Apache HBase 2.6 Installation on Windows OS

Table of Contents

This document outlines the steps needed to install **Apache HBase 2.6** on Windows Operating system and access HBase database from **Apache Hive**.

1. Overview:

Apache HBase is the Hadoop database which is a distributed, scalable storage system for Big Data. It is an open source non-relational database modeled after Google's Bigtable to store and fast access the large amount of Big Data on top of HDFS.

Google's **Bigtable** is a distributed storage system provided by Google File System for managing structured data of petabytes size across thousands of commodity servers whereas **Apache HBase** is a distributed storage system for managing non-relational data (structured, semi-structured and unstructured data) on top of Hadoop Distributed File System.

HBase can be installed in three different modes:

1. **Standalone Mode:** In Standalone Mode, HBase daemons – `HMaster`, `HRegionServer` and `ZooKeeper` – runs within a single JVM (Java Virtual Machine) process /instance on a single host with HBase data stored on local file system. This is the most basic deployment mode.

2. **Pseudo-Distributed Mode**: In this mode, HBase still runs completely on a single host but each HBase daemon such as `HMaster`, `HRegionServer` and `ZooKeeper` runs as a separate process. This mode allows us to configure HBase data storage either on local file system or on HDFS. This deployment mode is for testing purposes only but not for Production.

3. **Fully-Distributed Mode:** In this mode, HBase cluster contains multiple nodes or hosts and each node can run one or more HBase daemons including primary and backup `Master` instances, multiple `RegionServer` nodes and multiple `ZooKeeper` nodes. This mode can only run on top of HDFS which makes a fully distributed configuration for Production and is used in real-time scenarios.

2. Prerequisites:

The following prerequisites need to be installed before running HBase.

1. **File Archiver:** Any file archiver such as **7zip** or **WinRAR** is needed to unzip the downloaded Spark binaries. 7zip can be downloaded from the 7zip Downloads website and WinRAR can be downloaded from the RAR lab Downloads website.

2. **JRE 8:** HBase 3.x requires Java 8 runtime environment *(HBase 2.3+ supports Java 8 mainly and Java 11 with limited capability. Other Java versions are not supported).* See the HBase Java support page for more details.

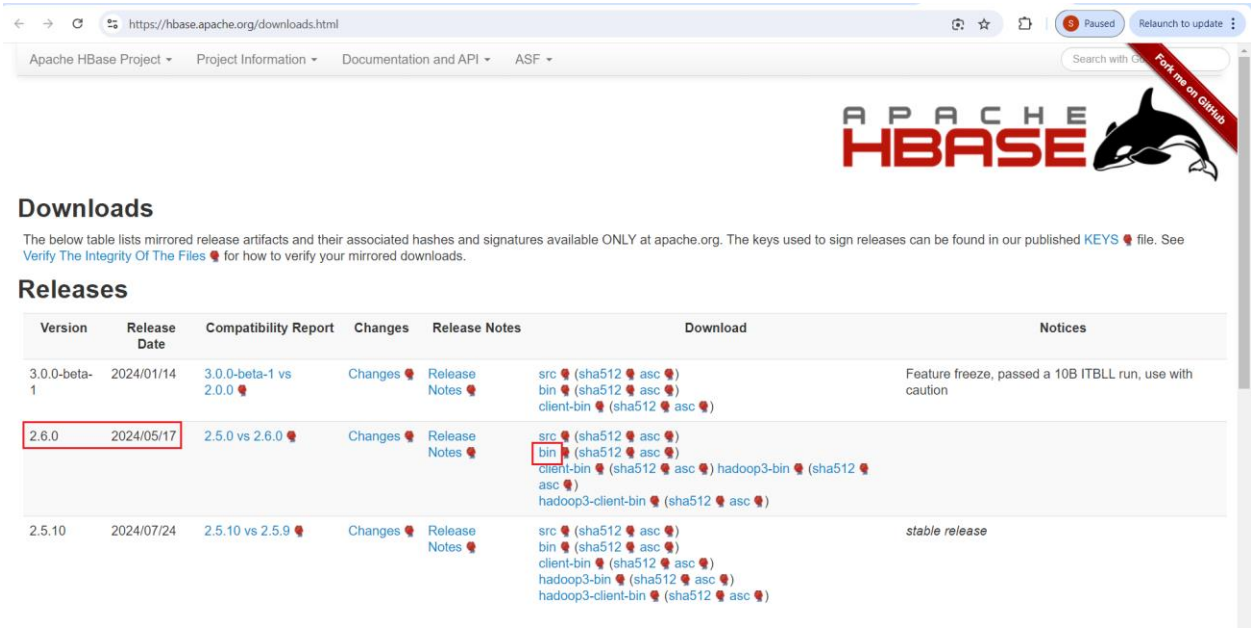
   We can either download just JRE 8 (Java Runtime Environment) for Windows offline installation from the official Java Download for Windows Offline website or download the whole JDK 8 (Java Development Kit) directly from Oracle Java Downloads website. For the complete JDK installation steps, look at here.

3. Install Standalone HBase:

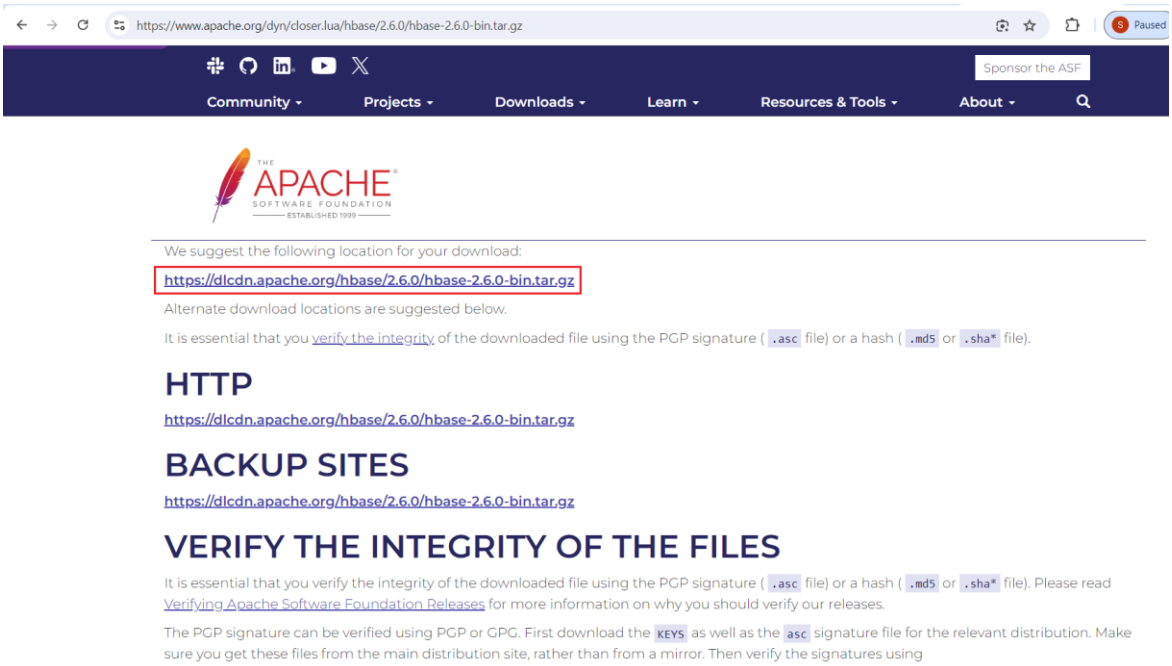
Let us see how HBase can be installed in standalone mode without Hadoop requirement.

3.1. Download HBase Binaries:

Download the stable version of HBase from the official Apache HBase Downloads website. At the time of this document preparation, the most recent stable release is 2.6.0. Select the latest version and click on **bin** link as shown below *(Check here for previous versions of HBase).*

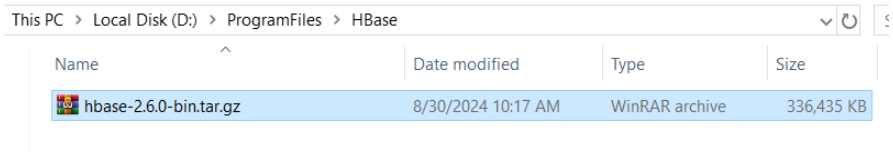


You will be navigated to Hbase 2.6.0 mirror website where click on the suggested location for hbase-2.6.0-bin.tar.gz file that gets downloaded to your **Downloads** folder in your machine.

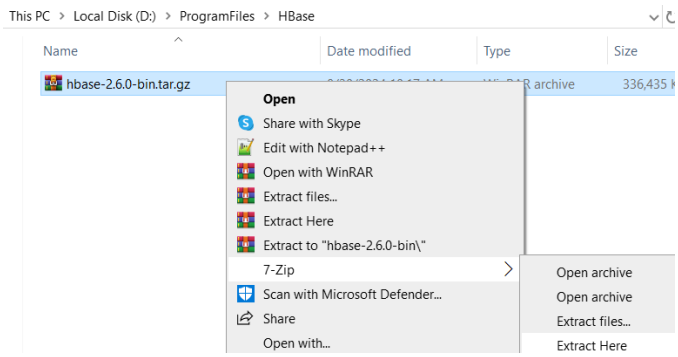


After the binary file is downloaded, unpack it using any file archiver (**7zip** or **WinRAR**) utility as below:

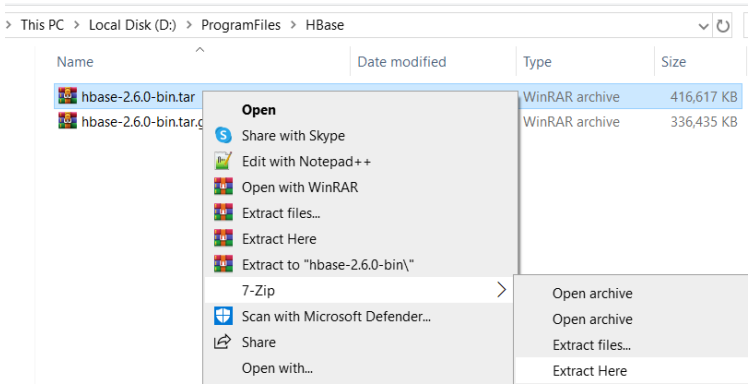
- Choose the installation directory in your machine and copy `hbase-2.6.0-bin.tar.gz` file to that directory. Here, we are choosing HBase installation directory as `D:\ProgramFiles\HBase`.



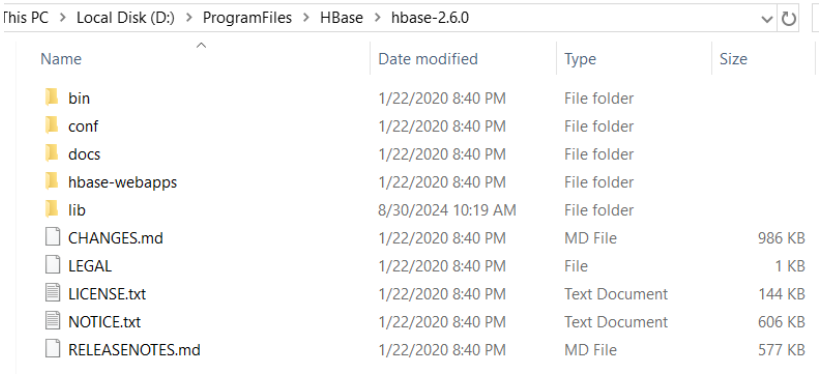
- Right click on `hbase-2.6.0-bin.tar.gz` and choose **7-Zip -> Extract Here** option which extracts a new packed file `hbase-2.6.0-bin.tar`.



- Next, unpack `hbase-2.6.0-bin.tar` file using **7zip** utility.



- The tar file extraction may take few minutes to finish. After finishing, you see a folder named `hbase-2.6.0` which consists of HBase binaries and libraries.



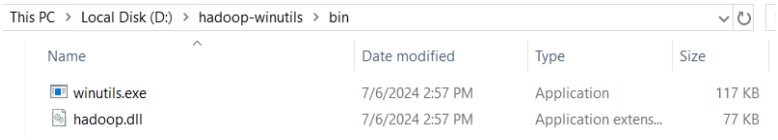
3.2. Download Hadoop Binaries:

Since HBase is not configured to run on Windows system by default, it is necessary to get Hadoop's native IO utilities for Windows from cdarlint GitHub repository.
Note that HBase 2.6.x version supports Hadoop 3.3.5 or more. Look at HBase Hadoop Support Matrix to know more about the supported versions of Hadoop for HBase.

- Since we installed the HBase 2.6.0 version, download utilities such as `winutils.exe` and `hadoop.dll` of the latest Hadoop 3.3.6 version from here.

- Create a folder named `hadoop-winutils` in **D drive** and create a sub-folder named `bin` inside it.

- Move `winutils.exe` and `hadoop.dll` files from your downloaded location to `D:\hadoop-winutils\bin` directory.



3.3. Set up Environment variables:

After installing pre-requisites and HBase binaries, we should configure the below environment variables defining Java and HBase default paths.

- **JAVA_HOME**: This is the JDK installation directory path in the machine *(in my machine, it is `D:\ProgramFiles\Java\jdk-1.8`)*. Ignore it if this is already done.

