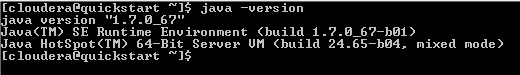


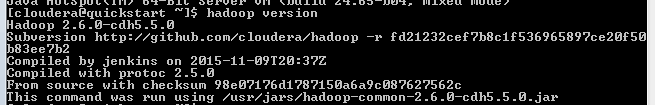
HIPI is written in Java and has been tested with Java 7 and 8. And Hadoop 2.6.0

Step 1 :

Check your version of Java with the following command:



Check your version of Hadoop with the following command:



Ensure that the Hadoop bin directory is in your system path

[cloudera@quickstart Desktop]$ which hadoop

/usr/bin/hadoop

Step 2: Gradle Setup :

1. download gradle from <http://gradle.org/gradle-download/>.
2. unzip zip file.

C:\Users\kshitish\Desktop\imageprocess\2.jpg

1. Move gradle to your installation directory

C:\Users\kshitish\Desktop\imageprocess\3.jpg

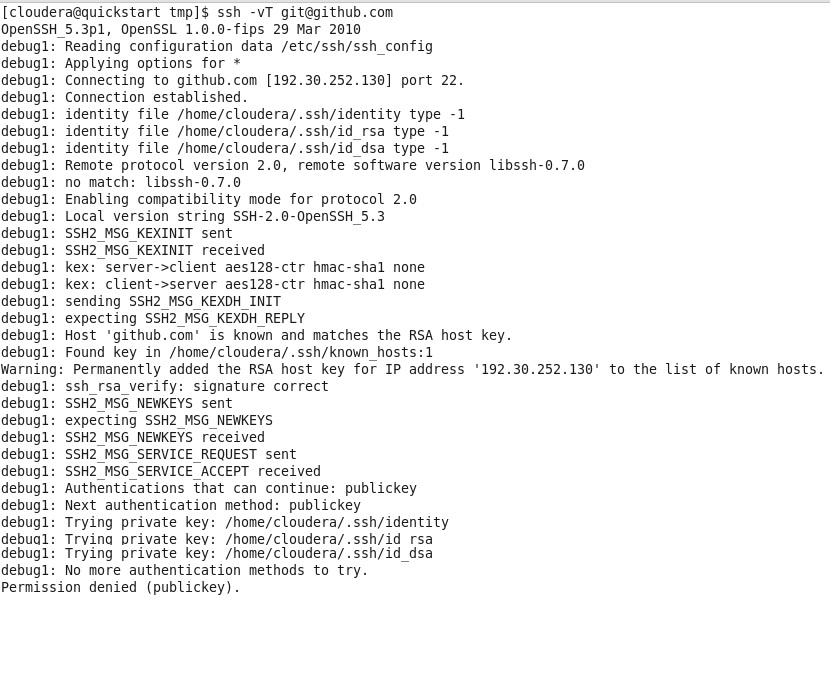
Step 3: GIT

C:\Users\kshitish\Desktop\imageprocess\6.jpg

Run below command for check you are able to ssh git or not

ssh -vT [git@github.com](mailto:git@github.com)

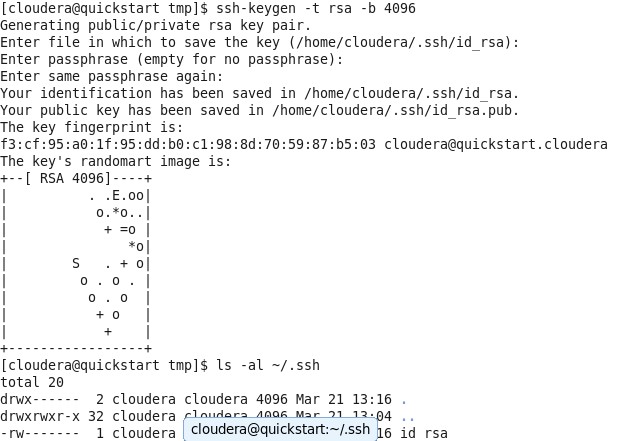
If you are getting error like permission denied as shown in below picture

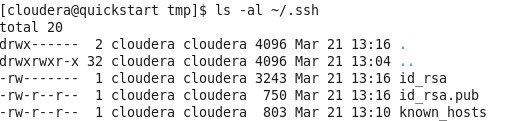


This means, on your local machine, you haven't made any SSH keys.

