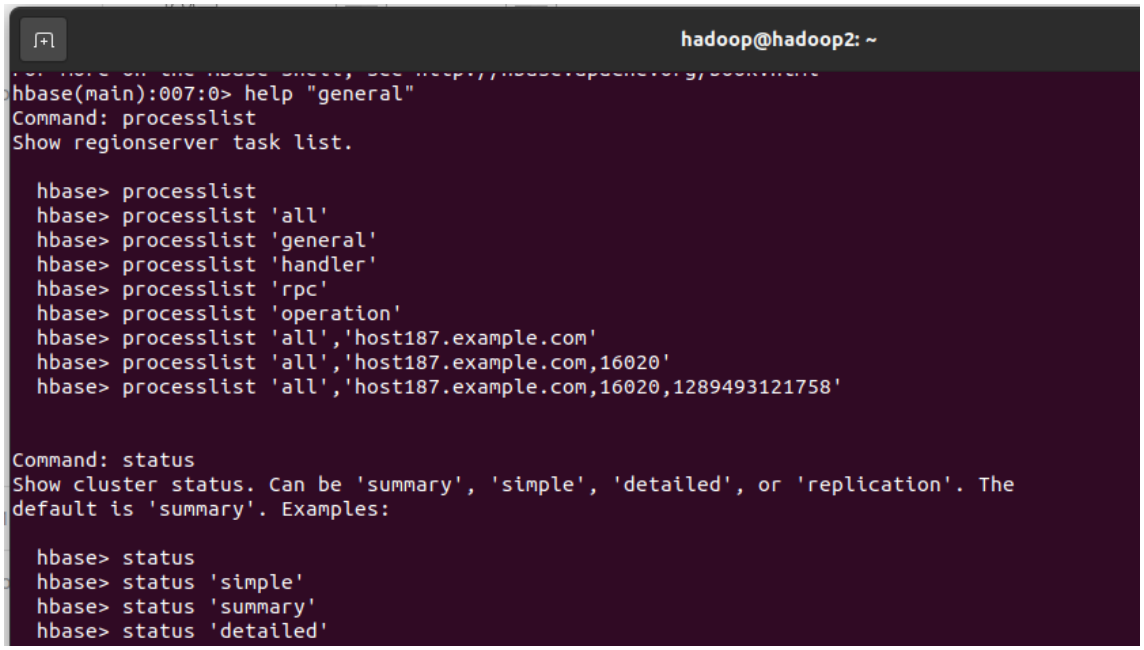
Iniciar un interprete de JRuby JRB con comandos específicos de HBASE

```
$ hbase shell
```



Para obtener ayuda sobre un grupo de comandos utilizar help “Nombre del grupo”

```
> help “general”
```

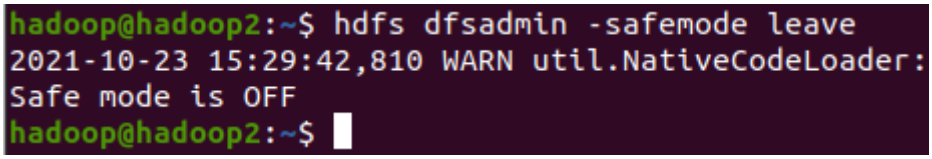


Para comprobar los servicios ejecutar $ sudo jps debería mostar HMaster

7.3. Primeras instrucciones Hbase

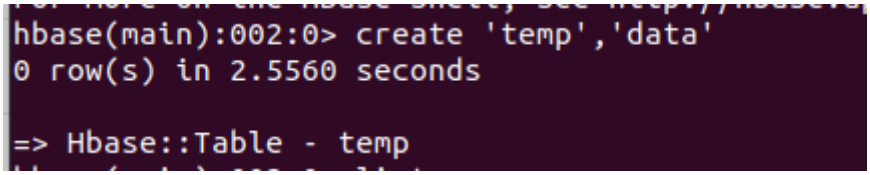
Previo a trabaja con Hbase tenemos que modificar el nivel de seguridad de Hdfs. HBASE en local no puede trabajar si tiene hdfs la seguridad activada, por lo que para hacer pruebas tenemos que desactivarla

```
$ hdfs dfsadmin -safemode leave
```



