

## Overview

The package installs HDP2.6 + HiBench package to assist in benchmark tests of various environments and settings. The installer assumes Centos-7 installed on all participating nodes.

## Package Content

- `hdp_2.6_install.sh` – script to install HDP either as single node or cluster
- `HiBench_install_config_generator.pl` – CFG file generator for `HiBench_install.sh` installer
- `HiBench_install.sh` – script to install ML section from the HiBench Package.

## Short setup instructions

```
# Download the installer
yum install git -y
git clone https://github.com/lovenugulu/hdp26_and_hibench_install.git
cd hdp26_and_hibench_install

# On the master node run the following line.
# To install on a single node run without parameters.
./hdp_2.6_install.sh [slave1 slave2 ... ]

# Generate CFG file for installing HiBench
./HiBench_install_config_generator.pl

# Review the config file. Edit it if needed.
cat HiBench_install.cfg

# Once satisfied, run the installer:
./HiBench_install.sh

# to run tests, login as "hdfs":
su - hdfs

# prepare test sample:
/opt/HiBench/bin/workloads/ml/kmeans/prepare/prepare.sh

# run a test:
/opt/HiBench/bin/workloads/ml/kmeans/spark/run.sh
```

## Detailed Instructions

### 1. Install HDP using `hdp_2.6_install.sh`

The script installs the latest HDP.2.6 environment on the host it run on. To install slaves, write the hostnames of the slaves in the command line.

Example:

```
# installing master + three slaves:

./hdp_2.6_install.sh  slave1 slave2 slave3
```

Once the installer completes, one may monitor the installation process using by pointing the browser to the master at port 8080. Example: <http://masterhost:8080/>  
The default username/password is: admin/admin.

Once the install is done, review the default installed configuration and change if needed.  
See Instructions below in bullet 5 (Set YARN memory properties) below.

2. Generate CFG file by running: `./HiBench_install_config_generator.pl`

The config file generated should look like the following:

```
[root@hdp02 ~]# cat HiBench_install.cfg
# HIBENCH_SCALE_PROFILE - Available value is tiny, small, large, huge, gigantic and bigdata
HIBENCH_SCALE_PROFILE= gigantic
```

```
# executor number and cores when running on Yarn
HIBENCH_YARN_EXECUTOR_NUM=25
HIBENCH_YARN_EXECUTOR_CORES=5
```

```
# executor and driver memory in standalone & YARN mode
SPARK_EXECUTOR_MEMORY=35g
SPARK_DRIVER_MEMORY=2g
SPARK_YARN_EXECUTOR_MEMORYOVERHEAD=2500
SPARK_YARN_DRIVER_MEMORYOVERHEAD=400
SPARK_MEMORY_OFFHEAP_SIZE=1024m
```

```
# Be sure to configure here the correct directory. (remember that spark and spark2 are NOT in
the same directory)
HIBENCH_SPARK_HOME=/usr/hdp/2.6.4.0-91/spark2
#-----
```

The user may edit the file carefully. Carefully means not to “beautify” the file by padding spaces or changing to invalid values.

Instructions for tuning spark is out of the scope of these instructions.

3. Run the HiBench installer: `./HiBench_install.sh`

The Installer performs the following:

- a. Download the HiBench package from github
- b. Download Oracle's Java JDK
- c. Download MVN
- d. Compile/Install the needed JARS for spark1.6 or spark2 – according the settings in the CFG file.
- e. Patch the HiBench package configuration file according the settings in the CFG file.

4. Run K-Means benchmark test:

# login to hdfs:

**su - hdfs**

# prepare test sample:

**/opt/HiBench/bin/workloads/ml/kmeans/prepare/prepare.sh**

# run a test:

**/opt/HiBench/bin/workloads/ml/kmeans/spark/run.sh**

Once the test (or prepare) is running, it can be monitored with the TBD as follows TBD.

If error occurs, log lines like the following shows:

```
ERROR: Spark job com.intel.hibench.sparkbench.ml.DenseKMeans failed to run successfully.
```

```
Hint: You can goto /opt/HiBench/report/kmeans/spark/conf/../../bench.log to check for detailed log.
```

```
Opening log tail for you:
```

```
at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at
org.apache.spark.deploy.SparkSubmit$.org$apache$spark$deploy$SparkSubmit$$runMain(SparkSubmit.scala:782)
at
org.apache.spark.deploy.SparkSubmit$.doRunMain$1(SparkSubmit.scala:180)
at org.apache.spark.deploy.SparkSubmit$.submit(SparkSubmit.scala:205)
at org.apache.spark.deploy.SparkSubmit$.main(SparkSubmit.scala:119)
at org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)
```

Most likely the “tailed log” is useless. Follow the “hint” in the lines above and open the detailed log at:

```
/opt/HiBench/report/kmeans/spark/conf/../../bench.log
```

The log (in this example) shows the following error:

```
18/05/14 07:56:19 ERROR SparkContext: Error initializing SparkContext.  
java.lang.IllegalArgumentException: Required executor memory (35+2.5  
GB) is above the max threshold (5120 MB) of this cluster! Please check  
the values of 'yarn.scheduler.maximum-allocation-mb' and/or  
'yarn.nodemanager.resource.memory-mb'.
```

This error is because the job requires more memory than the default YARN settings. Change YARN memory properties and repeat the test.