1. Solution: To create the SSH keys, type ssh-keygen -t rsa -C .

This will create both id\_rsa andid\_rsa.pub files.

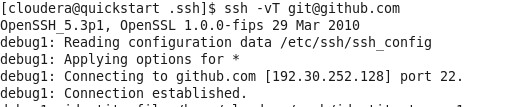


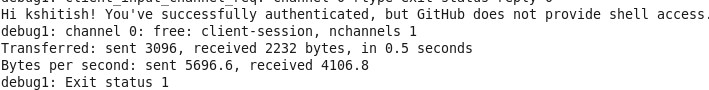


1. Now, go and open id\_rsa.pub in text editor.
2. Copy the contents--exactly as it appears, with no extra spaces or lines--of id\_rsa.pub and paste it into GitHub and/or BitBucket under the Account Settings > SSH Keys. **NOTE:** I like to give the SSH key a descriptive name, usually with the name of the workstation.
3. Now that you've added your public key to Github and/or BitBucket, try to git push again and see if it works. It should!

Test below command

# ssh -vT [git@github.com](mailto:git@github.com)



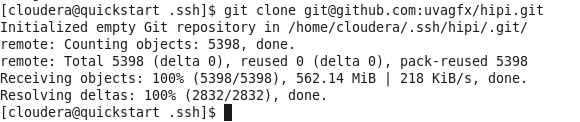


### Step 3: Clone the HIPI GitHub Repository

If it work fine then move to further steps to HIPI folder download or Clone the latest HIPI distribution from GitHub and build from source.