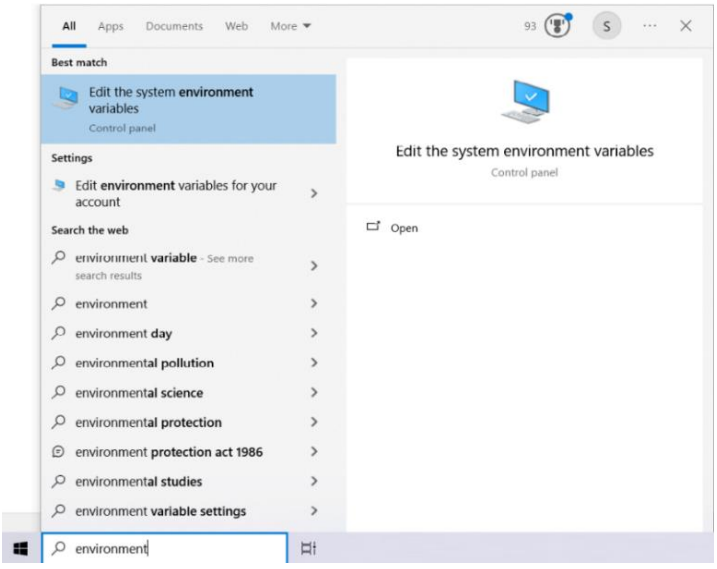
- **HADOOP_HOME**: This is the Hadoop's WinUtils path in the machine *(in our case, it is `D:\hadoop-winutils`)*

- **HBASE_HOME**: This is the HBase installation directory path in the machine *(in our case, it is `D:\ProgramFiles\HBase\hbase-2.6.0`)*

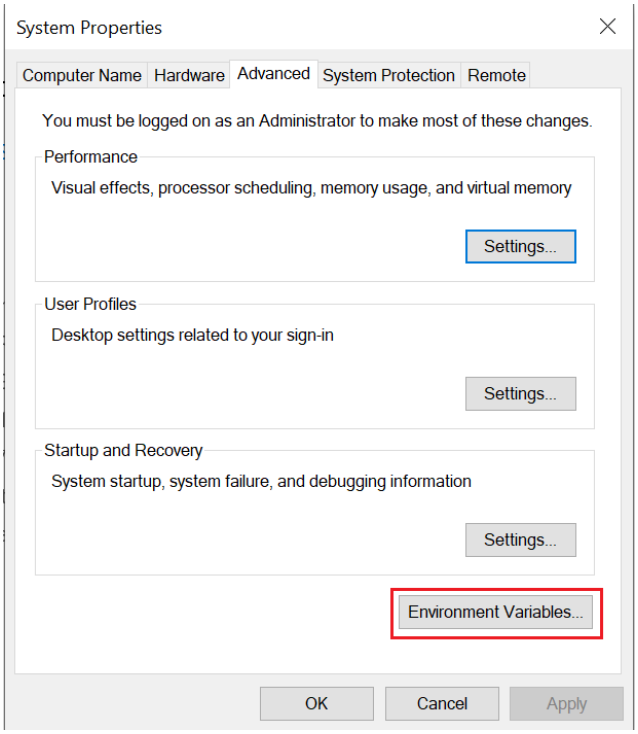
These variables need to be added to either **User environment variables** or **System environment variables** depending on HBase configuration needed **for a single user** or **for multiple users**.

In this tutorial, we will add User environment variables since we are configuring HBase for a single user. If you would like to configure HBase for multiple users, then define System environment variables.

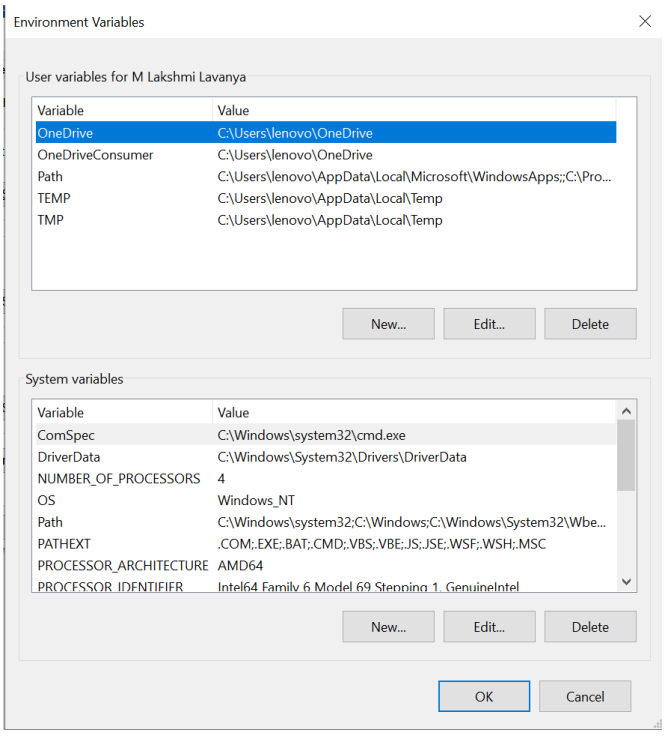
In the Windows search bar, start typing "environment variables" and select the first match which opens up **System Properties** dialog.



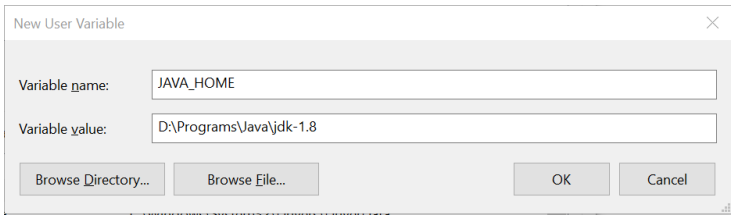
On the **System Properties** window, press **Environment Variables** button.



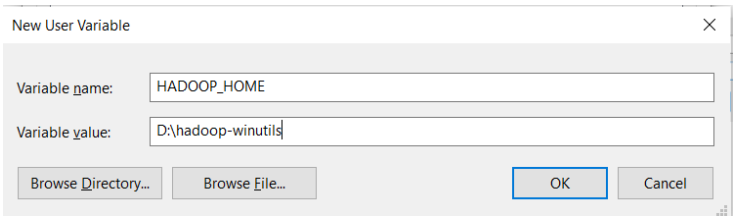
In the **Environment Variables** dialog, click on **New** under **User variables** section.



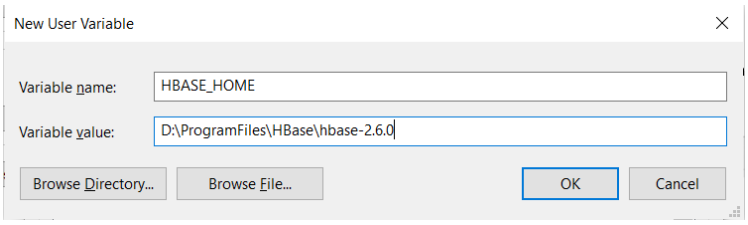
Add `JAVA_HOME` variable and press OK.



Click on **New** again and add `HADOOP_HOME` variable and press OK.

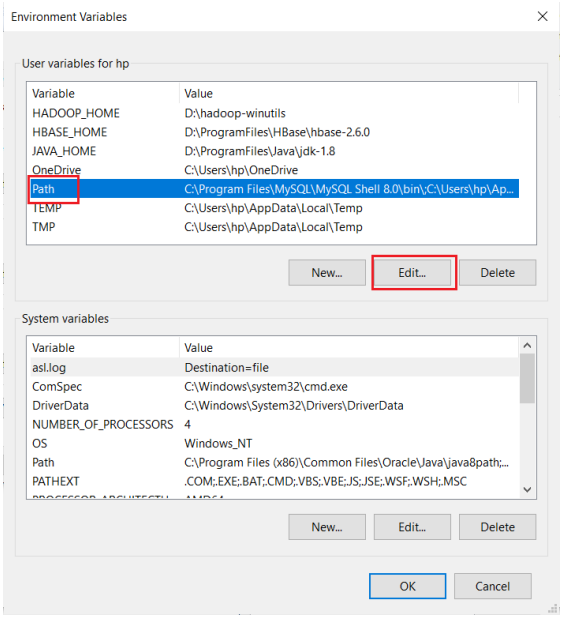


Click on **New** again and add `HBASE_HOME` variable and press OK.



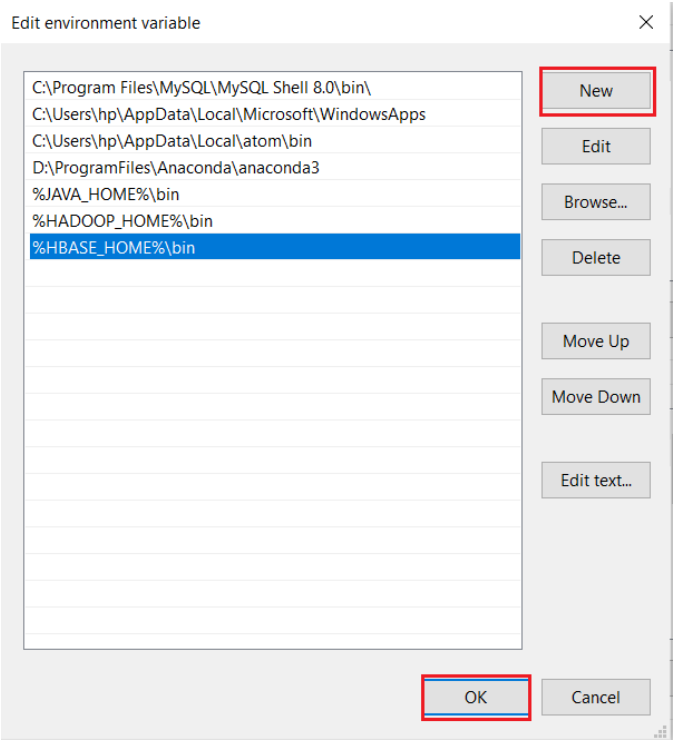
Now, we will update `PATH` variable to add Java, Hadoop and HBase binary paths.

Select `PATH` variable under **User Variables** and press **Edit** button.

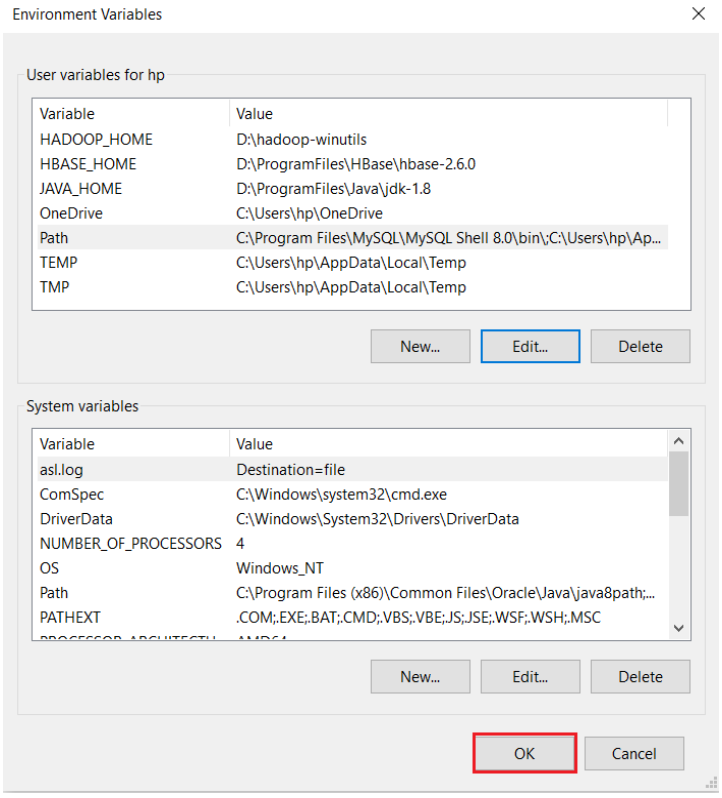


Press **New** and add the following values and press OK.

```
%JAVA_HOME%\bin
%HADOOP_HOME%\bin
%HBASE_HOME%\bin
```



Press OK again to apply environment variable changes and close window.