7.3.1. Creación de una tabla con la estructura por defecto.

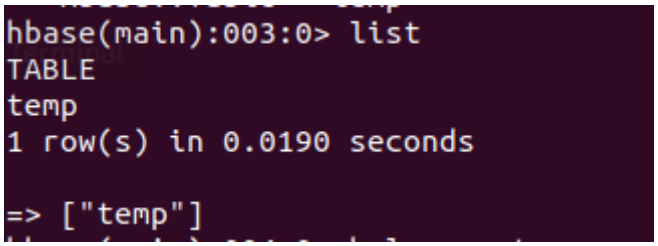
```
hbase(main):000:0>  create 'temp','data'
```



Nota: con el comando help “create” muestra diferentes opciones para crear y rellenar tablas

Verificar la creación de la tabla con list. Muestra todas la tablas en el espacio de nombres

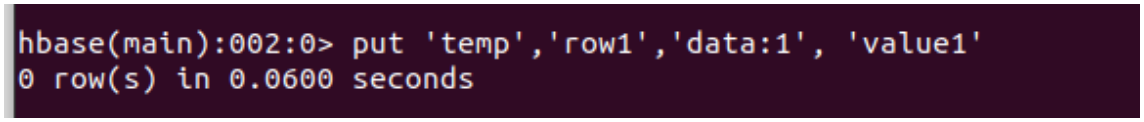
```
hbase(main):000:0> list
```



7.3.2. Cargar datos manualmente

En este ejemplo, indicamos la tabla, el identificador, la fila, la familia y el valor.

```
hbase(main):000:0> put 'temp','row1','data:1', 'value1'
```



Repetimos el proceso con la fila 2 y 3

```
hbase(main):000:0> put 'temp','row2','data:2', 'value2'

hbase(main):000:0> put 'temp','row3','data:3', 'value3'
```

7.3.3. Leer datos

Para obtener los datos de una fila utilizamos get

```
hbase(main):000:0> get 'temp','row1'
```

```
hbase(main):004:0> get 'temp','row1'
COLUMN                                CELL
 data:1                               timestamp=1634996653631, value=value1
1 row(s) in 0.0340 seconds
```

Para ver el contenido de una tabla utilizamos scan

```
hbase(main):003:0> scan 'temp'
```

```
hbase(main):003:0> scan 'temp'
ROW                                   COLUMN+CELL
 row1                                 column=data:1, timestamp=1634996653631, value=value1
 row2                                 column=data:2, timestamp=1634996762654, value=value2
 row3                                 column=data:3, timestamp=1634997141386, value=value3
3 row(s) in 0.0120 seconds
```

7.3.4. Eliminar tablas

Primero tenemos que deshabilitar la tabla (disable) y posteriormente borrarla (drop)