# git clone [git@github.com:uvagfx/hipi.git](mailto:git@github.com:uvagfx/hipi.git)

It will take little bit long time to download



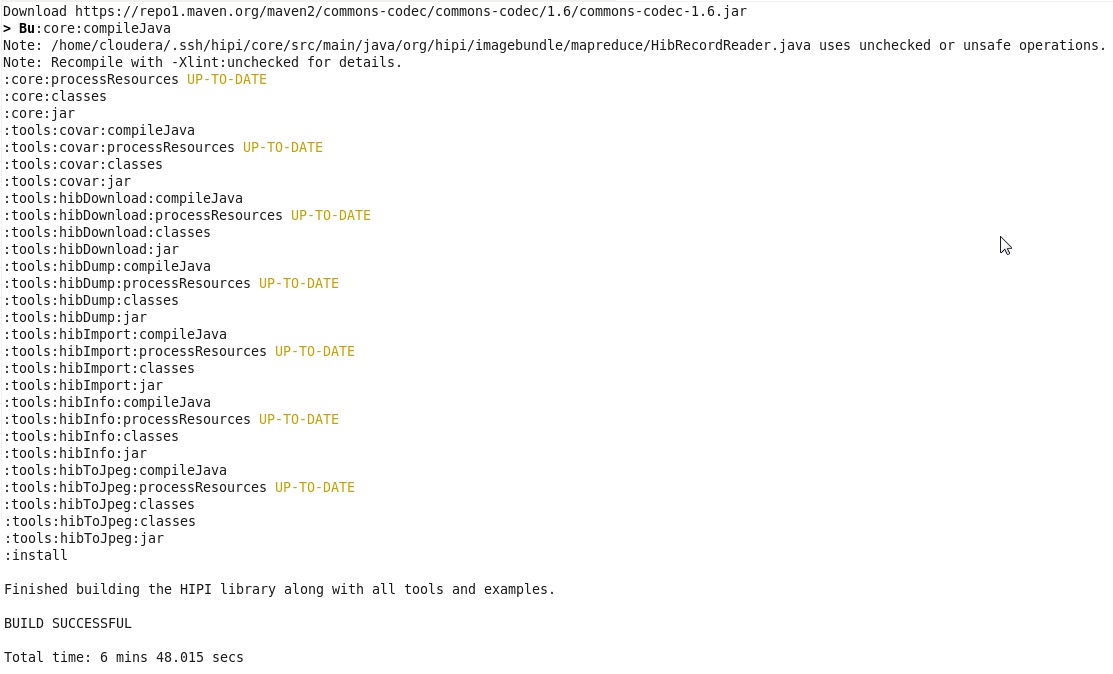
### Step 4: Build the HIPI Library

### C:\Users\kshitish\Desktop\imageprocess\10.jpg

Go to hipi folder

# cd hipi

C:\Users\kshitish\Desktop\imageprocess\11.jpg



# gradle

After the build finishes, you may want to inspect the settings.gradle file in the root directory and the build.gradle files in each directory to familiarize yourself with the various build targets.

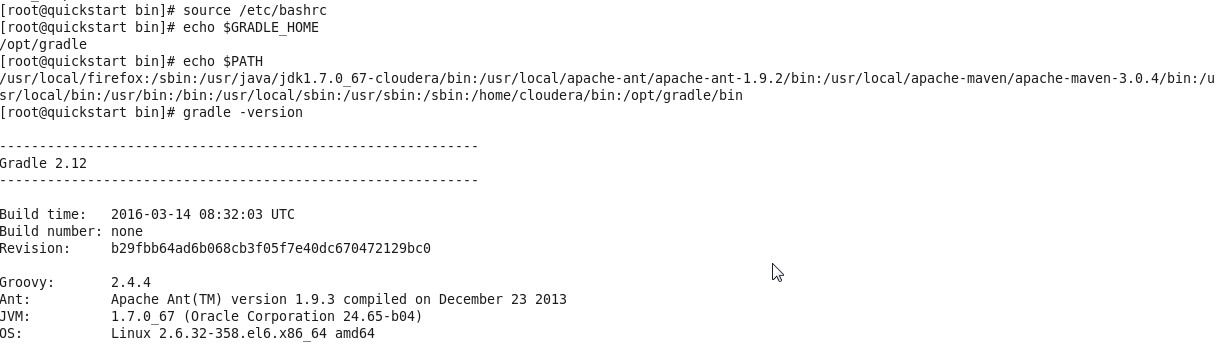
[cloudera@quickstart hipi]$ gradle clean tools:hibImport:jar

C:\Users\kshitish\Desktop\imageprocess\13.jpg



HIPI is now installed on your system.

Set Environment variable for Gradle



created image folder on desktop

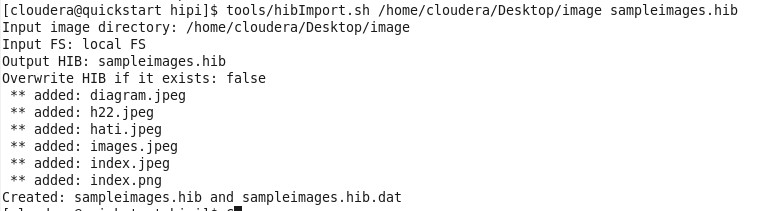
# mkdir –p /home/cloudera/Desktop/image

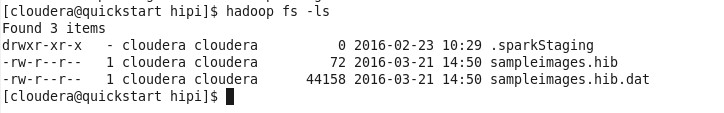
Then run below command

[cloudera@quickstart hipi]$ tools/hibImport.sh /home/cloudera/Desktop/image sampleimages.hib

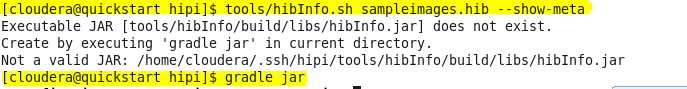
/home/cloudera/Desktop/image is Local System Path