3.4. Configure HBase:

Next, we should modify the following files to configure the standalone HBase:

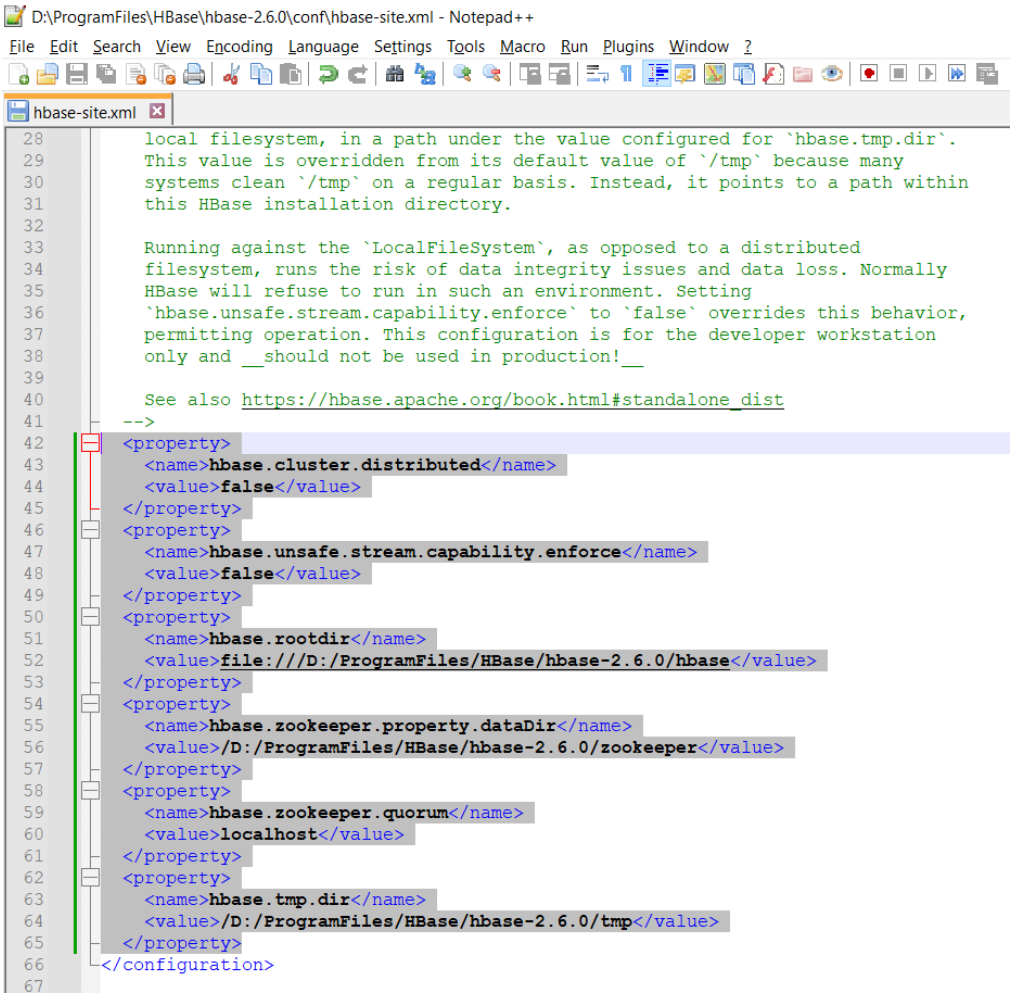
```
%HBASE_HOME%\conf\hbase-site.xml
%HBASE_HOME%\conf\hbase-env.cmd
```

3.4.1. HBase Site Configuration:

HBase needs a directory location to store the HBase and Zookeeper files which should be specified in `hbase-site.xml` file. If not specified, it automatically creates directories under `%TEMP%` directory.

Open `hbase-site.xml` file in `HBASE_HOME\conf` folder and replace the existing properties with the below lines inside the `<configuration>` tag.

```
<property>
  <name>hbase.cluster.distributed</name>
  <value>false</value>
</property>
<property>
  <name>hbase.unsafe.stream.capability.enforce</name>
  <value>false</value>
</property>
<property>
  <name>hbase.rootdir</name>
  <value>file:///D:/ProgramFiles/HBase/hbase-2.6.0/hbase</value>
</property>
<property>
  <name>hbase.zookeeper.property.dataDir</name>
  <value>/D:/ProgramFiles/HBase/hbase-2.6.0/zookeeper</value>
</property>
<property>
  <name>hbase.zookeeper.quorum</name>
  <value>localhost</value>
</property>
<property>
  <name>hbase.tmp.dir</name>
  <value>/D:/ProgramFiles/HBase/hbase-2.6.0/tmp</value>
</property>
```



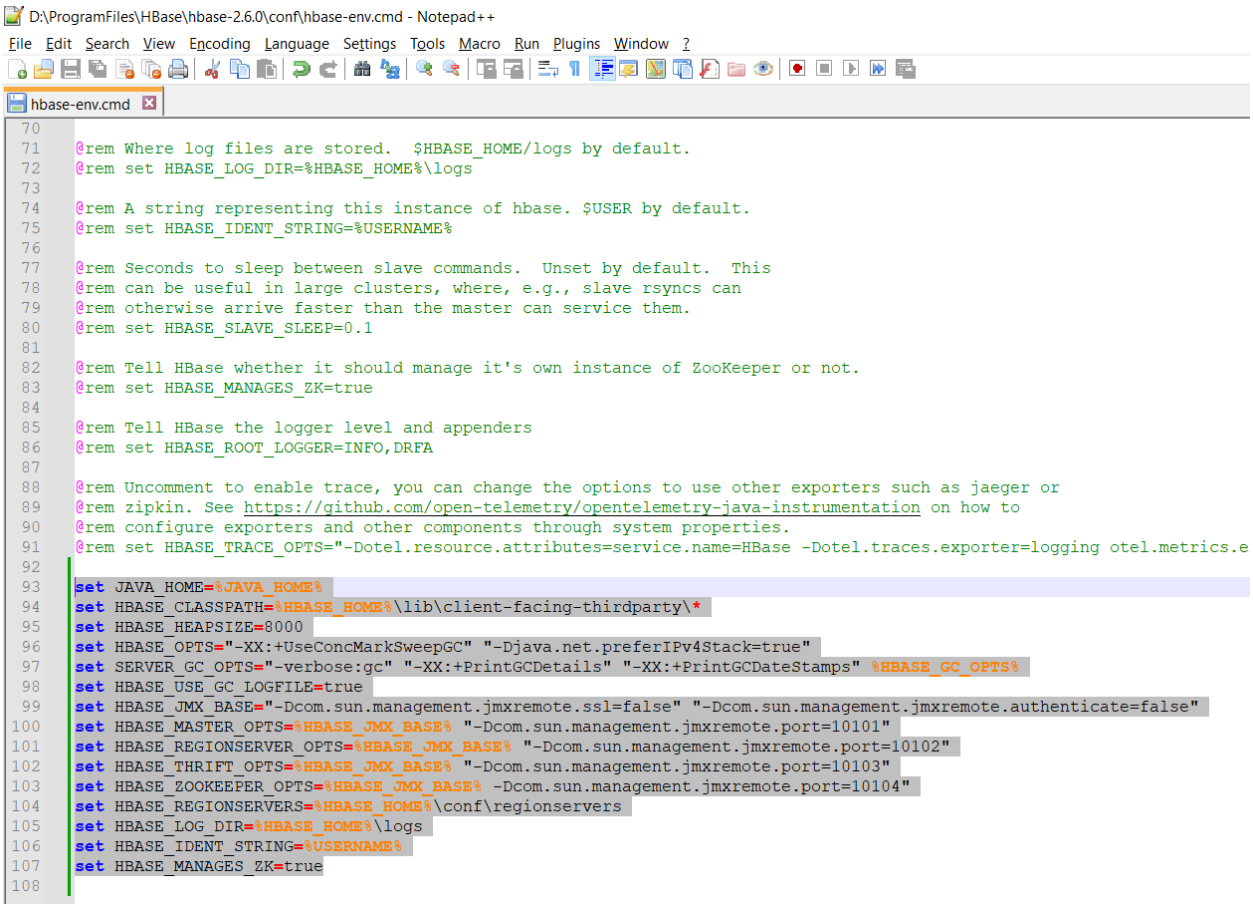
Note that HBase creates `hbase, zookeeper` and `logs` folders in `%HBASE_HOME%` location automatically when it is started.

3.4.2. HBase Env Configuration:

Open `hbase-env.cmd` file in `%HBASE_HOME%\conf` folder and add the following lines:

```
set JAVA_HOME=%JAVA_HOME%
set HBASE_CLASSPATH=%HBASE_HOME%\lib\client-facing-thirdparty\*
set HBASE_HEAPSIZE=8000
set HBASE_OPTS="-XX:+UseConcMarkSweepGC" "-Djava.net.preferIPv4Stack=true"
set SERVER_GC_OPTS="-verbose:gc" "-XX:+PrintGCDetails" "-XX:+PrintGCDateStamps"
%HBASE_GC_OPTS%
set HBASE_USE_GC_LOGFILE=true
set HBASE_JMX_BASE="-Dcom.sun.management.jmxremote.ssl=false" "-
Dcom.sun.management.jmxremote.authenticate=false"
set HBASE_MASTER_OPTS=%HBASE_JMX_BASE% "-
Dcom.sun.management.jmxremote.port=10101"
set HBASE_REGIONSERVER_OPTS=%HBASE_JMX_BASE% "-
Dcom.sun.management.jmxremote.port=10102"
```

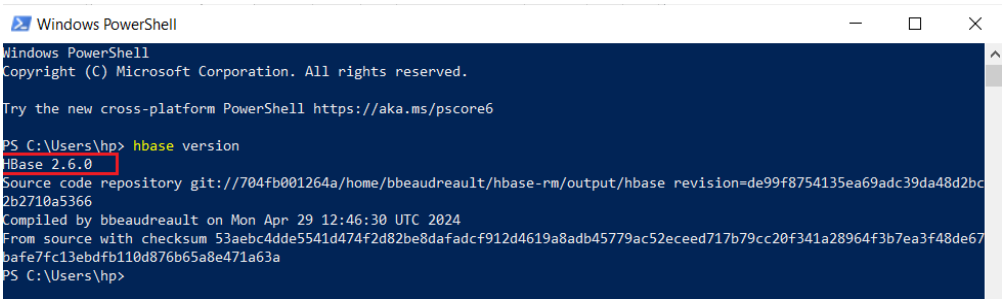
```
set HBASE_THRIFT_OPTS=%HBASE_JMX_BASE% "-
Dcom.sun.management.jmxremote.port=10103"
set HBASE_ZOOKEEPER_OPTS=%HBASE_JMX_BASE% -
Dcom.sun.management.jmxremote.port=10104"
set HBASE_REGIONSERVERS=%HBASE_HOME%\conf\regionservers
set HBASE_LOG_DIR=%HBASE_HOME%\logs
set HBASE_IDENT_STRING=%USERNAME%
set HBASE_MANAGES_ZK=true
```



3.5. Verify HBase Installation:

Open **Windows PowerShell** and run the following command to verify if HBase is installed properly:

```
hbase version
```

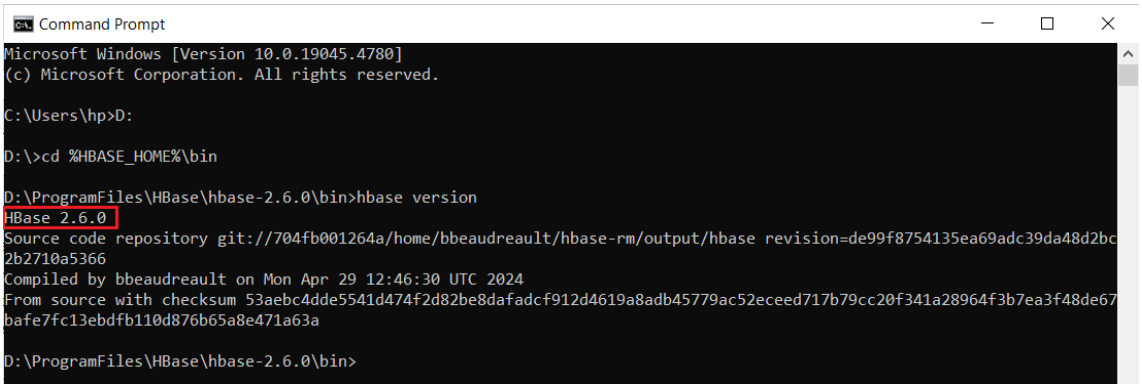


Here, it shows **HBase 2.6.0** which indicates that HBase has been installed successfully.

Note:

If you would like to check the hbase version using **Command Prompt**, then you must first navigate to the location where HBase is installed as below.