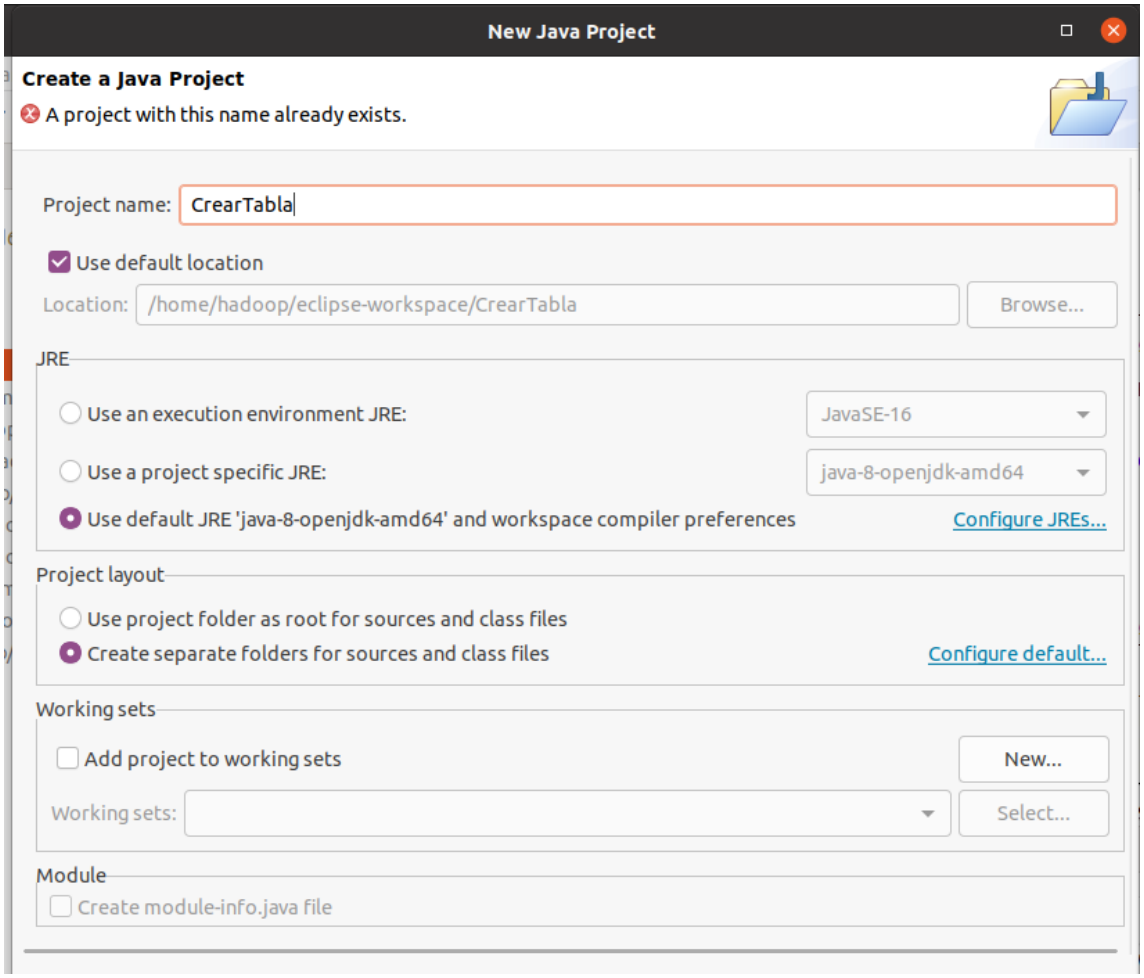
```
hbase(main):004:0> disable 'temp'

hbase(main):005:0> drop 'temp'
```

```
hbase(main):004:0> disable 'temp'
0 row(s) in 2.3410 seconds

hbase(main):005:0> drop 'temp'
0 row(s) in 1.2620 seconds

hbase(main):006:0> list
TABLE
0 row(s) in 0.0170 seconds

=> []
hbase(main):007:0>
```