sampleimages.hib – it will crate in HDFS





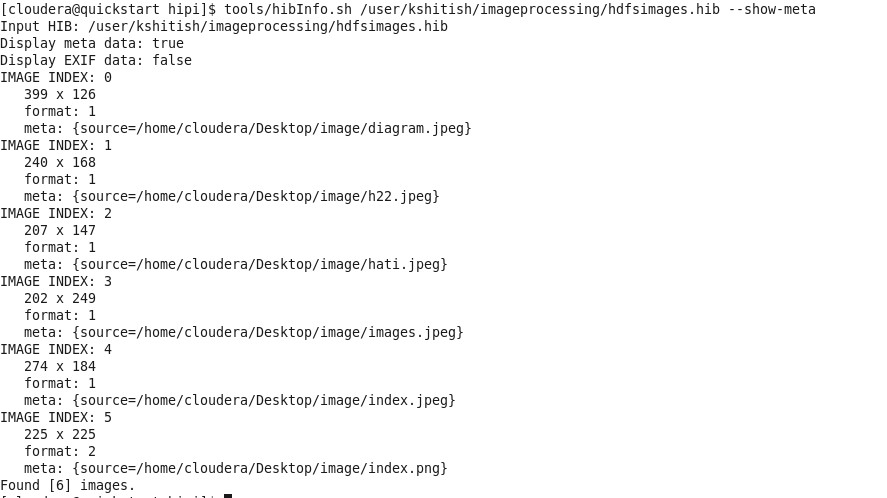
[cloudera@quickstart hipi]$ tools/hibInfo.sh sampleimages.hib --show-meta



If you are facing above issue then add your project name in settings.gradle.

After adding when you enter below command it wiil give

# tools/hibInfo.sh sampleimages.hib --show-meta



**#mkdir –p /kshitish/imageProcessing/src/main/java/org/hipi/examples/**

**#vi HelloWorld.java (enter below code)**

****

**HelloWorld.java**

package org.hipi.examples;

import org.hipi.image.FloatImage;

import org.hipi.image.HipiImageHeader;

import org.hipi.imagebundle.mapreduce.HibInputFormat;

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

public class HelloWorld extends Configured implements Tool {

public static class HelloWorldMapper extends Mapper<HipiImageHeader, FloatImage, IntWritable, FloatImage> {

public void map(HipiImageHeader key, FloatImage value, Context context)

throws IOException, InterruptedException {

// Verify that image was properly decoded, is of sufficient size, and has three color channels (RGB)

if (value != null && value.getWidth() > 1 && value.getHeight() > 1 && value.getNumBands() == 3) {

// Get dimensions of image

int w = value.getWidth();

int h = value.getHeight();

// Get pointer to image data

float[] valData = value.getData();

// Initialize 3 element array to hold RGB pixel average

float[] avgData = {0,0,0};

// Traverse image pixel data in raster-scan order and update running average

for (int j = 0; j < h; j++) {

for (int i = 0; i < w; i++) {

avgData[0] += valData[(j\*w+i)\*3+0]; // R

avgData[1] += valData[(j\*w+i)\*3+1]; // G

avgData[2] += valData[(j\*w+i)\*3+2]; // B

}

}

// Create a FloatImage to store the average value

FloatImage avg = new FloatImage(1, 1, 3, avgData);

// Divide by number of pixels in image

avg.scale(1.0f/(float)(w\*h));

// Emit record to reducer

context.write(new IntWritable(1), avg);

} // If (value != null...

} // map()

} // HelloWorldMapper

public static class HelloWorldReducer extends Reducer<IntWritable, FloatImage, IntWritable, Text> {

public void reduce(IntWritable key, Iterable<FloatImage> values, Context context)

throws IOException, InterruptedException {

// Create FloatImage object to hold final result

FloatImage avg = new FloatImage(1, 1, 3);

// Initialize a counter and iterate over IntWritable/FloatImage records from mapper

int total = 0;

for (FloatImage val : values) {

avg.add(val);

total++;

}

if (total > 0) {

// Normalize sum to obtain average

avg.scale(1.0f / total);

// Assemble final output as string

float[] avgData = avg.getData();

String result = String.format("Average pixel value: %f %f %f", avgData[0], avgData[1], avgData[2]);

// Emit output of job which will be written to HDFS

context.write(key, new Text(result));

}

} // reduce()

} // HelloWorldReducer

public int run(String[] args) throws Exception {

// Check input arguments

if (args.length != 2) {

System.out.println("Usage: helloWorld <input HIB> <output directory>");