##### 5. Set YARN memory properties

The default YARN properties are set by default for minimal values.

If you are familiar with Hadoop and its setting, set it up. If not, follow the instruction below for setting “good enough” configuration:

The screenshot displays the Ambari web interface for configuring YARN. On the left sidebar, the 'YARN' service is selected (marked with a red '1'). The top navigation bar shows the 'Configs' tab is active (marked with a red '2'). Below this, a configuration group 'Default (3)' is selected. A summary bar shows 'V2' configuration for 'internal' (marked with a green checkmark and a red '3') and 'admin' (marked with a red '4'). A 'Save' button is visible (marked with a red '5'). The main content area is divided into 'Memory' and 'YARN Features' sections. The 'Memory' section includes a 'Node' memory bar (marked with a red '3') and 'Container' memory settings for 'Minimum Container Size (Memory)' and 'Maximum Container Size (Memory)', both marked with a red '4'. The 'YARN Features' section shows 'Node Labels' and 'Pre-emption' both set to 'Disabled'.

The Number in parenthesis in the instructions below refer to the red number in the picture above.

- Open the “Yarn config” tab by clicking on “YARN” (1) and then clicking “configs” (2)
- Adjust the memory by clicking on the Node memory bar near the recommendation mark (3)

- c. Verify that the “Maximum container Size” adjusted accordingly to the same value (4)
- d. Press “save” (5)
- e. Approve the change by clicking on “Save” once again

Dependent Configuration screen opens – click on the green “OK” button:

Dependent Configurations

Recommended Changes

Based on your configuration changes, Ambari is recommending the following dependent configuration changes. Ambari will update all checked configuration changes to the **Recommended Value**. Uncheck any configuration to retain the **Current Value**.

<input checked="" type="checkbox"/>	Property	Service	Config Group	File Name	Current Value	Recommended Value
<input checked="" type="checkbox"/>	yarn.scheduler.maximum-allocation-mb	YARN	Default	yarn-site	5120	12288
<input checked="" type="checkbox"/>	hive.auto.convert.join.noconditionaltask.size	Hive	Default	hive-site	52428800	1145324612
<input checked="" type="checkbox"/>	hive.tez.container.size	Hive	Default	hive-site	682	4096
<input checked="" type="checkbox"/>	mapreduce.map.java.opts	MapReduce2	Default	mapred-site	-Xmx410m	-Xmx3276m
<input checked="" type="checkbox"/>	mapreduce.reduce.java.opts	MapReduce2	Default	mapred-site	-Xmx756m	-Xmx6553m
<input checked="" type="checkbox"/>	yarn.app.mapreduce.am.command-opts	MapReduce2	Default	mapred-site	-Xmx410m	-Xmx3276m -Dhdp.version=\${hdp.version}
<input checked="" type="checkbox"/>	yarn.app.mapreduce.am.resource.mb	MapReduce2	Default	mapred-site	512	4096
<input checked="" type="checkbox"/>	mapreduce.map.memory.mb	MapReduce2	Default	mapred-site	512	4096
<input checked="" type="checkbox"/>	mapreduce.reduce.memory.mb	MapReduce2	Default	mapred-site	1024	8192

Cancel

OK

Configuration warning appears. Click on the “Proceed anyway” red button:

Configurations

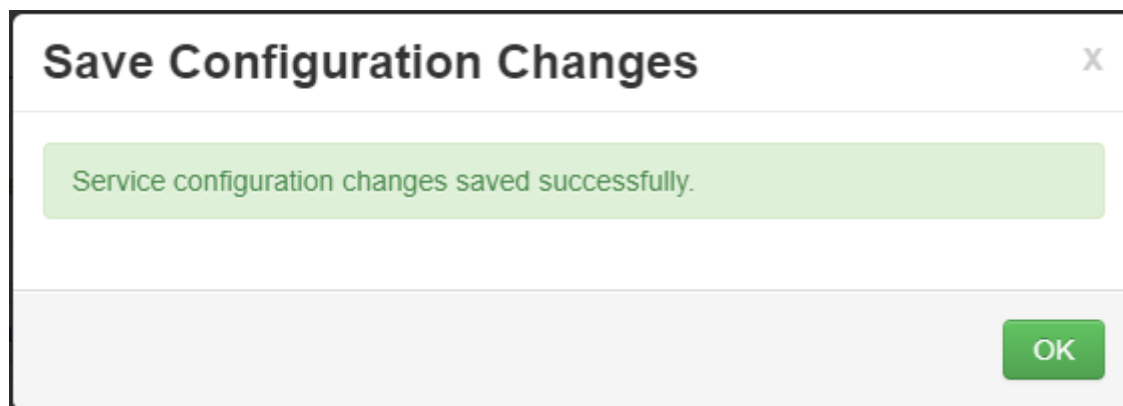
The following configuration changes are highly recommended, but can be skipped.