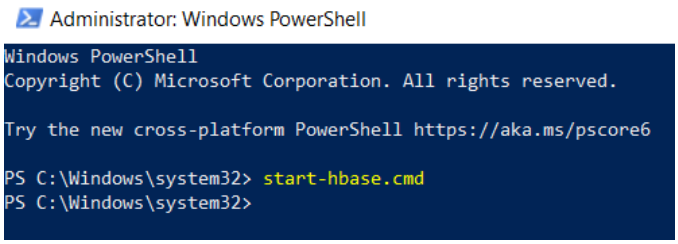
```
D:
cd %HBASE_HOME%\bin
hbase version
```



3.6. Start HBase:

Open **Windows PowerShell** as **Administrator** and start HBase using the below command

```
start-hbase.cmd
```

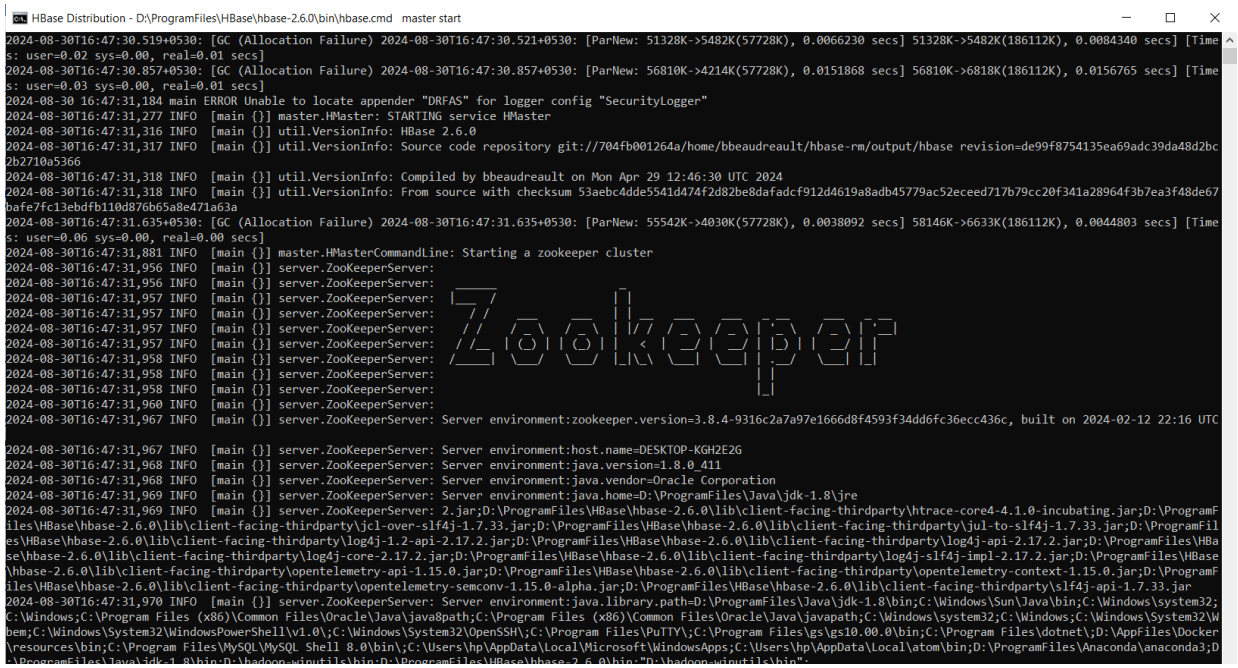


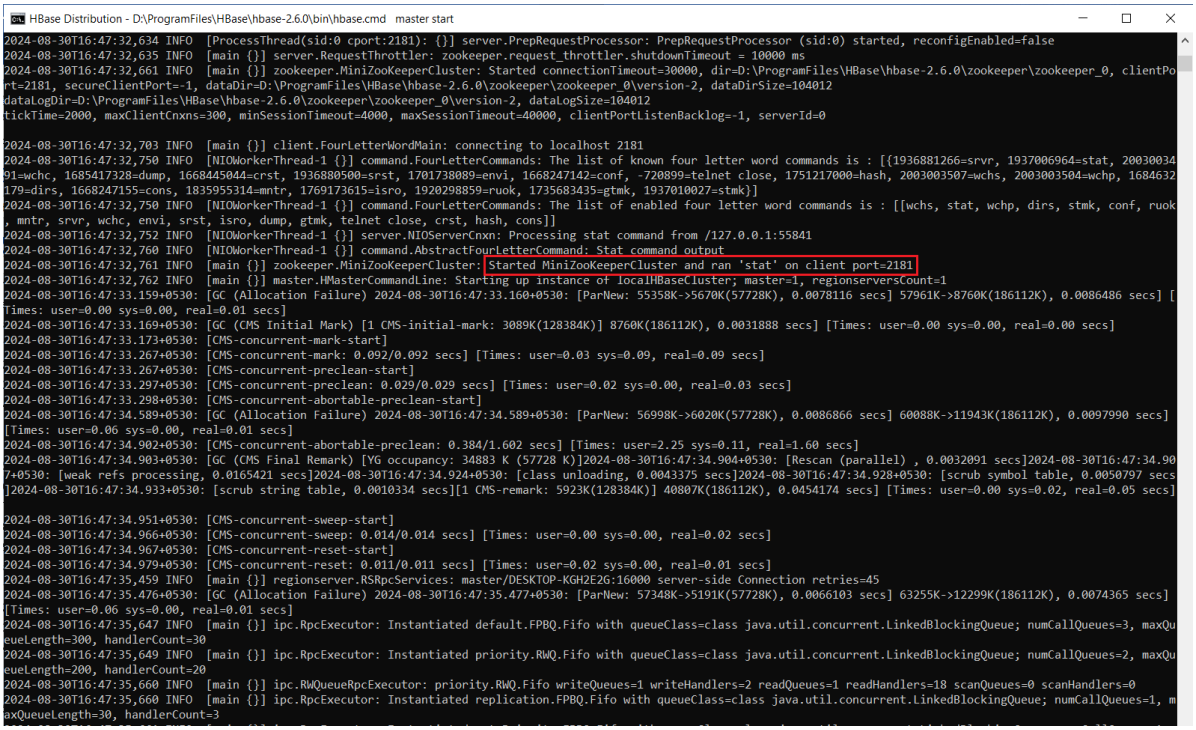
Note:

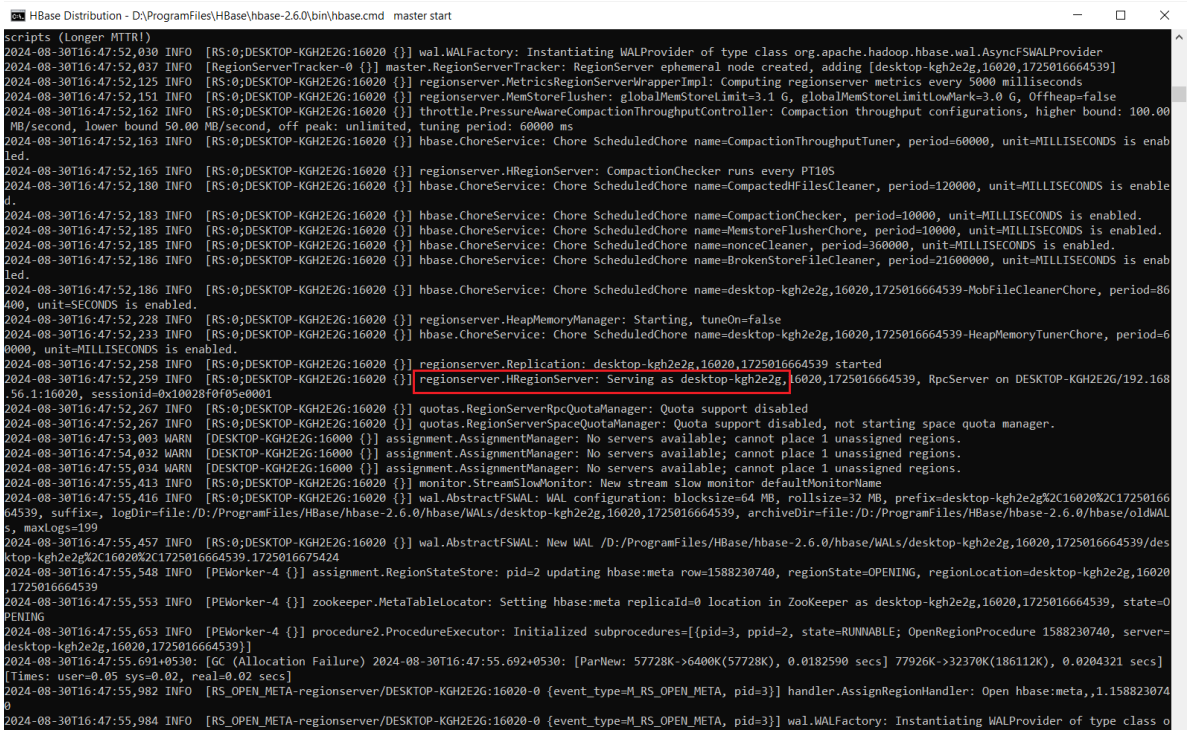
If you would like to start hbase using **Command Prompt**, then you must first navigate to the location where HBase is installed as below.

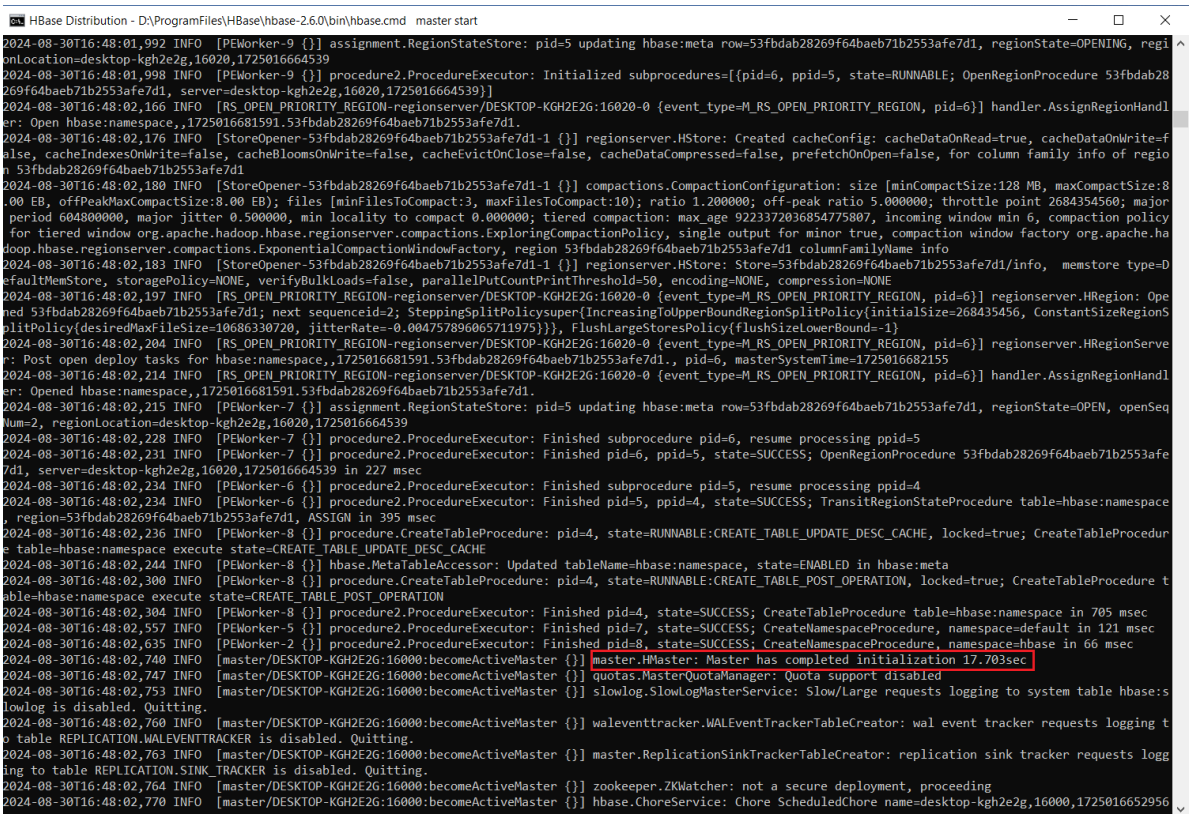
```
D:
cd %HBASE_HOME%\bin
start-hbase.cmd
```

It opens up a new command prompt where we can see three daemon services `Zookeeper`, `HMaster` and `HRegion` server are started. Wait until the HMaster initialization is complete.