System.exit(0);

}

// Initialize and configure MapReduce job

Job job = Job.getInstance();

// Set input format class which parses the input HIB and spawns map tasks

job.setInputFormatClass(HibInputFormat.class);

// Set the driver, mapper, and reducer classes which express the computation

job.setJarByClass(HelloWorld.class);

job.setMapperClass(HelloWorldMapper.class);

job.setReducerClass(HelloWorldReducer.class);

// Set the types for the key/value pairs passed to/from map and reduce layers

job.setMapOutputKeyClass(IntWritable.class);

job.setMapOutputValueClass(FloatImage.class);

job.setOutputKeyClass(IntWritable.class);

job.setOutputValueClass(Text.class);

// Set the input and output paths on the HDFS

FileInputFormat.setInputPaths(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

// Execute the MapReduce job and block until it complets

boolean success = job.waitForCompletion(true);

// Return success or failure

return success ? 0 : 1;

}

public static void main(String[] args) throws Exception {

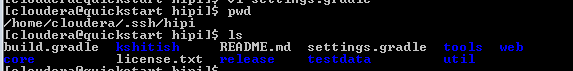
ToolRunner.run(new HelloWorld(), args);

System.exit(0);

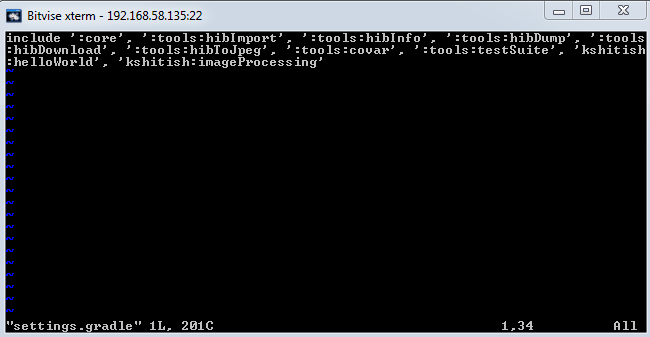
}

}

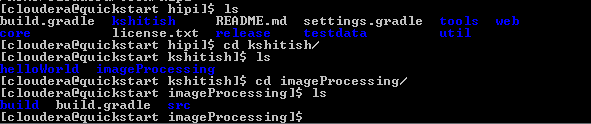
**Add imageProcessing folder in setting.gradle file**







**Then modify on build.gradle file inside your project folder directory as shown in below picture. imageProcessing is my project directory.**



####################################################################

if you are facing error in gradle jar command

Error

###################################################################

[cloudera@quickstart imageProcessing]$ gradle jar

:kshitish:imageProcessing:compileJava

/home/cloudera/.ssh/hipi/kshitish/imageProcessing/src/main/java/org/hipi/examples/HelloWorld.java:3: error: package org.hipi.image does not exist

import org.hipi.image.FloatImage;

^

/home/cloudera/.ssh/hipi/kshitish/imageProcessing/src/main/java/org/hipi/examples/HelloWorld.java:4: error: package org.hipi.image does not exist

import org.hipi.image.HipiImageHeader;

^

/home/cloudera/.ssh/hipi/kshitish/imageProcessing/src/main/java/org/hipi/examples/HelloWorld.java:5: error: package org.hipi.imagebundle.mapreduce does not exist

import org.hipi.imagebundle.mapreduce.HibInputFormat;

^

/home/cloudera/.ssh/hipi/kshitish/imageProcessing/src/main/java/org/hipi/examples/HelloWorld.java:25: error: cannot find symbol

public static class HelloWorldMapper extends Mapper<HipiImageHeader, FloatImage, IntWritable, FloatImage> {

^

symbol: class HipiImageHeader

location: class HelloWorld

/home/cloudera/.ssh/hipi/kshitish/imageProcessing/src/main/java/org/hipi/examples/HelloWorld.java:25: error: cannot find symbol

public static class HelloWorldMapper extends Mapper<HipiImageHeader, FloatImage, IntWritable, FloatImage> {