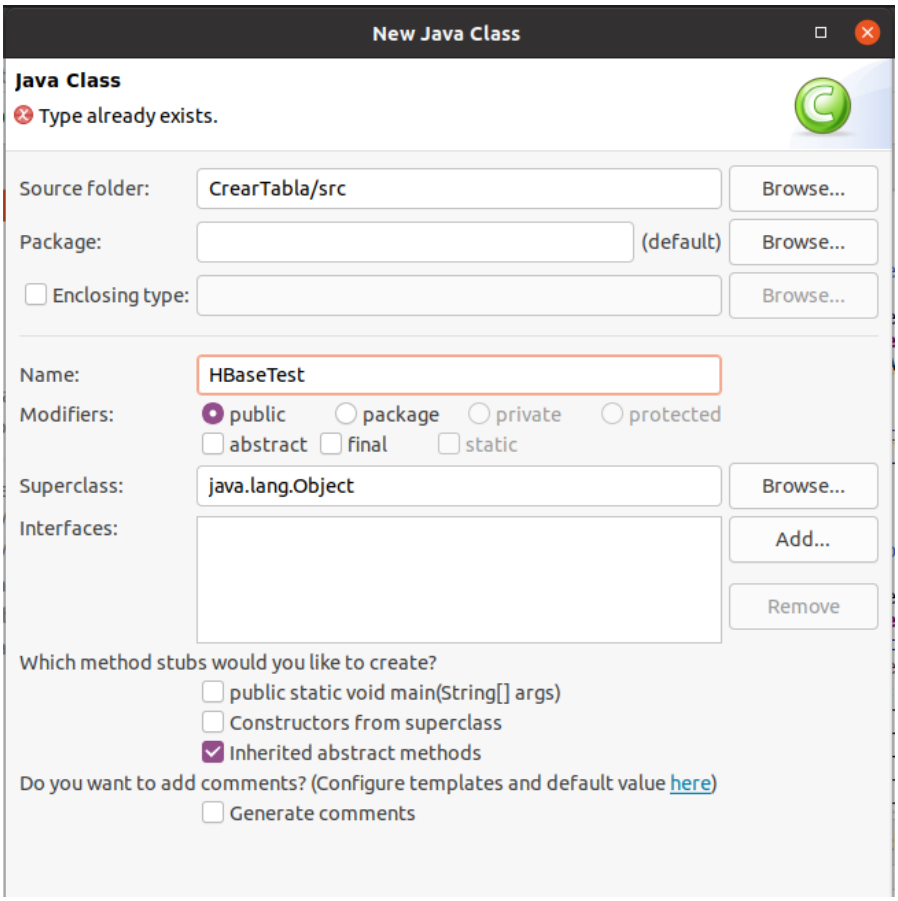
7.4. Creación de aplicaciones con JAVA

Crearemos un proyecto de java con las referencias de HBASE

7.4.1. Crear nuevo proyecto java en eclipse

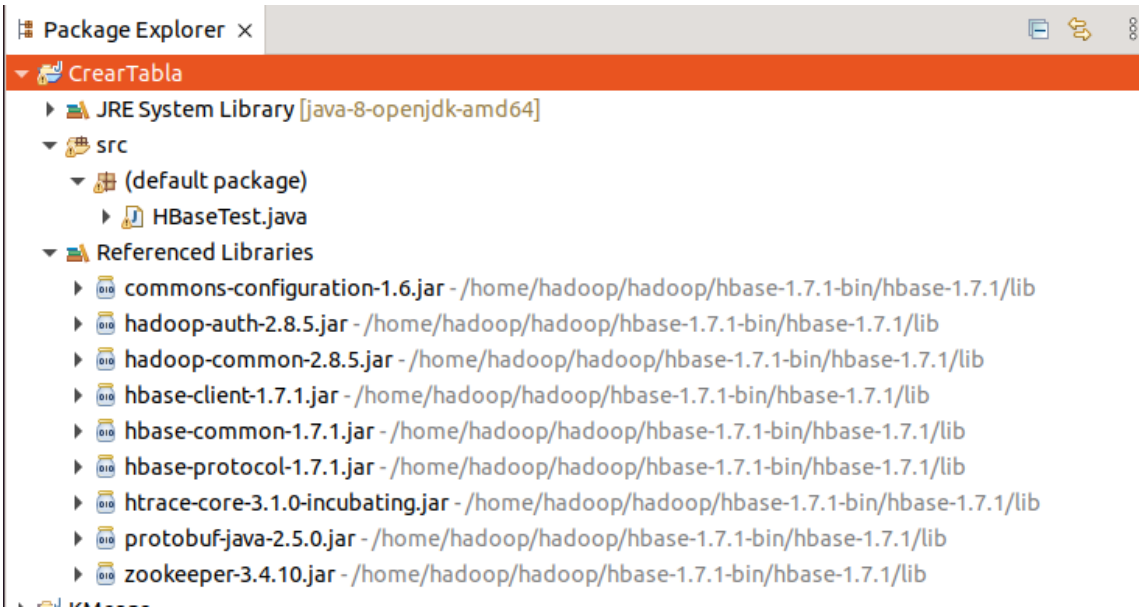


7.4.2. Agregar la clase HBaseTest



7.4.3. Configurar Build Path

Agregar las referencias para generar el programa. Desde el menú contextual del proyecto > Build Path > Configure Bulid Path. Agregar los siguientes archivos que se encuentran $HBASE_HOME/lib.



7.4.4. Copiar código a la clase HBaseTest

Este ejemplo contiene los principales procesos de manipulación de la información de Hbase

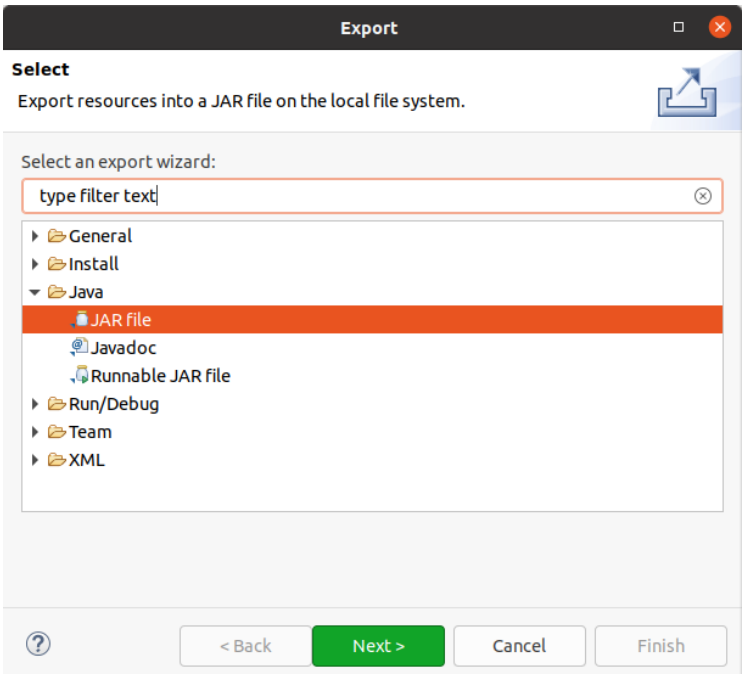
```
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.HColumnDescriptor;
import org.apache.hadoop.hbase.HTableDescriptor;
import org.apache.hadoop.hbase.KeyValue;
import org.apache.hadoop.hbase.MasterNotRunningException;
import org.apache.hadoop.hbase.ZooKeeperConnectionException;
import org.apache.hadoop.hbase.client.Delete;
import org.apache.hadoop.hbase.client.Get;
import org.apache.hadoop.hbase.client.HBaseAdmin;
import org.apache.hadoop.hbase.client.HTable;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.ResultScanner;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.util.Bytes;

public class HBaseTest {

    private static Configuration conf =null;
     /**
      * Initial configuration
     */
     static {
         conf = HBaseConfiguration.create();
     }

    /**
     * Create a table
     */
    public static void creatTable(String tableName, String[] familys) throws
Exception {
        HBaseAdmin admin = new HBaseAdmin(conf);