HBase Distribution - D:\ProgramFiles\HBase\hbase-2.6.0\bin\hbase.cmd master start
2024-08-30T16:47:32,634 INFO [ProcessThread(sid:0 cport:2181): {}] server.PrepRequestProcessor: PrepRequestProcessor (sid:0) started, reconfigEnabled=false
2024-08-30T16:47:32,635 INFO [main {}] server.RequestThrottler: zookeeper.request_throttler.shutdownTimeout = 10000 ms
2024-08-30T16:47:32,661 INFO [main {}] zookeeper.MiniZooKeeperCluster: Started connectionTimeout=30000, dir=D:\ProgramFiles\HBase\hbase-2.6.0\zookeeper\zookeeper_0, clientPort=2181, secureClientPort=-1, dataDir=D:\ProgramFiles\HBase\hbase-2.6.0\zookeeper\zookeeper_0\version-2, dataDirSize=104012
dataLogDir=D:\ProgramFiles\HBase\hbase-2.6.0\zookeeper\zookeeper_0\version-2, dataLogSize=104012
tickTime=2000, maxClientCnxns=300, minSessionTimeout=4000, maxSessionTimeout=40000, clientPortListenBacklog=-1, serverId=0
2024-08-30T16:47:32,703 INFO [main {}] client.FourLetterWordMain: connecting to localhost 2181
2024-08-30T16:47:32,750 INFO [NIOWorkerThread-1 {}] command.FourLetterCommands: The list of known four letter word commands is : [{1936881266=srvr, 1937006964=stat, 20030034
91=wchc, 1685417328=dump, 1668445044=crst, 1936880500=srst, 1701738089=envi, 1668247142=conf, -720899=telnet close, 1751217000=hash, 2003003507=wchs, 2003003504=wchp, 1684632
179=dirs, 1668247155=cons, 1835955314=mntr, 1769173615=isro, 1920298859=ruok, 1735683435=gtmk, 1937010027=stmk}]
2024-08-30T16:47:32,750 INFO [NIOWorkerThread-1 {}] command.FourLetterCommands: The list of enabled four letter word commands is : [[wchs, stat, wchp, dirs, stmk, conf, ruok
, mntr, srvr, wchc, envi, srst, isro, dump, gtmk, telnet close, crst, hash, cons]]
2024-08-30T16:47:32,752 INFO [NIOWorkerThread-1 {}] server.NIOServerCnxn: Processing stat command from /127.0.0.1:55841
2024-08-30T16:47:32,760 INFO [NIOWorkerThread-1 {}] command.AbstractFourLetterCommand: Stat command output
2024-08-30T16:47:32,761 INFO [main {}] zookeeper.MiniZooKeeperCluster: Started MiniZooKeeperCluster and ran 'stat' on client port=2181
2024-08-30T16:47:32,762 INFO [main {}] master.HMasterCommandLine: Starting up instance of localHBaseCluster; master=1, regionserversCount=1
2024-08-30T16:47:33.159+0530: [GC (Allocation Failure) 2024-08-30T16:47:33.160+0530: [ParNew: 55358K->5670K(57728K), 0.0078116 secs] 57961K->8760K(186112K), 0.0086486 secs] [
Times: user=0.00 sys=0.00, real=0.01 secs]
2024-08-30T16:47:33.169+0530: [GC (CMS Initial Mark) [1 CMS-initial-mark: 3089K(128384K)] 8760K(186112K), 0.0031888 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]
2024-08-30T16:47:33.173+0530: [CMS-concurrent-mark-start]
2024-08-30T16:47:33.267+0530: [CMS-concurrent-mark: 0.092/0.092 secs] [Times: user=0.03 sys=0.09, real=0.09 secs]
2024-08-30T16:47:33.267+0530: [CMS-concurrent-preclean-start]
2024-08-30T16:47:33.297+0530: [CMS-concurrent-preclean: 0.029/0.029 secs] [Times: user=0.02 sys=0.00, real=0.03 secs]
2024-08-30T16:47:33.298+0530: [CMS-concurrent-abortable-preclean-start]
2024-08-30T16:47:34.589+0530: [GC (Allocation Failure) 2024-08-30T16:47:34.589+0530: [ParNew: 56998K->6020K(57728K), 0.0086866 secs] 60088K->11943K(186112K), 0.0097990 secs]
[Times: user=0.06 sys=0.00, real=0.01 secs]
2024-08-30T16:47:34.902+0530: [CMS-concurrent-abortable-preclean: 0.384/1.602 secs] [Times: user=2.25 sys=0.11, real=1.60 secs]
2024-08-30T16:47:34.903+0530: [GC (CMS Final Remark) [YG occupancy: 34883 K (57728 K)]2024-08-30T16:47:34.904+0530: [Rescan (parallel) , 0.0032091 secs]2024-08-30T16:47:34.90
7+0530: [weak refs processing, 0.0165421 secs]2024-08-30T16:47:34.924+0530: [class unloading, 0.0043375 secs]2024-08-30T16:47:34.928+0530: [scrub symbol table, 0.0050797 secs
]2024-08-30T16:47:34.933+0530: [scrub string table, 0.0010334 secs][1 CMS-remark: 5923K(128384K)] 40807K(186112K), 0.0454174 secs] [Times: user=0.00 sys=0.02, real=0.05 secs]
2024-08-30T16:47:34.951+0530: [CMS-concurrent-sweep-start]
2024-08-30T16:47:34.966+0530: [CMS-concurrent-sweep: 0.014/0.014 secs] [Times: user=0.00 sys=0.00, real=0.02 secs]
2024-08-30T16:47:34.967+0530: [CMS-concurrent-reset-start]
2024-08-30T16:47:34.979+0530: [CMS-concurrent-reset: 0.011/0.011 secs] [Times: user=0.02 sys=0.00, real=0.01 secs]
2024-08-30T16:47:35,459 INFO [main {}] regionserver.RSRpcServices: master/DESKTOP-KGH2E2G:16000 server-side Connection retries=45
2024-08-30T16:47:35.476+0530: [GC (Allocation Failure) 2024-08-30T16:47:35.477+0530: [ParNew: 57348K->5191K(57728K), 0.0066103 secs] 63255K->12299K(186112K), 0.0074365 secs]
[Times: user=0.06 sys=0.00, real=0.01 secs]
2024-08-30T16:47:35,647 INFO [main {}] ipc.RpcExecutor: Instantiated default.FPBQ.Fifo with queueClass=class java.util.concurrent.LinkedBlockingQueue; numCallQueues=3, maxQueueLength=300, handlerCount=30
2024-08-30T16:47:35,649 INFO [main {}] ipc.RpcExecutor: Instantiated priority.RWQ.Fifo with queueClass=class java.util.concurrent.LinkedBlockingQueue; numCallQueues=2, maxQueueLength=200, handlerCount=20
2024-08-30T16:47:35,660 INFO [main {}] ipc.RWQueueRpcExecutor: priority.RWQ.Fifo writeQueues=1 writeHandlers=2 readQueues=1 readHandlers=18 scanQueues=0 scanHandlers=0
2024-08-30T16:47:35,660 INFO [main {}] ipc.RpcExecutor: Instantiated replication.FPBQ.Fifo with queueClass=class java.util.concurrent.LinkedBlockingQueue; numCallQueues=1, maxQueueLength=30, handlerCount=3

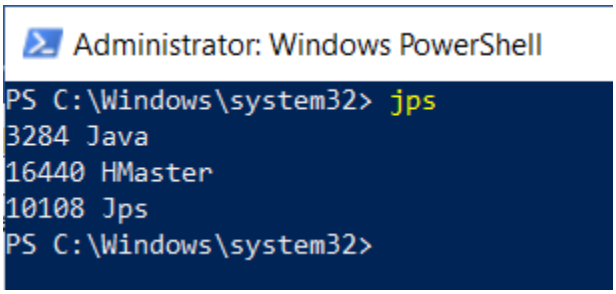

HBase Distribution - D:\ProgramFiles\HBase\hbase-2.6.0\bin\hbase.cmd master start
scripts (Longer MTTR!)
2024-08-30T16:47:52,030 INFO [RS:0;DESKTOP-KGH2E2G:16020 {}] wal.WALFactory: Instantiating WALProvider of type class org.apache.hadoop.hbase.wal.AsyncFSWALProvider
2024-08-30T16:47:52,037 INFO [RegionServerTracker-0 {}] master.RegionServerTracker: RegionServer ephemeral node created, adding [desktop-kgh2e2g,16020,1725016664539]
2024-08-30T16:47:52,125 INFO [RS:0;DESKTOP-KGH2E2G:16020 {}] regionserver.MetricsRegionServerWrapperImpl: Computing regionserver metrics every 5000 milliseconds
2024-08-30T16:47:52,151 INFO [RS:0;DESKTOP-KGH2E2G:16020 {}] regionserver.MemStoreFlusher: globalMemStoreLimit=3.1 G, globalMemStoreLimitLowMark=3.0 G, Offheap=false
2024-08-30T16:47:52,162 INFO [RS:0;DESKTOP-KGH2E2G:16020 {}] throttle.PressureAwareCompactionThroughputController: Compaction throughput configurations, higher bound: 100.00
MB/second, lower bound 50.00 MB/second, off peak: unlimited, tuning period: 60000 ms
2024-08-30T16:47:52,163 INFO [RS:0;DESKTOP-KGH2E2G:16020 {}] hbase.ChoreService: Chore ScheduledChore name=CompactionThroughputTuner, period=60000, unit=MILLISECONDS is enabled.
2024-08-30T16:47:52,165 INFO [RS:0;DESKTOP-KGH2E2G:16020 {}] regionserver.HRegionServer: CompactionChecker runs every PT10S
2024-08-30T16:47:52,180 INFO [RS:0;DESKTOP-KGH2E2G:16020 {}] hbase.ChoreService: Chore ScheduledChore name=CompactedHFilesCleaner, period=120000, unit=MILLISECONDS is enabled.
2024-08-30T16:47:52,183 INFO [RS:0;DESKTOP-KGH2E2G:16020 {}] hbase.ChoreService: Chore ScheduledChore name=CompactionChecker, period=10000, unit=MILLISECONDS is enabled.
2024-08-30T16:47:52,185 INFO [RS:0;DESKTOP-KGH2E2G:16020 {}] hbase.ChoreService: Chore ScheduledChore name=MemstoreFlusherChore, period=10000, unit=MILLISECONDS is enabled.
2024-08-30T16:47:52,185 INFO [RS:0;DESKTOP-KGH2E2G:16020 {}] hbase.ChoreService: Chore ScheduledChore name=nonceCleaner, period=360000, unit=MILLISECONDS is enabled.
2024-08-30T16:47:52,186 INFO [RS:0;DESKTOP-KGH2E2G:16020 {}] hbase.ChoreService: Chore ScheduledChore name=BrokenStoreFileCleaner, period=21600000, unit=MILLISECONDS is enabled.
2024-08-30T16:47:52,186 INFO [RS:0;DESKTOP-KGH2E2G:16020 {}] hbase.ChoreService: Chore ScheduledChore name=desktop-kgh2e2g,16020,1725016664539-MobFileCleanerChore, period=86400, unit=SECONDS is enabled.
2024-08-30T16:47:52,228 INFO [RS:0;DESKTOP-KGH2E2G:16020 {}] regionserver.HeapMemoryManager: Starting, tuneOn=false
2024-08-30T16:47:52,233 INFO [RS:0;DESKTOP-KGH2E2G:16020 {}] hbase.ChoreService: Chore ScheduledChore name=desktop-kgh2e2g,16020,1725016664539-HeapMemoryTunerChore, period=60000, unit=MILLISECONDS is enabled.
2024-08-30T16:47:52,258 INFO [RS:0;DESKTOP-KGH2E2G:16020 {}] regionserver.Replication: desktop-kgh2e2g,16020,1725016664539 started
2024-08-30T16:47:52,259 INFO [RS:0;DESKTOP-KGH2E2G:16020 {}] regionserver.HRegionServer: Serving as desktop-kgh2e2g,16020,1725016664539, RpcServer on DESKTOP-KGH2E2G/192.168.56.1:16020, sessionid=0x10028f0f05e0001
2024-08-30T16:47:52,267 INFO [RS:0;DESKTOP-KGH2E2G:16020 {}] quotas.RegionServerRpcQuotaManager: Quota support disabled
2024-08-30T16:47:52,267 INFO [RS:0;DESKTOP-KGH2E2G:16020 {}] quotas.RegionServerSpaceQuotaManager: Quota support disabled, not starting space quota manager.
2024-08-30T16:47:53,003 WARN [DESKTOP-KGH2E2G:16000 {}] assignment.AssignmentManager: No servers available; cannot place 1 unassigned regions.
2024-08-30T16:47:54,032 WARN [DESKTOP-KGH2E2G:16000 {}] assignment.AssignmentManager: No servers available; cannot place 1 unassigned regions.
2024-08-30T16:47:55,034 WARN [DESKTOP-KGH2E2G:16000 {}] assignment.AssignmentManager: No servers available; cannot place 1 unassigned regions.
2024-08-30T16:47:55,413 INFO [RS:0;DESKTOP-KGH2E2G:16020 {}] monitor.StreamSlowMonitor: New stream slow monitor defaultMonitorName
2024-08-30T16:47:55,416 INFO [RS:0;DESKTOP-KGH2E2G:16020 {}] wal.AbstractFSWAL: WAL configuration: blocksize=64 MB, rollsize=32 MB, prefix=desktop-kgh2e2g%2C16020%2C1725016664539, suffix=, logDir=file:/D:/ProgramFiles/HBase/hbase-2.6.0/hbase/WALs/desktop-kgh2e2g,16020,1725016664539, archiveDir=file:/D:/ProgramFiles/HBase/hbase-2.6.0/hbase/oldWALs, maxLogs=199
2024-08-30T16:47:55,457 INFO [RS:0;DESKTOP-KGH2E2G:16020 {}] wal.AbstractFSWAL: New WAL /D:/ProgramFiles/HBase/hbase-2.6.0/hbase/WALs/desktop-kgh2e2g,16020,1725016664539/desktop-kgh2e2g%2C16020%2C1725016664539.1725016675424
2024-08-30T16:47:55,548 INFO [PEWorker-4 {}] assignment.RegionStateStore: pid=2 updating hbase:meta row=1588230740, regionState=OPENING, regionLocation=desktop-kgh2e2g,16020,1725016664539
2024-08-30T16:47:55,553 INFO [PEWorker-4 {}] zookeeper.MetaTableLocator: Setting hbase:meta replicaId=0 location in ZooKeeper as desktop-kgh2e2g,16020,1725016664539, state=OPENING
2024-08-30T16:47:55,653 INFO [PEWorker-4 {}] procedure2.ProcedureExecutor: Initialized subprocedures=[{pid=3, ppid=2, state=RUNNABLE; OpenRegionProcedure 1588230740, server=desktop-kgh2e2g,16020,1725016664539}]
2024-08-30T16:47:55.691+0530: [GC (Allocation Failure) 2024-08-30T16:47:55.692+0530: [ParNew: 57728K->6400K(57728K), 0.0182590 secs] 77926K->32370K(186112K), 0.0204321 secs]
[Times: user=0.05 sys=0.02, real=0.02 secs]
2024-08-30T16:47:55,982 INFO [RS_OPEN_META-regionserver/DESKTOP-KGH2E2G:16020-0 {event_type=M_RS_OPEN_META, pid=3}] handler.AssignRegionHandler: Open hbase:meta,,1.1588230740
2024-08-30T16:47:55,984 INFO [RS_OPEN_META-regionserver/DESKTOP-KGH2E2G:16020-0 {event_type=M_RS_OPEN_META, pid=3}] wal.WALFactory: Instantiating WALProvider of type class o



To verify the HBase has been started successfully, use the `jps` command as below

```
jps
```

It should display **HMaster** process which denotes **HMaster**, single **HRegionServer** and **Zookeeper** services are running in a single JVM.