^

symbol: class FloatImage

location: class HelloWorld

/home/cloudera/.ssh/hipi/kshitish/imageProcessing/src/main/java/org/hipi/examples/HelloWorld.java:25: error: cannot find symbol

public static class HelloWorldMapper extends Mapper<HipiImageHeader, FloatImage, IntWritable, FloatImage> {

^

symbol: class FloatImage

location: class HelloWorld

/home/cloudera/.ssh/hipi/kshitish/imageProcessing/src/main/java/org/hipi/examples/HelloWorld.java:27: error: cannot find symbol

public void map(HipiImageHeader key, FloatImage value, Context context)

^

symbol: class HipiImageHeader

location: class HelloWorldMapper

/home/cloudera/.ssh/hipi/kshitish/imageProcessing/src/main/java/org/hipi/examples/HelloWorld.java:27: error: cannot find symbol

public void map(HipiImageHeader key, FloatImage value, Context context)

^

symbol: class FloatImage

location: class HelloWorldMapper

/home/cloudera/.ssh/hipi/kshitish/imageProcessing/src/main/java/org/hipi/examples/HelloWorld.java:67: error: cannot find symbol

public static class HelloWorldReducer extends Reducer<IntWritable, FloatImage, IntWritable, Text> {

^

symbol: class FloatImage

location: class HelloWorld

/home/cloudera/.ssh/hipi/kshitish/imageProcessing/src/main/java/org/hipi/examples/HelloWorld.java:69: error: cannot find symbol

public void reduce(IntWritable key, Iterable<FloatImage> values, Context context)

^

symbol: class FloatImage

location: class HelloWorldReducer

/home/cloudera/.ssh/hipi/kshitish/imageProcessing/src/main/java/org/hipi/examples/HelloWorld.java:53: error: cannot find symbol

FloatImage avg = new FloatImage(1, 1, 3, avgData);

^

symbol: class FloatImage

location: class HelloWorldMapper

/home/cloudera/.ssh/hipi/kshitish/imageProcessing/src/main/java/org/hipi/examples/HelloWorld.java:53: error: cannot find symbol

FloatImage avg = new FloatImage(1, 1, 3, avgData);

^

symbol: class FloatImage

location: class HelloWorldMapper

/home/cloudera/.ssh/hipi/kshitish/imageProcessing/src/main/java/org/hipi/examples/HelloWorld.java:73: error: cannot find symbol

FloatImage avg = new FloatImage(1, 1, 3);

^

symbol: class FloatImage

location: class HelloWorldReducer

/home/cloudera/.ssh/hipi/kshitish/imageProcessing/src/main/java/org/hipi/examples/HelloWorld.java:73: error: cannot find symbol

FloatImage avg = new FloatImage(1, 1, 3);

^

symbol: class FloatImage

location: class HelloWorldReducer

/home/cloudera/.ssh/hipi/kshitish/imageProcessing/src/main/java/org/hipi/examples/HelloWorld.java:77: error: cannot find symbol

for (FloatImage val : values) {

^

symbol: class FloatImage

location: class HelloWorldReducer

/home/cloudera/.ssh/hipi/kshitish/imageProcessing/src/main/java/org/hipi/examples/HelloWorld.java:106: error: cannot find symbol

job.setInputFormatClass(HibInputFormat.class);

^

symbol: class HibInputFormat

location: class HelloWorld

/home/cloudera/.ssh/hipi/kshitish/imageProcessing/src/main/java/org/hipi/examples/HelloWorld.java:113: error: cannot find symbol

job.setMapOutputValueClass(FloatImage.class);

^

symbol: class FloatImage

location: class HelloWorld

17 errors

:kshitish:imageProcessing:compileJava FAILED

FAILURE: Build failed with an exception.

\* What went wrong:

Execution failed for task ':kshitish:imageProcessing:compileJava'.

> Compilation failed; see the compiler error output for details.

\* Try:

Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output.