        if (admin.tableExists(tableName)) {
            System.out.println("table already exists!");
        } else {
            HTableDescriptor tableDesc = new HTableDescriptor(tableName);
            for(int i=0; i<familys.length; i++){
                tableDesc.addFamily(new HColumnDescriptor(familys[i]));
            }
            admin.createTable(tableDesc);
            System.out.println("create table " + tableName + " ok.");
        }
    }

    /**
     * Delete table
     */
    public static void deleteTable(String tableName) throws Exception {
       try {
           HBaseAdmin admin = new HBaseAdmin(conf);
           admin.disableTable(tableName);
           admin.deleteTable(tableName);
           System.out.println("delete table " + tableName + " ok.");
       } catch (MasterNotRunningException e) {
           e.printStackTrace();
       } catch (ZooKeeperConnectionException e) {
           e.printStackTrace();
       }
    }

    /**
```

```
     * Insert a row of records
     */
    public static void addRecord (String tableName, String rowKey, String
family, String qualifier, String value)
            throws Exception{
        try {
            HTable table = new HTable(conf, tableName);
            Put put = new Put(Bytes.toBytes(rowKey));

put.add(Bytes.toBytes(family),Bytes.toBytes(qualifier),Bytes.toBytes(value))
;
            table.put(put);
            System.out.println("insert recored " + rowKey + " to table " +
tableName +" ok.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    /**
     * Delete a row of records
     */
    public static void delRecord (String tableName, String rowKey) throws