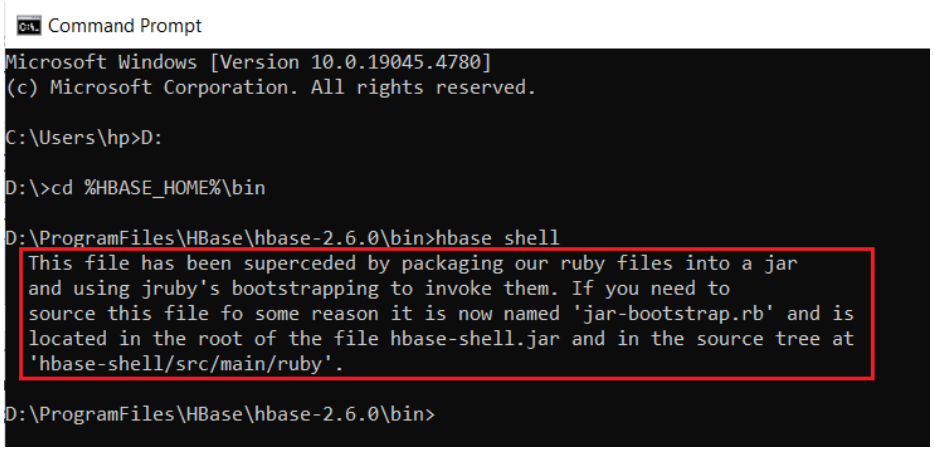
3.7. HBase Shell:

Open **Windows PowerShell** and run the below command to launch HBase shell.

```
hbase shell
```

If you would like to launch hbase from **Command Prompt**, then you must first navigate to the location where HBase is installed as below.

```
D:
cd %HBASE_HOME%\bin
hbase shell
```



While launching the hbase shell, you may encounter error *"This file has been superceded by packaging our ruby files into a jar and using jruby's bootstrapping to invoke them."*

To resolve this error, follow the below steps:

- Edit `hbase.cmd` file in `%HBASE_HOME%\bin` location and replace the following lines of code under `:shell` label.

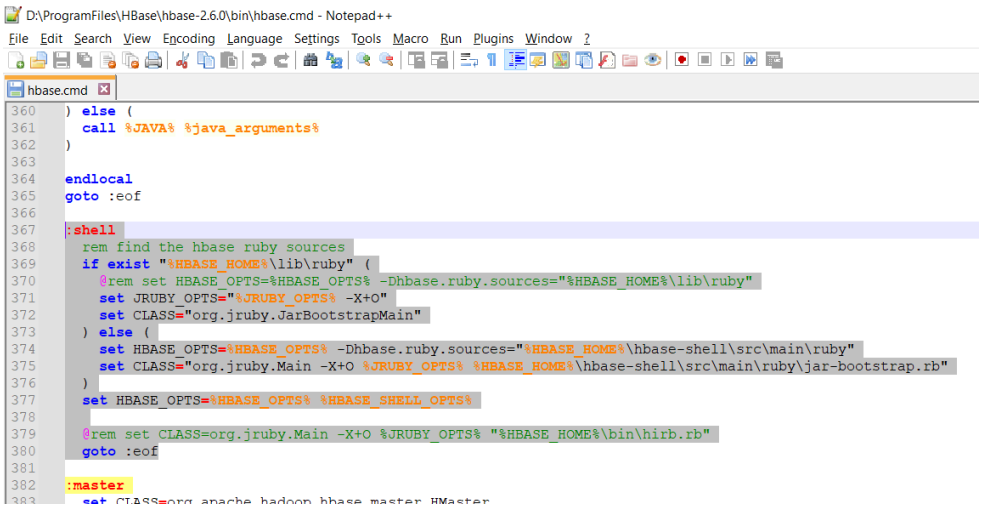
Before:

```
:shell
  rem find the hbase ruby sources
  if exist "%HBASE_HOME%\lib\ruby" (
    set HBASE_OPTS=%HBASE_OPTS% -Dhbase.ruby.sources="%HBASE_HOME%\lib\ruby"
  ) else (
    set HBASE_OPTS=%HBASE_OPTS% -Dhbase.ruby.sources="%HBASE_HOME%\hbase-
shell\src\main\ruby"
  )
  set HBASE_OPTS=%HBASE_OPTS% %HBASE_SHELL_OPTS%

  set CLASS=org.jruby.Main -X+O %JRUBY_OPTS% "%HBASE_HOME%\bin\hirb.rb"
  goto :eof
```

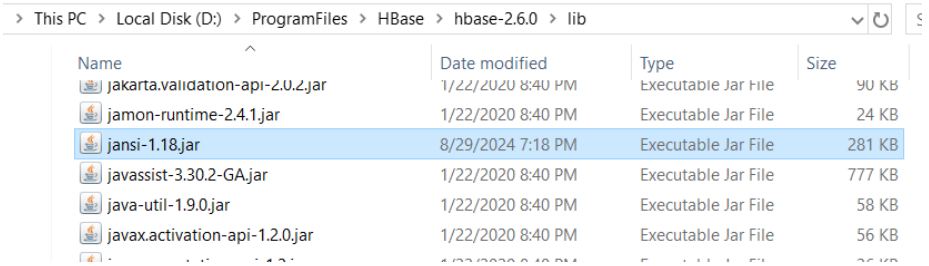
After:

```
:shell
  rem find the hbase ruby sources
  if exist "%HBASE_HOME%\lib\ruby" (
    @rem set HBASE_OPTS=%HBASE_OPTS% -
Dhbase.ruby.sources="%HBASE_HOME%\lib\ruby"
  set JRUBY_OPTS="${JRUBY_OPTS} -X+O"
```

```
  set CLASS="org.jruby.JarBootstrapMain"
  ) else (
    set HBASE_OPTS=%HBASE_OPTS% -Dhbase.ruby.sources="%HBASE_HOME%\hbase-
shell\src\main\ruby"
  set CLASS="org.jruby.Main -X+O %JRUBY_OPTS% %HBASE_HOME%/hbase-
shell/src/main/ruby/jar-bootstrap.rb"
  )
  set HBASE_OPTS=%HBASE_OPTS% %HBASE_SHELL_OPTS%

  @rem set CLASS=org.jruby.Main -X+O %JRUBY_OPTS% "%HBASE_HOME%\bin\hirb.rb"
  goto :eof
```

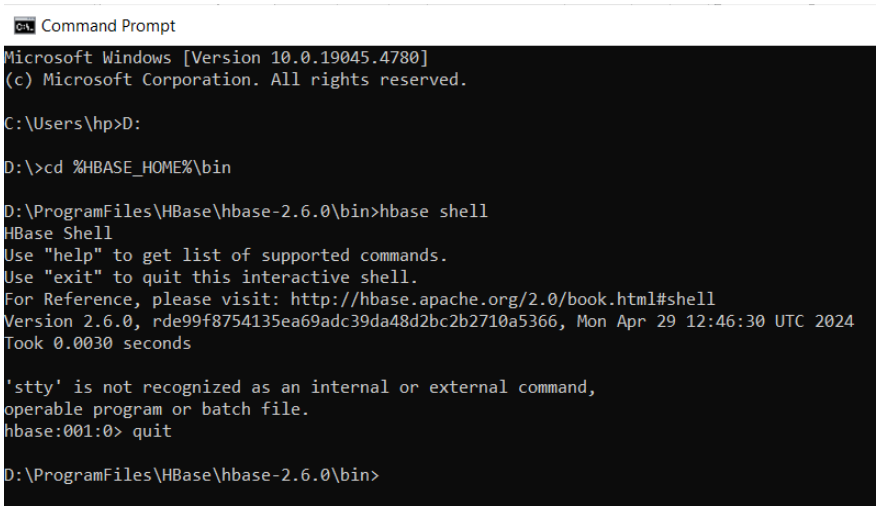


- Then download `jansi-1.18.jar` file from [this Maven repository] and place the file under `%HBASE_HOME%\lib` location.



- Now, you should be able to launch the hbase shell

```
hbase shell
```



Use `quit` or `exit` to come out of HBase interactive shell.

Note:

HBase 2.x version does not support installing Pseudo Distributed mode of HBase in Windows Operating System.

4. HBase Shell Commands:

HBase provides an interactive tool called “hbase shell” where you can execute commands to interact with HBase server.

To interact with the database, HBase Shell commands are divided into 14 groups:

1. General Commands
2. Namespace Commands
3. Data Definition Commands
4. Data Manipulation Commands
5. Tool Commands
6. Cluster Replication Commands
7. Snapshot Commands
8. Configuration Commands
9. Quota Commands
10. Security Commands
11. Procedure Commands
12. Visibility Label Commands
13. RS Group Commands
14. Store File Tracker Commands

Once you entered into HBase shell, just run `help` to get the list of all shell commands.

```
hbase:001:0> help
HBase Shell, version 2.6.0, rde99f8754135ea69adc39da48d2bc2b2710a5366, Mon Apr 29 12:46:30 UTC 2024
Type 'help "COMMAND"', (e.g. 'help "get"' -- the quotes are necessary) for help on a specific command.
Commands are grouped. Type 'help "COMMAND_GROUP"', (e.g. 'help "general"') for help on a command group.

COMMAND GROUPS:
  Group name: general
  Commands: processlist, status, table_help, version, whoami

  Group name: ddl
  Commands: alter, alter_async, alter_status, clone_table_schema, create, describe, disable, disable_all, drop, drop_all
, enable, enable_all, exists, get_table, is_disabled, is_enabled, list, list_disabled_tables, list_enabled_tables, list_
regions, locate_region, show_filters

  Group name: namespace
  Commands: alter_namespace, create_namespace, describe_namespace, drop_namespace, list_namespace, list_namespace_tables


  Group name: dml
  Commands: append, count, delete, deleteall, get, get_counter, get_splits, incr, put, scan, truncate, truncate_preserve


  Group name: tools
  Commands: assign, balance_switch, balancer, balancer_enabled, catalogjanitor_enabled, catalogjanitor_run, catalogjanit
or_switch, cleaner_chore_enabled, cleaner_chore_run, cleaner_chore_switch, clear_block_cache, clear_compaction_queues, c
lear_deadservers, clear_slowlog_responses, close_region, compact, compact_rs, compaction_state, compaction_switch, decom
mission_regionservers, flush, flush_master_store, get_balancer_decisions, get_balancer_rejections, get_largelog_response
s, get_slowlog_responses, hbck_chore_run, is_in_maintenance_mode, list_deadservers, list_decommissioned_regionservers, l
ist_liveservers, list_unknownservers, major_compact, merge_region, move, normalize, normalizer_enabled, normalizer_switc
h, recommission_regionserver, regioninfo, rit, snapshot_cleanup_enabled, snapshot_cleanup_switch, split, splitormerge_en
abled, splitormerge_switch, stop_master, stop_regionserver, trace, truncate_region, unassign, wal_roll, zk_dump

  Group name: replication
  Commands: add_peer, append_peer_exclude_namespaces, append_peer_exclude_tableCFs, append_peer_namespaces, append_peer_
tableCFs, disable_peer, disable_table_replication, enable_peer, enable_table_replication, get_peer_config, list_peer_con
figs, list_peers, list_replicated_tables, peer_modification_enabled, peer_modification_switch, remove_peer, remove_peer_
exclude_namespaces, remove_peer_exclude_tableCFs, remove_peer_namespaces, remove_peer_tableCFs, set_peer_bandwidth, set_
peer_exclude_namespaces, set_peer_exclude_tableCFs, set_peer_namespaces, set_peer_replicate_all, set_peer_serial, set_pe
er_tableCFs, show_peer_tableCFs, update_peer_config
```

To know the usage of each shell command, run `help '<command>'` *(i.e after `help`, give a space and enter the command name in single quotes)*.

For example, run the following to get more help of `status` command

```
help 'status'
```

```
hbase:004:0> help 'status'
Show cluster status. Can be 'summary', 'simple', 'detailed', 'tasks', or 'replication'. The
default is 'summary'. Examples:

  hbase> status
  hbase> status 'summary'
  hbase> status 'simple'
  hbase> status 'detailed'
  hbase> status 'tasks'
  hbase> status 'replication'
  hbase> status 'replication', 'source'
  hbase> status 'replication', 'sink'
hbase:005:0>
```

Some of the HBase general commands include:

- `status`: It provides the HBase system status such as number of servers present in the cluster, active servers count and average load value.
- `version`: It displays the current HBase version value.
- `table_help`: It provides what and how to use table-reference commands such as `create, put, get, scan`, etc.
- `whoami`: It provides information such as user information and groups present in the HBase.
- `processlist`: It provides the total number of tasks executed on the HBase cluster, active running tasks etc.

Some of the HBase namespace commands include:

- `create_namespace`: This command is used to create a namespace in HBase database.
- `list_namespace`: It displays all namespaces present in HBase.
- `describe_namespace`: It provides more information of the given namespace.
- `alter_namespace`: It alters the namespace properties such as adding a new property, modifying the existing property and deleting the existing property.
- `drop_namespace`: It drops the given namespace from HBase.