BUILD FAILED

Total time: 1 mins 44.588 secs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SOLUTION

Add dependencies{

compile project(':core')

}

in below code

jar {

manifest {

attributes("Main-Class": "org.hipi.examples.HelloWorld")

}

dependencies{

compile project(':core')

}

}

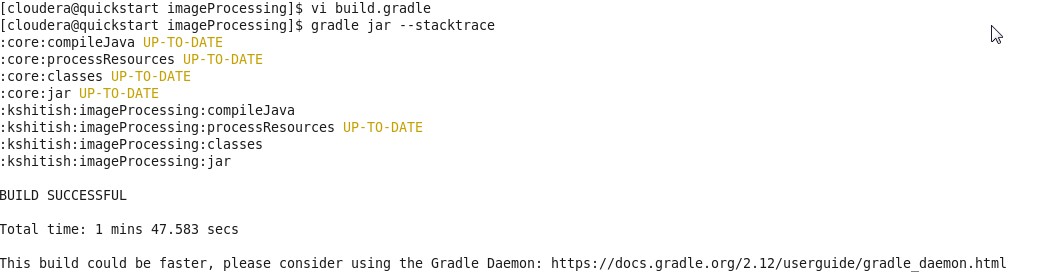
then it will get resolved

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Now build helloWorld.jar and run it using the HIB we created at the beginning:

#gradle jar





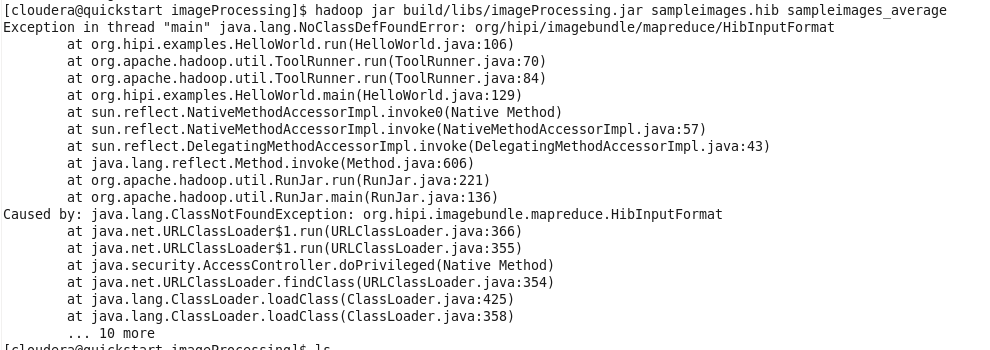
Then run below command for run Mapreduce job to process image

# hadoop jar build/libs/imageProcessing.jar sampleimages.hib sampleimages\_average

sampleimages.hib - > is from HDFS which we have created previously

sampleimages\_average-> is directory name in HDFS where we want to keep our output

After running this command if you are facing below error then



To resolve above issue we have to add below line in build.gradle file

from {

configurations.compile.collect { it.isDirectory() ? it : zipTree(it) }

}

Now build.gradle file is like

jar {

manifest {

attributes("Main-Class": "org.hipi.examples.HelloWorld")

}

dependencies{

compile project(':core')

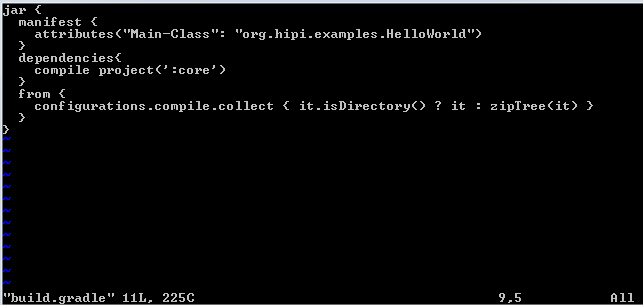
}

from {

configurations.compile.collect { it.isDirectory() ? it : zipTree(it) }

}

}



After running above command sampleimages\_average will contain two files:

hadoop fs -ls sampleimages\_average

Found 2 items

-rw-r--r-- 1 user group 0 2016-03-23 03:52 sampleimages\_average/\_SUCCESS

-rw-r--r-- 1 user group 50 2016-03-23 03:52 sampleimages\_average/part-r-00000

Whenever a MapReduce program successfully finishes, it creates the file \_SUCCESS in the output directory along with a part-r-XXXXX file for each reduce task. The average pixel value can be retrieved using the cat command:

# hadoop fs -cat sampleimages\_average/part-r-00000





Refference Link: <http://hipi.cs.virginia.edu/gettingstarted.html>