IOException{
        HTable table = new HTable(conf, tableName);
        List list = new ArrayList();
        Delete del = new Delete(rowKey.getBytes());
        list.add(del);
        table.delete(list);
        System.out.println("del recored " + rowKey + " ok.");
    }

    /**
     * Find a row of records
     */
    public static void getOneRecord (String tableName, String rowKey) throws
IOException{
        HTable table = new HTable(conf, tableName);
        Get get = new Get(rowKey.getBytes());
        Result rs = table.get(get);
        for(KeyValue kv : rs.raw()){
            System.out.print(new String(kv.getRow()) + " " );
            System.out.print(new String(kv.getFamily()) + ":" );
            System.out.print(new String(kv.getQualifier()) + " " );
            System.out.print(kv.getTimestamp() + " " );
            System.out.println(new String(kv.getValue()));
        }
    }

    /**
     * Show all data
     */
    public static void getAllRecord (String tableName) {
        try{
             HTable table = new HTable(conf, tableName);
             Scan s = new Scan();
             ResultScanner ss = table.getScanner(s);
             for(Result r:ss){
                 for(KeyValue kv : r.raw()){
                    System.out.print(new String(kv.getRow()) + " ");
                    System.out.print(new String(kv.getFamily()) + ":");
                    System.out.print(new String(kv.getQualifier()) + " ");
                    System.out.print(kv.getTimestamp() + " ");
                    System.out.println(new String(kv.getValue()));
                 }
             }
        } catch (IOException e){
```