Some of the HBase table management (Data Definition) commands include:

- `create`: This command is used to create a table in HBase.
- `list`: It displays all tables present in HBase.
- `describe`: It provides more information about column families, associated filters, versions, etc. of the given table.
- `disable`: It disables the given table. It is necessary to disable the table before deleting or dropping it.
- `drop`: It drops the given table from HBase. Make sure that the table is already disabled before dropping it.
- `enable`: It starts enabling the given table in HBase. This command retrieves the disabled table to its previous state.
- `alter`: It alters the column family schema such as altering single or multiple column family names, deleting column families, etc.
- `show_filters`: It displays all filters such as `RowFilter`, `ColumnPrefixFilter`, `TimestampsFilter`, `PageFilter`, `FamilyFilter`, etc that are present in HBase.

Some of the HBase data manipulation commands include:

- `count`: It retrieves the count of rows in a HBase table.

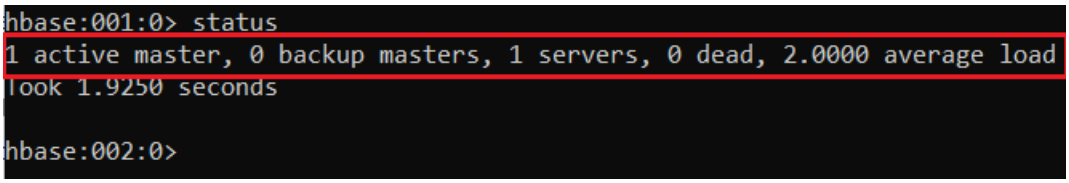
- `put`: It inserts a cell value at the defined table or row or column and / or timestamp coordinates.
- `Append`: It appends a cell value at specified table or row or column coordinates.
- `get`: It retrieves a row or cell contents present in a HBase table. We can also add additional parameters such as `TIMERANGE`, `TIMESTAMP`, `VERSIONS` and `FILTERS`.
- `scan`: It scans the entire HBase table and displays the table contents. We can specify optional parameters such as `TIMERANGE`, `TIMESTAMP`, `FILTER`, `LIMIT`, `STARTROW`, `STOPROW`, `MAXLENGTH`, `COLUMNS`, `CACHE`, `VERSIONS`, etc.
- `delete`: It deletes a cell value at the defined table of row or column coordinates.
- `truncate`: It deletes the entire contents in a HBase table. This command internally performs 3 operations - disabling the table, dropping the table and then recreating the table.

Now, we will see how to use the above commands to get the HBase status, version, and create a namespace and a table, insert data into table and retrieve contents of it.

4.1. HBase Status:

On the `hbase>` prompt, run the following command to get the status of HBase cluster.

```
status
```

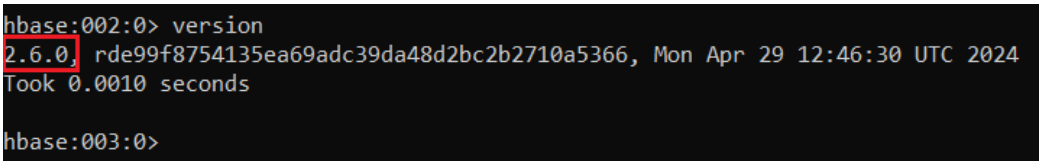


Here, we can see that there is **1 active master** and **0 backup masters** available out of which **1 server is active** and **0 servers are dead**. The **average load value is 2.00.**

4.2. HBase Version:

Run the following command to get the installed version of HBase.

```
version
```

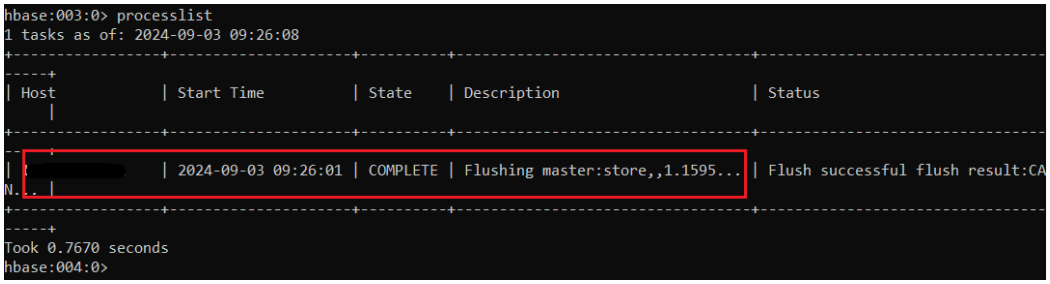


Here, we can see that HBase version as **2.6.0**.

4.3. Process List:

Run the following command to get the list of process running or completed in HBase.

```
processlist
```

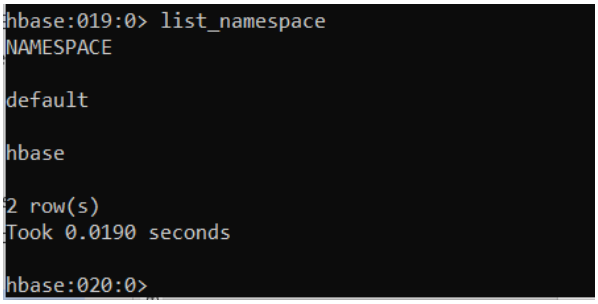


Here, we can see that one process of flushing the master has been completed.

4.4. List Namespace:

On `hbase>` prompt, use `list_namespace` command to get the list of existing namespaces available on the HBase database as below

```
list_namespace
```



4.5. Create Namespace:

Create a namespace using `create_namespace` command in HBase database. It takes a namespace name as mandatory argument.

Run the following command to create a namespace called `'test'` in our HBase

```
create_namespace 'test'
```

```
hbase:012:0> create_namespace 'test'
Took 0.3490 seconds

hbase:013:0>
```

4.6. Describe Namespace:

Get the existing namespace details using `describe_namespace` command.

Run the following command to get the properties of `test` namespace:

```
describe_namespace 'test'
```

```
hbase:021:0> describe_namespace 'test'
DESCRIPTION

{NAME => 'test'}

Quota is disabled
Took 0.0270 seconds

hbase:022:0>
```

4.7. Create Table:

Use `create` command to create a HBase table. It takes a table name and column family as mandatory arguments. The syntax to create a table is as follows:

```
create 'namespace:table_name', 'column_family1', 'column_family2'
```

or

```
create 'namespace:table_name', {NAME => 'column_family1'}, {NAME =>
'column_family2'}
```

Note that namespace is optional here and when it is not specified, the table name is created in the `default` namespace. You can specify the column family as either a simple string or in the form of dictionaries using `NAME` attribute.

Run the following command to create an `employee` table with `emp` column family and `office` column family in `test` namespace.

```
create 'test:employee', 'emp', 'office'
```

```
hbase:024:0> create 'test:employee', 'emp', 'office'
Created table test:employee
Took 1.3180 seconds

=> Hbase::Table - test:employee
hbase:025:0>
```

4.8. List Table:

Use `list` command to show all user tables in HBase

```
list
```

```
hbase:025:0> list
TABLE
test:employee
1 row(s)
Took 0.0480 seconds
=> ["test:employee"]
hbase:026:0>
```

You can also filter the output using regular expression in the `list` command as below

```
list '.*emp.*'
```

```
hbase:016:0> list '.*emp.*'
TABLE

test:employee

1 row(s)
Took 0.0160 seconds

=> ["test:employee"]
hbase:017:0>
```

It displays the list of table names containing `emp` string.

4.9. Describe Table:

Use `describe` command to describe details such as version, compression, blocksize, replication, etc. of HBase table.

Run the following command to get the details of `employee` table available in `test` namespace.

```
describe 'test:employee'
```

```
hbase:019:0> describe 'test:employee'
Table test:employee is ENABLED

test:employee, {TABLE_ATTRIBUTES => {METADATA => {'hbase.store.file-tracker.impl' => 'DEFAULT'}}}

COLUMN FAMILIES DESCRIPTION

{NAME => 'emp', INDEX_BLOCK_ENCODING => 'NONE', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => '

NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', COMP

RESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536 B (64KB)'}


{NAME => 'office', INDEX_BLOCK_ENCODING => 'NONE', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING =

> 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', C

OMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536 B (64KB)'}


2 row(s)
Quota is disabled
Took 0.2410 seconds

hbase:020:0>
```

4.10. Insert Data into Table:

Use `put` command to insert data to rows and columns in a HBase table. This is similar to INSERT statement in RDBMS but the syntax is different as follows:

```
put 'namespace:table_name', 'row_key', 'column_family:column_name',
'value'
```

Run the following commands to create 3 `employee` records using their employee IDs as row key and into columns such as `name`, `gender` under `emp` column family and `address`, `city`, `state`, `country` under `office` column family.

```
put 'test:employee', '101', 'emp:name', 'Scott'
put 'test:employee', '101', 'emp:gender', 'M'
put 'test:employee', '102', 'emp:name', 'Mark'
put 'test:employee', '103', 'emp:name', 'Linda'
put 'test:employee', '103', 'emp:gender', 'F'
put 'test:employee', '102', 'office:address', '101 BayHill Drive'
put 'test:employee', '102', 'office:city', 'Bentonville'
put 'test:employee', '102', 'office:state', 'Arkansas'
put 'test:employee', '102', 'office:country', 'USA'
put 'test:employee', '103', 'office:city', 'Tampa'
put 'test:employee', '103', 'office:state', 'Florida'
put 'test:employee', '103', 'office:country', 'USA'
put 'test:employee', '103', 'office:city', 'Jacksonville'
```

```
hbase:039:0> put 'test:employee', '101', 'emp:name', 'Scott'
Took 0.1380 seconds

hbase:040:0> put 'test:employee', '101', 'emp:gender', 'M'
Took 0.0030 seconds

hbase:041:0> put 'test:employee', '102', 'emp:name', 'Mark'
Took 0.0040 seconds

hbase:042:0> put 'test:employee', '103', 'emp:name', 'Linda'
Took 0.0030 seconds

hbase:043:0> put 'test:employee', '103', 'emp:gender', 'F'
Took 0.0030 seconds

hbase:044:0> put 'test:employee', '102', 'office:address', '101 BayHill Drive'
Took 0.0030 seconds

hbase:045:0> put 'test:employee', '102', 'office:city', 'Bentonville'
Took 0.0030 seconds

hbase:046:0> put 'test:employee', '102', 'office:state', 'Arkansas'
Took 0.0020 seconds

hbase:047:0> put 'test:employee', '102', 'office:country', 'USA'
Took 0.0040 seconds

hbase:048:0> put 'test:employee', '103', 'office:city', 'Tampa'
Took 0.0030 seconds

hbase:049:0> put 'test:employee', '103', 'office:state', 'Florida'
Took 0.0050 seconds

hbase:050:0> put 'test:employee', '103', 'office:country', 'USA'
Took 0.0030 seconds

hbase:051:0> put 'test:employee', '103', 'office:city', 'Jacksonville'
Took 0.0110 seconds

hbase:052:0>
```

Note that the last command inserts a new value `Jacksonville` at `city` column in `office` column family for the `103` row key. When you put a new value at the same column for the same row key, then HBase does not do any update but it assigns the same column with a new timestamp so that when you retrieve the data, it fetches the latest data from columns.

4.11. Read Data from Table:

We can use `get` and `scan` commands to retrieve data from HBase table depending on whether to retrieve single row or multiple rows of data.

- Use `get` command to get single row data and its columns from a HBase table.

  For example, run the following command to get the row key `102` from `employee` table:

  ```
  get 'test:employee', '102'
  ```

```
hbase:021:0> get 'test:employee', '102'
COLUMN                          CELL

 emp:name                       timestamp=2024-09-03T11:19:05.937, value=Mark

 office:address                 timestamp=2024-09-03T11:19:37.189, value=101 BayHill Drive

 office:city                    timestamp=2024-09-03T11:19:43.373, value=Bentonville

 office:country                 timestamp=2024-09-03T11:19:52.748, value=USA

 office:state                   timestamp=2024-09-03T11:19:48.119, value=Arkansas

1 row(s)
Took 0.0720 seconds

hbase:022:0>
```

Run the following command to get the row key `103` with `emp:name` and `emp:gender` columns only from `employee` table:

```
get 'test:employee', '103', {COLUMNS =>
['emp:name','emp:gender']}
```

```
hbase:001:0> get 'test:employee', '103', {COLUMNS => ['emp:name','emp:gender']}
COLUMN                          CELL

 emp:gender                     timestamp=2024-09-03T11:19:15.729, value=F

 emp:name                       timestamp=2024-09-03T11:19:11.129, value=Linda

1 row(s)
Took 1.4350 seconds

hbase:002:0>
```

- Use `scan` command to fetch multiple rows of data from a HBase table. By default, it fetches all data from the table.

  Run the following command to fetch all records from `employee` table available in `test` namespace.

  ```
  scan 'test:employee'
  ```

```
hbase:002:0> scan 'test:employee'
ROW                             COLUMN+CELL

 101                            column=emp:gender, timestamp=2024-09-03T11:18:59.202, value=M

 101                            column=emp:name, timestamp=2024-09-03T11:18:54.252, value=Scott

 102                            column=emp:name, timestamp=2024-09-03T11:19:05.937, value=Mark

 102                            column=office:address, timestamp=2024-09-03T11:19:37.189, value=101 BayHill Drive

 102                            column=office:city, timestamp=2024-09-03T11:19:43.373, value=Bentonville

 102                            column=office:country, timestamp=2024-09-03T11:19:52.748, value=USA

 102                            column=office:state, timestamp=2024-09-03T11:19:48.119, value=Arkansas

 103                            column=emp:gender, timestamp=2024-09-03T11:19:15.729, value=F

 103                            column=emp:name, timestamp=2024-09-03T11:19:11.129, value=Linda

 103                            column=office:city, timestamp=2024-09-03T11:27:13.116, value=Jacksonville

 103                            column=office:country, timestamp=2024-09-03T11:20:17.180, value=USA

 103                            column=office:state, timestamp=2024-09-03T11:20:09.770, value=Florida

3 row(s)
Took 0.3960 seconds

hbase:003:0>
```

The `scan` command can take additional parameters in dictionary format to retrieve specific columns with specific rows of data.