Type	Service	Property	Value	Description
Error	Tez	tez.tez-ui.history-url.base		Value should be set for tez.tez-ui.history-url.base
Warning	YARN	yarn.scheduler.maximum-allocation-vcores	8	Value is greater than the recommended maximum of 3
Warning	HDFS	dfs.datanode.du.reserved	1073741824	Value is less than the recommended default of 3219522048 Reserved space in bytes per volume. Always leave this much space free for non dfs use.
Warning	Tez	tez.am.resource.memory.mb	1536	Value is less than the recommended default of 4096 The amount of memory to be used by the AppMaster. Used only if the value is not specified explicitly by the DAG definition.

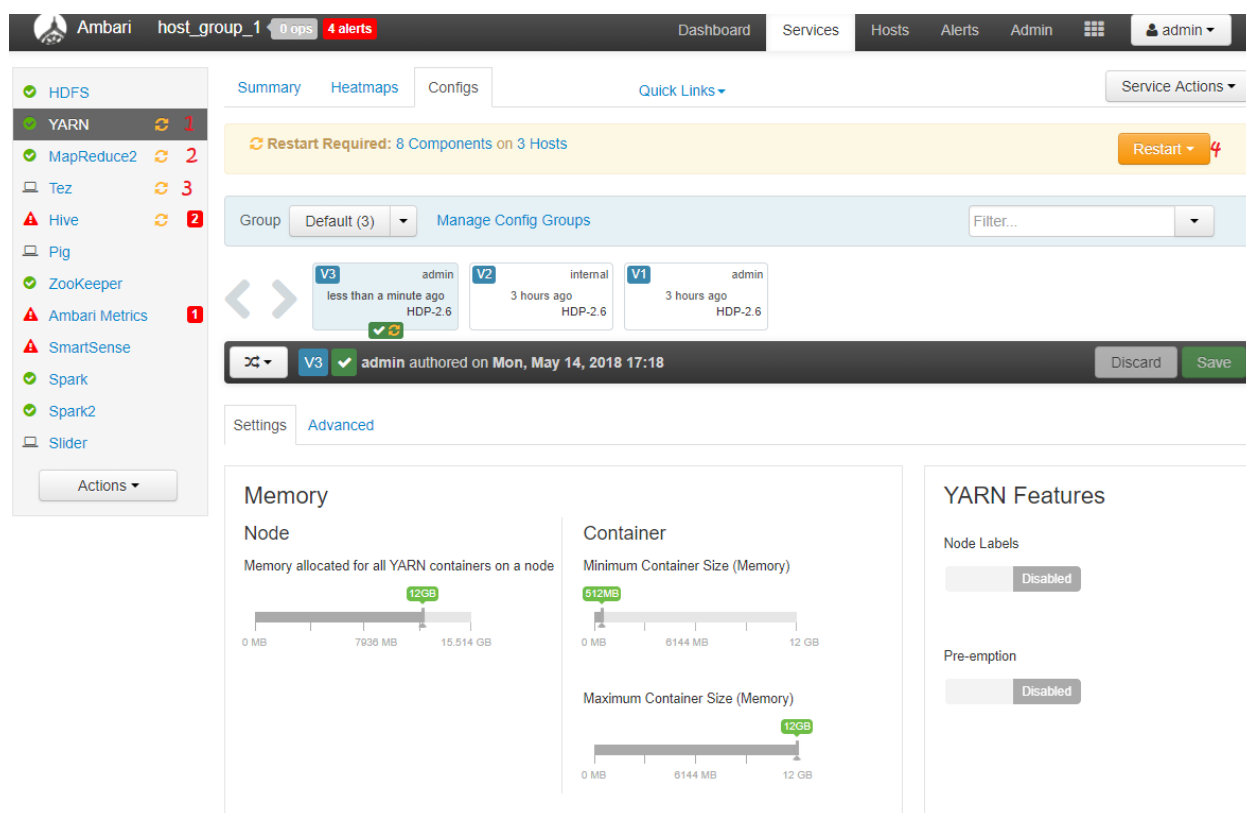
Cancel

Proceed Anyway

“Save Configuration Changes” notice appears. Click on the “OK” green button:



The “Services” page open once again. The yellow “Restart” button indicates that few services requires restart for the new setup to be in effect:



To restart, click on a service to restart. In the picture above, “YARN” is selected. Once service is selected, click on the yellow “Restart” button. (It is not mandatory to restart “Hive” as this test doesn’t use it.

Your system should now be ready.

Happy Benchmarking....