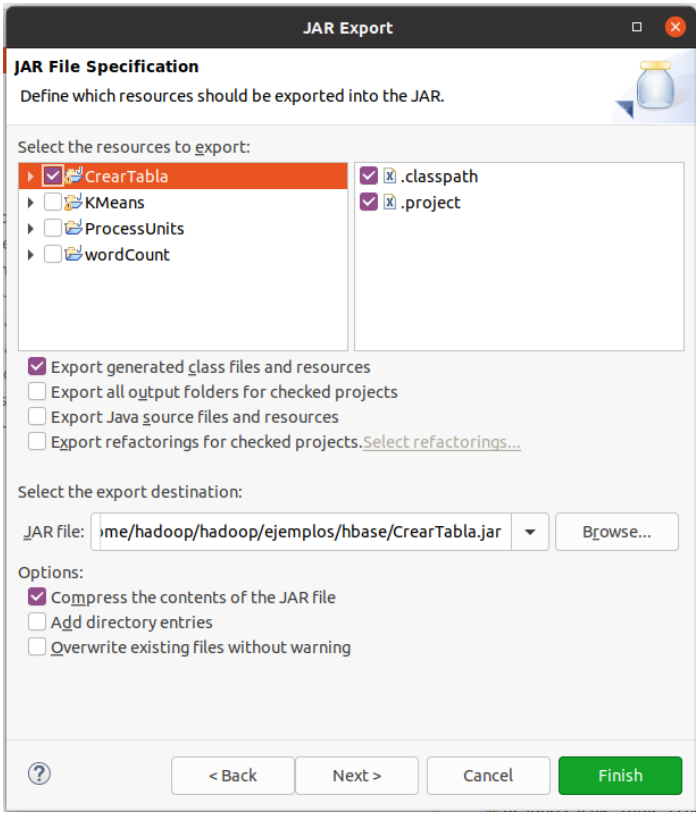
```
            e.printStackTrace();
        }
    }

    public static void  main (String [] agrs) {
        try {
            String tablename = "scores";
            String[] familys = {"grade", "course"};
            HBaseTest.creatTable(tablename, familys);

            //add record zkb
            HBaseTest.addRecord(tablename,"zkb","grade","","5");
            HBaseTest.addRecord(tablename,"zkb","course","","90");
            HBaseTest.addRecord(tablename,"zkb","course","math","97");
            HBaseTest.addRecord(tablename,"zkb","course","art","87");
            //add record  baoniu
            HBaseTest.addRecord(tablename,"baoniu","grade","","4");
            HBaseTest.addRecord(tablename,"baoniu","course","math","89");

            System.out.println("===========get one record========");
            HBaseTest.getOneRecord(tablename, "zkb");

            System.out.println("===========show all record========");
            HBaseTest.getAllRecord(tablename);

            System.out.println("===========del one record========");
            HBaseTest.delRecord(tablename, "baoniu");
            HBaseTest.getAllRecord(tablename);

            System.out.println("===========show all record========");
            HBaseTest.getAllRecord(tablename);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

7.4.5. Generar archivo JAR

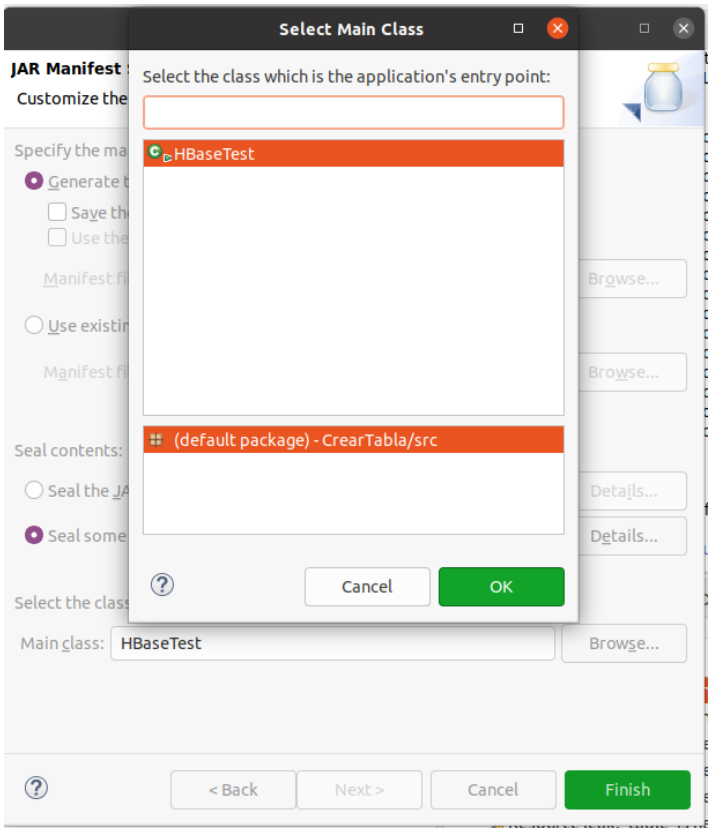
Desde el menú contextual del proyecto seleccionar Export. Seleccionar tipo de archivo JAR



Configurar el archivo JAR, indicando el nombre del archivo. En este caso es CrearTabla.jar dentro de la carpeta ejemplos



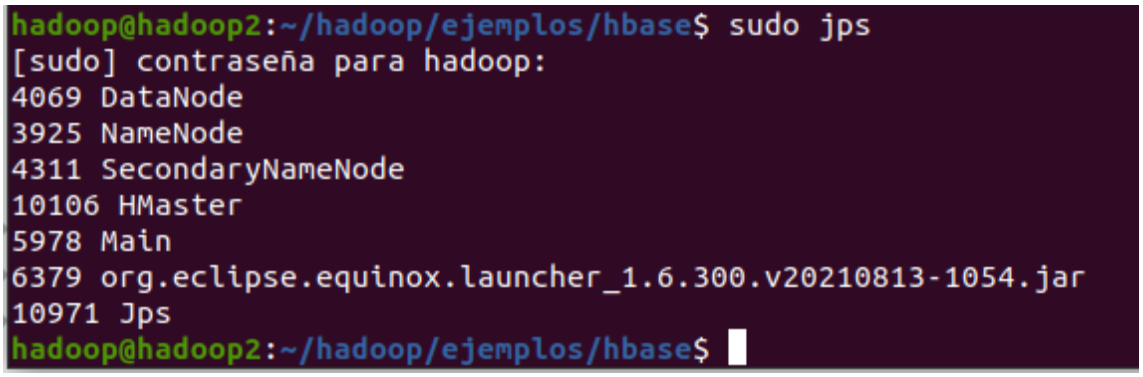
Continuar con la configuración del archivo indicando en el paso JAR Manifest file for the JAR file, la clase HBaseTest como principal



7.4.6. Ejecutar el archivo JAR

Verificar que el servicio Hbase está iniciado

