Overview

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

For more info on HDP please consult: https://docs.hortonworks.com/

For more info on HiBench please consult: https://github.com/intel-hadoop/HiBench

Package Content

- hdp_2.6_install.sh – script to install HDP, either as a 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

`Please refer to the instructions at the README:`
`https://github.com/lovengulu/hdp26_and_hibench_install`

Detailed Instructions

1. **Install HDP using hdp_2.6_install.sh**
   This script installs the latest HDP.2.6 environment on the host it is 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 by pointing the browser to the master at port 8080. Example: http://masterhost:8080/
   The default username/password is: admin/admin.

   Once the installation is done, review the default installed configuration and change according your specific needs. Minimal but mandatory setup change is described 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 values are 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 the correct directory here. (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 carefully edit the file. Carefully means not to "beautify" the file by adding 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 to the settings in the CFG file.
   e. Patch the HiBench package configuration file according to the settings in the CFG file.

4. Run K-Means benchmark test:

   GENERAL COMMENT:

   The HiBench-KMenas prepare.sh & run.sh complains once started about the authenticity of the host. **No need to do anything as the scripts continues by itself after few seconds.**

```
# 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 preparation) is running, it can be monitored with the TBD as follows TBD.

If an error occurs, log the lines as shown below:

```
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$$run
Main(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
```

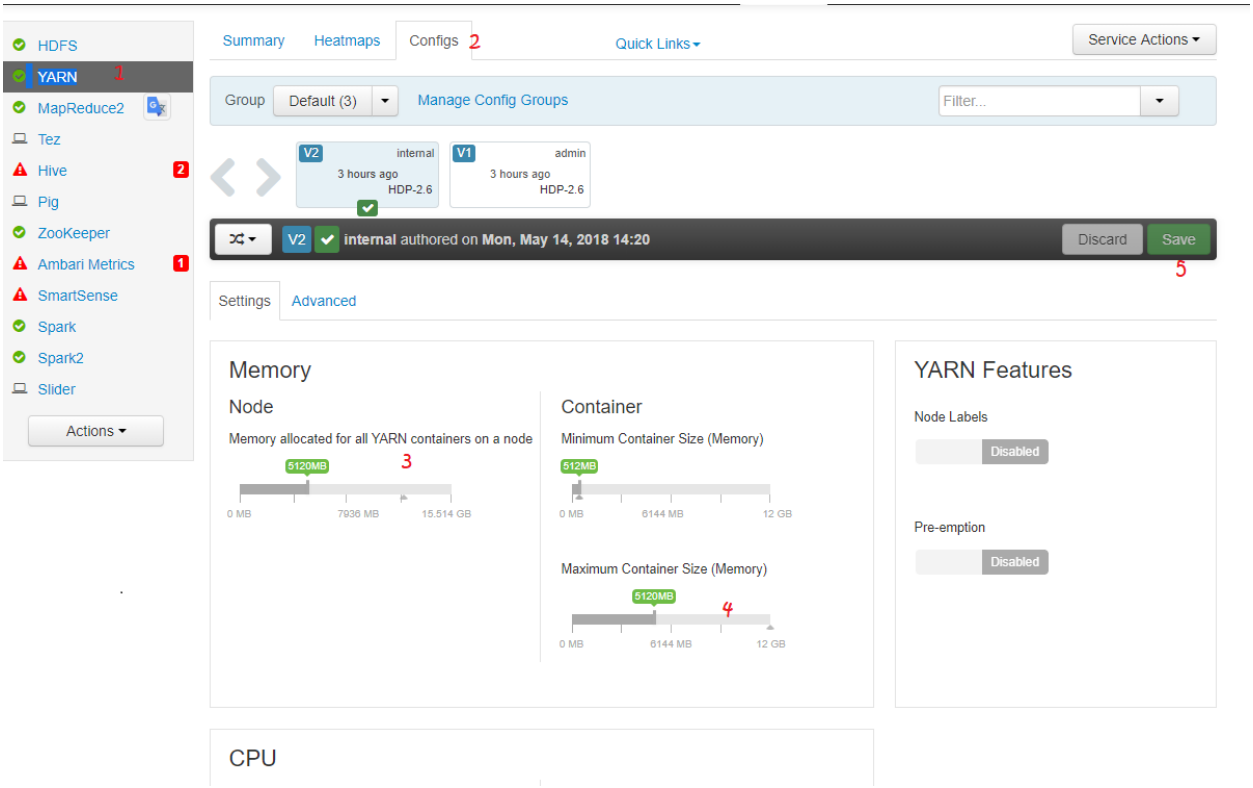
The log (in this example) shows the following error:

```
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 what is set in the default YARN settings. Change the YARN memory properties and repeat the test.

5. Set YARN memory properties
   The default YARN properties are set by default for minimal values.

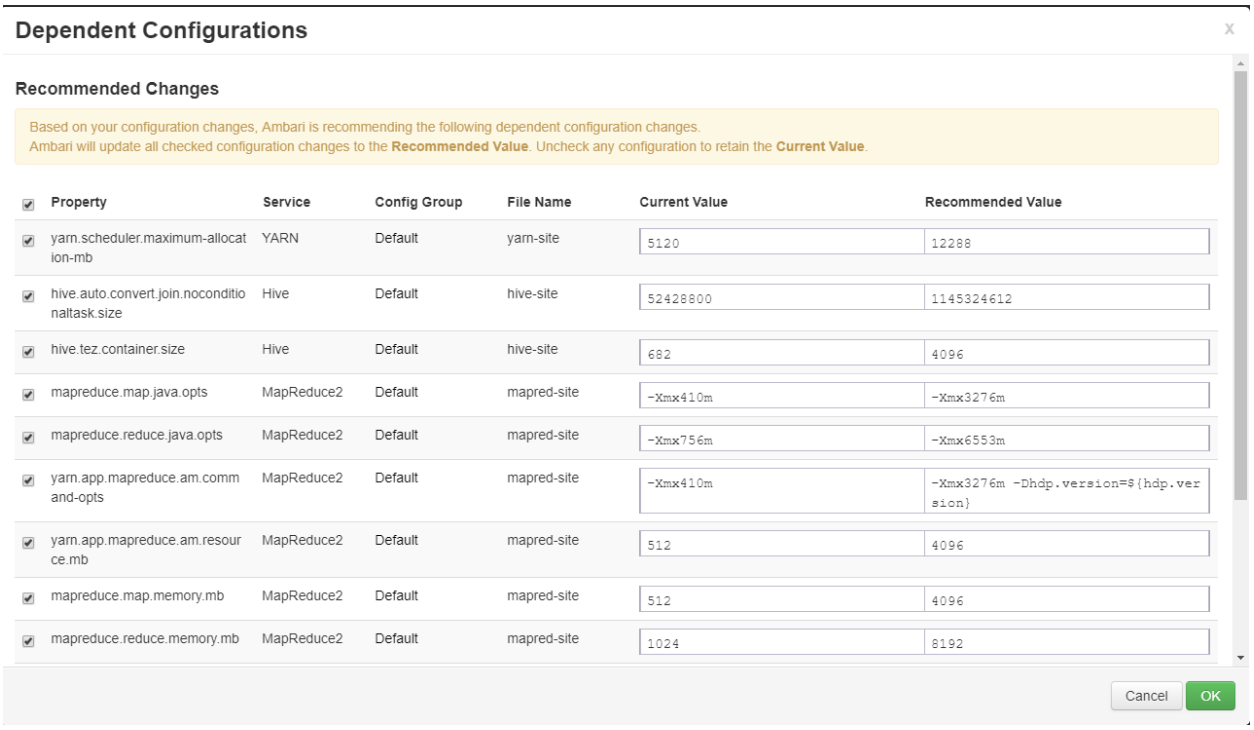
If you are familiar with Hadoop and its setting, set it up. If not, follow the instructions below for setting a “good enough” configuration:



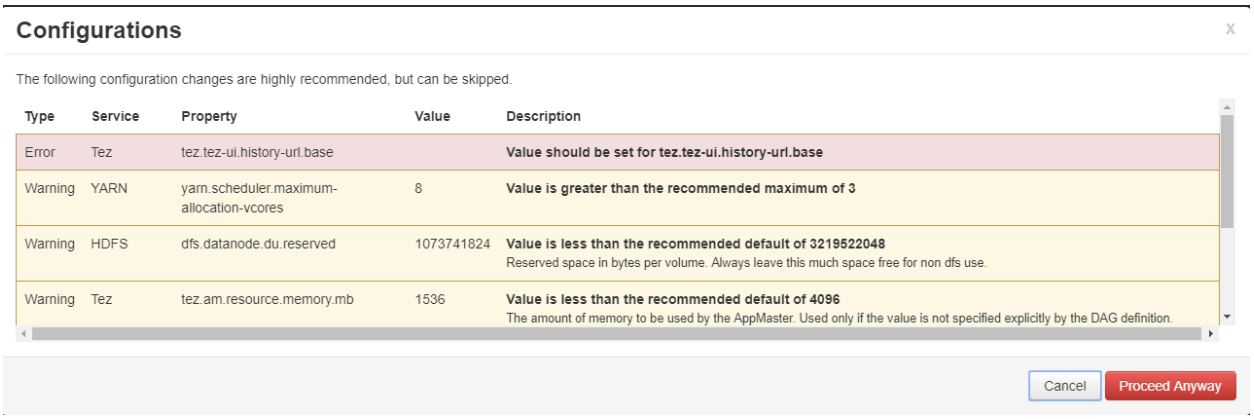
The Number in the parenthesis, in the instructions below, refers to the red number in the image above.

a. Open the “Yarn config” tab by clicking on “YARN” (1) and then clicking “configs” (2)
b. Adjust the memory by clicking on the Node memory bar near the recommendation mark (3)
c. Verify that the “Maximum container Size” is adjusted accordingly to the same value (4)
d. Press “save” (5)
e. Approve the change by clicking on “Save” once again

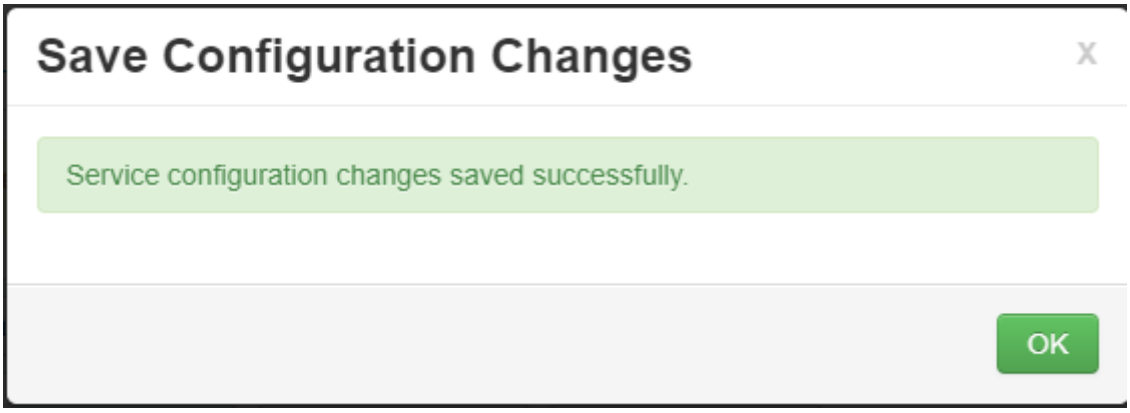
Dependent Configuration screen opens – click on the green “OK” button:



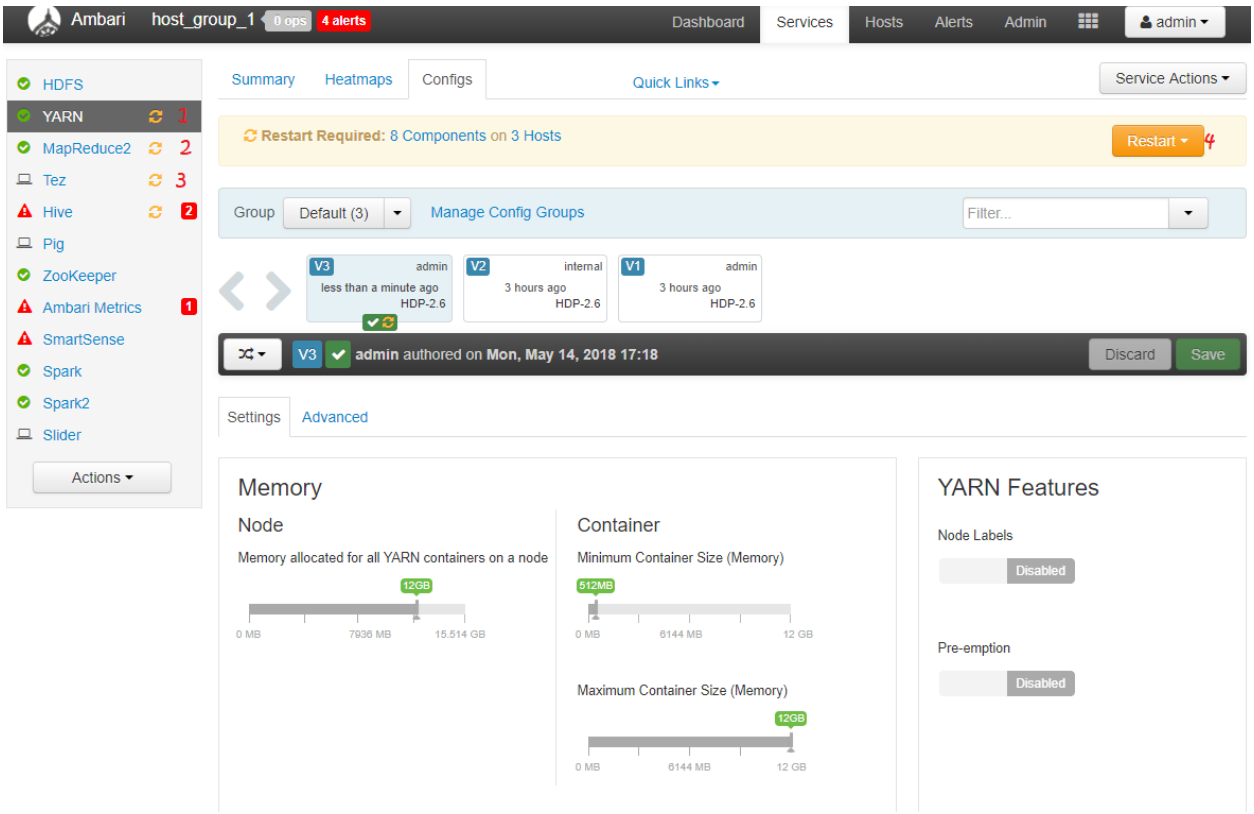
A configuration warning appears. Click on the red “Proceed Anyway” button:



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



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



To restart, click on a service. In the image above, “YARN” is selected. Once the service is selected, click on the yellow “Restart” button; select: “Restart All Affected” when asked. You may do the same for the other services that the Ambari-GUI It is not mandatory to restart “Hive” as this test doesn’t use it.)

Your system should now be ready.

Happy Benchmarking….