```
$ sudo jps
```



Indicar el archivo JAR como ruta para ejecutar el programa

```
export HBASE_CLASSPATH=CrearTabla.jar
```

Lanzar la aplicación llamando a la clase HbaseTest

```
$ hbase HBaseTest
```

```
hadoop@hadoop2:~/hadoop/ejemplos/hbase$ hbase HBaseTest
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/hbase-1.7.1-bin/hbas
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/hadoop-3.3.1-aarch64
```

.....

```
insert recored zkb to table scores ok.
insert recored zkb to table scores ok.
insert recored zkb to table scores ok.
insert recored zkb to table scores ok.
insert recored baoniu to table scores ok.
insert recored baoniu to table scores ok.
==========get one record========
zkb course: 1635066093005 90
zkb course:art 1635066093023 87
zkb course:math 1635066093018 97
zkb grade: 1635066092993 5
==========show all record========
baoniu course:math 1635066093034 89
baoniu grade: 1635066093029 4
zkb course: 1635066093005 90
zkb course:art 1635066093023 87
zkb course:math 1635066093018 97
zkb grade: 1635066092993 5
==========del one record========
del recored baoniu ok.
zkb course: 1635066093005 90
zkb course:art 1635066093023 87
zkb course:math 1635066093018 97
zkb grade: 1635066092993 5
==========show all record========
zkb course: 1635066093005 90
zkb course:art 1635066093023 87
zkb course:math 1635066093018 97
zkb grade: 1635066092993 5
hadoop@hadoop2:~/hadoop/ejemplos/hbase$
```

Desde la consola Hbase también podemos acceder a la información de la tabla scores

```
hbase(main):018:0> scan "scores"
```

```
hbase(main):018:0> scan "scores"
ROW                                   COLUMN+CELL
 zkb                                  column=course:, timestamp=1635066093005, value=90
 zkb                                  column=course:art, timestamp=1635066093023, value=87
 zkb                                  column=course:math, timestamp=1635066093018, value=97
 zkb                                  column=grade:, timestamp=1635066092993, value=5
1 row(s) in 0.0360 seconds

hbase(main):019:0>
```