Run the below command to retrieve the first 2 rows with `emp:name` and `office:country` columns:

```
scan 'test:employee',{COLUMNS => ['emp:name','office:country'],
LIMIT => 2}
```

```
hbase:019:0> scan 'test:employee',{COLUMNS => ['emp:name','office:country'], LIMIT => 2}
ROW                             COLUMN+CELL

 101                            column=emp:name, timestamp=2024-09-03T11:18:54.252, value=Scott

 102                            column=emp:name, timestamp=2024-09-03T11:19:05.937, value=Mark

 102                            column=office:country, timestamp=2024-09-03T11:19:52.748, value=USA

2 row(s)
Took 0.0230 seconds

hbase:020:0>
```

Run the below command to retrieve `emp:name` and `emp:gender` columns from starting row key `102` and ending row key `103`:

```
scan 'test:employee',{COLUMNS => ['emp:name','emp:gender'],
STARTROW => '102', STOPROW => '103'}
```

```
hbase:020:0> scan 'test:employee',{COLUMNS => ['emp:name','emp:gender'], STARTROW => '102', STOPROW => '103'}
ROW                             COLUMN+CELL

 102                            column=emp:name, timestamp=2024-09-03T11:19:05.937, value=Mark

1 row(s)
Took 0.0090 seconds

hbase:021:0>
```

4.12. Disable Table:

Use `disable` command to disable a table which is needed before deleting a table or changing its setting in HBase.

Run the following command to disable `employee` table available in `test` namespace.

```
disable 'test:employee'
```

```
hbase:002:0> disable 'test:employee'
Took 1.5640 seconds

hbase:003:0>
```

Use `is_disabled` command as below to check the disabled status of a table

```
is_disabled 'test:employee'
```

```
hbase:005:0> is_disabled 'test:employee'
true
Took 0.0150 seconds
=> true
hbase:006:0>
```

Here, it displays the status as true which means that our table is disabled already

When you describe the disabled table, it shows the table status as disabled:

```
describe 'test:employee'
```

```
hbase:004:0> describe 'test:employee'
Table test:employee is DISABLED
test:employee, {TABLE_ATTRIBUTES => {METADATA => {'hbase.store.file-tracker.impl' => 'DEFAULT'}}}
COLUMN FAMILIES DESCRIPTION
{NAME => 'emp', INDEX_BLOCK_ENCODING => 'NONE', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => '

NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', COMP

RESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536 B (64KB)'}

{NAME => 'office', INDEX_BLOCK_ENCODING => 'NONE', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING =

> 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', C

OMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536 B (64KB)'}

2 row(s)
Quota is disabled
Took 0.1160 seconds
hbase:005:0>
```

When you try to fetch data from the disabled table, it throws an error:

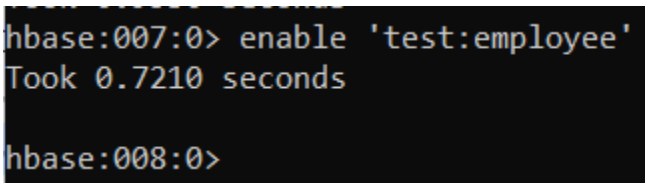
```
hbase:006:0> scan 'test:employee'
ROW                             COLUMN+CELL
org.apache.hadoop.hbase.TableNotEnabledException: test:employee is disabled.
        at org.apache.hadoop.hbase.client.ScannerCallable.prepare(ScannerCallable.java:162)
        at org.apache.hadoop.hbase.client.ScannerCallableWithReplicas$RetryingRPC.prepare(ScannerCallableWithReplicas.ja
va:449)
        at org.apache.hadoop.hbase.client.RpcRetryingCallerImpl.callWithRetries(RpcRetryingCallerImpl.java:102)
        at org.apache.hadoop.hbase.client.ResultBoundedCompletionService$QueueingFuture.run(ResultBoundedCompletionServi
ce.java:74)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:750)

ERROR: Table test:employee is disabled!

For usage try 'help "scan"'

Took 0.9530 seconds
hbase:007:0>
```

4.13. Enable Table:

Use `enable` command to enable a table which is needed to perform any regular commands

Run the following command to enable `employee` table available in `test` namespace.

```
enable 'test:employee'
```



Use `is_enabled` command as below to check the enabled status of a table

```
is_enabled 'test:employee'
```

```
hbase:008:0> is_enabled 'test:employee'
true

Took 0.0110 seconds

=> true
hbase:009:0>
```

Here, it displays the status as true which means that our table is enabled already

When you describe the enabled table, it shows the table status as enabled:

```
describe 'test:employee'
```

```
hbase:009:0> describe 'test:employee'
Table test:employee is ENABLED

test:employee, {TABLE_ATTRIBUTES => {METADATA => {'hbase.store.file-tracker.impl' => 'DEFAULT'}}}

COLUMN FAMILIES DESCRIPTION

{NAME => 'emp', INDEX_BLOCK_ENCODING => 'NONE', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => '

NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', COMP

RESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536 B (64KB)'}


{NAME => 'office', INDEX_BLOCK_ENCODING => 'NONE', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING =

> 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', C

OMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536 B (64KB)'}


2 row(s)
Quota is disabled
Took 0.0610 seconds

hbase:010:0>
```

4.14. Delete Rows from Table:

HBase provides `delete` and `deleteall` commands to delete a single cell or multiple cells at a given row in a table.

- Use `delete` command to delete a cell at specified row in a table.
  For example, run the following command to remove `office:state` column at `102` row key from `employee` table:

  ```
  delete 'test:employee', '102', 'office:state'
  ```

  ```
  hbase:015:0> delete 'test:employee', '102', 'office:state'
  Took 0.0050 seconds
  hbase:016:0>
  ```

When you scan the `employee` table, you will notice that `office:state` column has been removed at `102` row key only while this column data is present for other rows.

```
scan 'test:employee'
```

```
hbase:020:0> scan 'test:employee'
ROW                             COLUMN+CELL

 101                            column=emp:gender, timestamp=2024-09-03T11:18:59.202, value=M

 101                            column=emp:name, timestamp=2024-09-03T11:18:54.252, value=Scott

 102                            column=emp:name, timestamp=2024-09-03T11:19:05.937, value=Mark

 102                            column=office:address, timestamp=2024-09-04T10:41:59.221, value=101 BayHill Drive

 102                            column=office:city, timestamp=2024-09-03T11:19:43.373, value=Bentonville

 102                            column=office:country, timestamp=2024-09-03T11:19:52.748, value=USA

 103                            column=emp:gender, timestamp=2024-09-03T11:19:15.729, value=F

 103                            column=emp:name, timestamp=2024-09-03T11:19:11.129, value=Linda

 103                            column=office:city, timestamp=2024-09-03T11:27:13.116, value=Jacksonville

 103                            column=office:country, timestamp=2024-09-03T11:20:17.180, value=USA

 103                            column=office:state, timestamp=2024-09-03T11:20:09.770, value=Florida

3 row(s)
Took 0.0750 seconds

hbase:021:0>
```

- Use `deleteall` command to delete all cells at specified row in a table.
  For example, run the following command to remove all columns data at `103` row key from `employee` table:

```
deleteall 'test:employee', '103'
```

```
hbase:021:0> deleteall 'test:employee', '103'
Took 0.0040 seconds

hbase:022:0>
```

When you scan the `employee` table, you will notice that `103` row key has been removed completely.

```
scan 'test:employee'
```

```
hbase:022:0> scan 'test:employee'
ROW                             COLUMN+CELL

 101                            column=emp:gender, timestamp=2024-09-03T11:18:59.202, value=M

 101                            column=emp:name, timestamp=2024-09-03T11:18:54.252, value=Scott

 102                            column=emp:name, timestamp=2024-09-03T11:19:05.937, value=Mark

 102                            column=office:address, timestamp=2024-09-04T10:41:59.221, value=101 BayHill Drive

 102                            column=office:city, timestamp=2024-09-03T11:19:43.373, value=Bentonville

 102                            column=office:country, timestamp=2024-09-03T11:19:52.748, value=USA

2 row(s)
Took 0.0260 seconds

hbase:023:0>
```

4.15. Drop Table:

Use `drop` command to delete a table from HBase database. Note that you must disable the table before dropping it.

Run the following commands to disable `employee` table and then drop it in `test` namespace.

```
disable 'test:employee'
drop 'test:employee'
```

```
hbase:023:0> disable 'test:employee'
Took 1.0810 seconds
hbase:024:0> drop 'test:employee'
Took 0.4580 seconds
hbase:025:0>
```

HBase also provides `drop_all` command to delete multiple tables using a regular expression as below:

```
drop_all '.*emp.*'
```

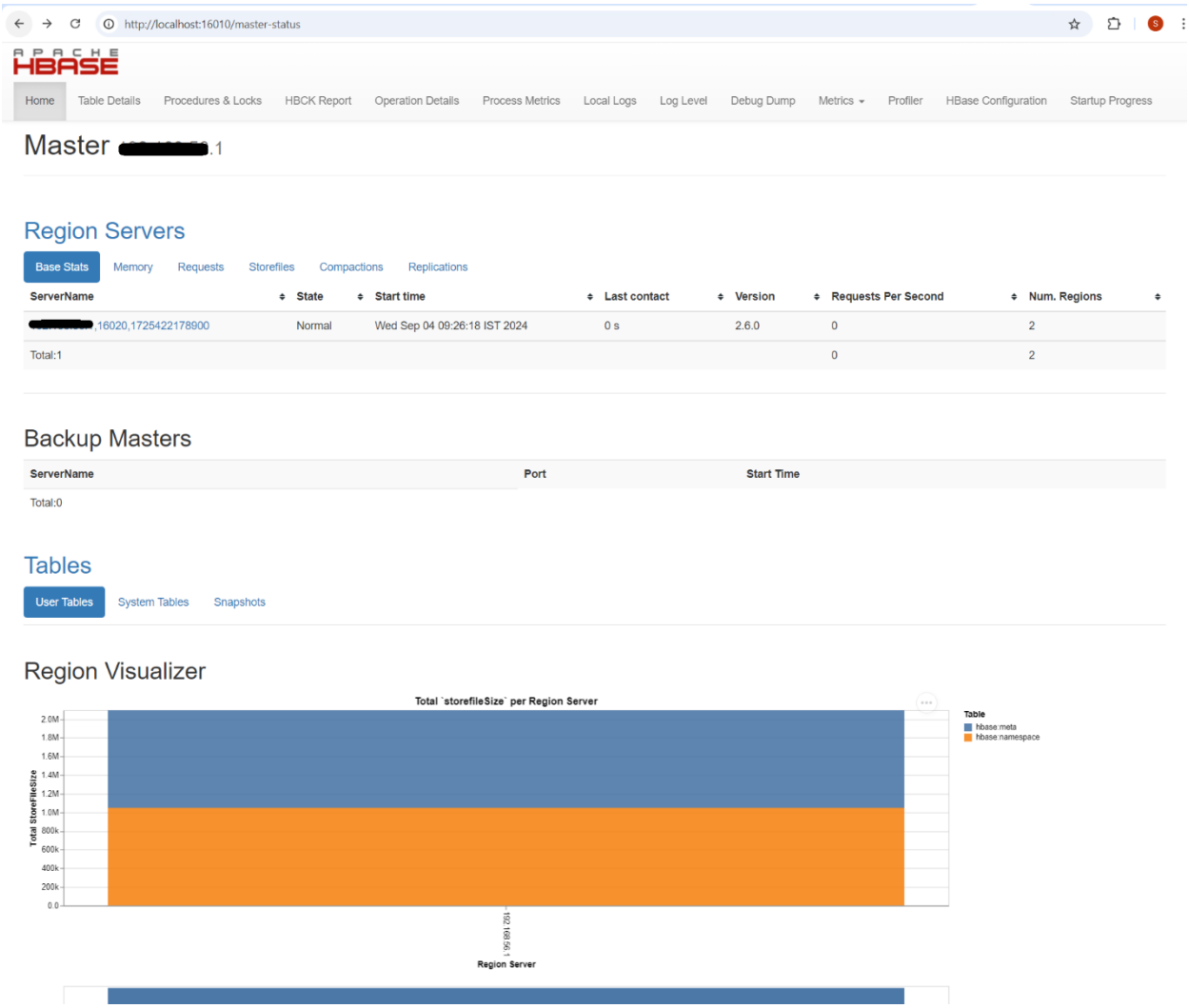
```
hbase:025:0> drop_all '.*emp.*'
No tables matched the regex .*emp.*
Took 0.0250 seconds

hbase:026:0>
```

5. HBase User Interface:

Each HBase service provides a simple Web UI, which gives us the basic information about the current cluster state, change log level, access logs and many more. These Web UIs are useful for troubleshooting and diagnostics. The Web UIs are embedded using the Jetty Web Server (which has a low memory footprint and fast response times), so they start when we run the relevant service.

5.1. Master Web UI:

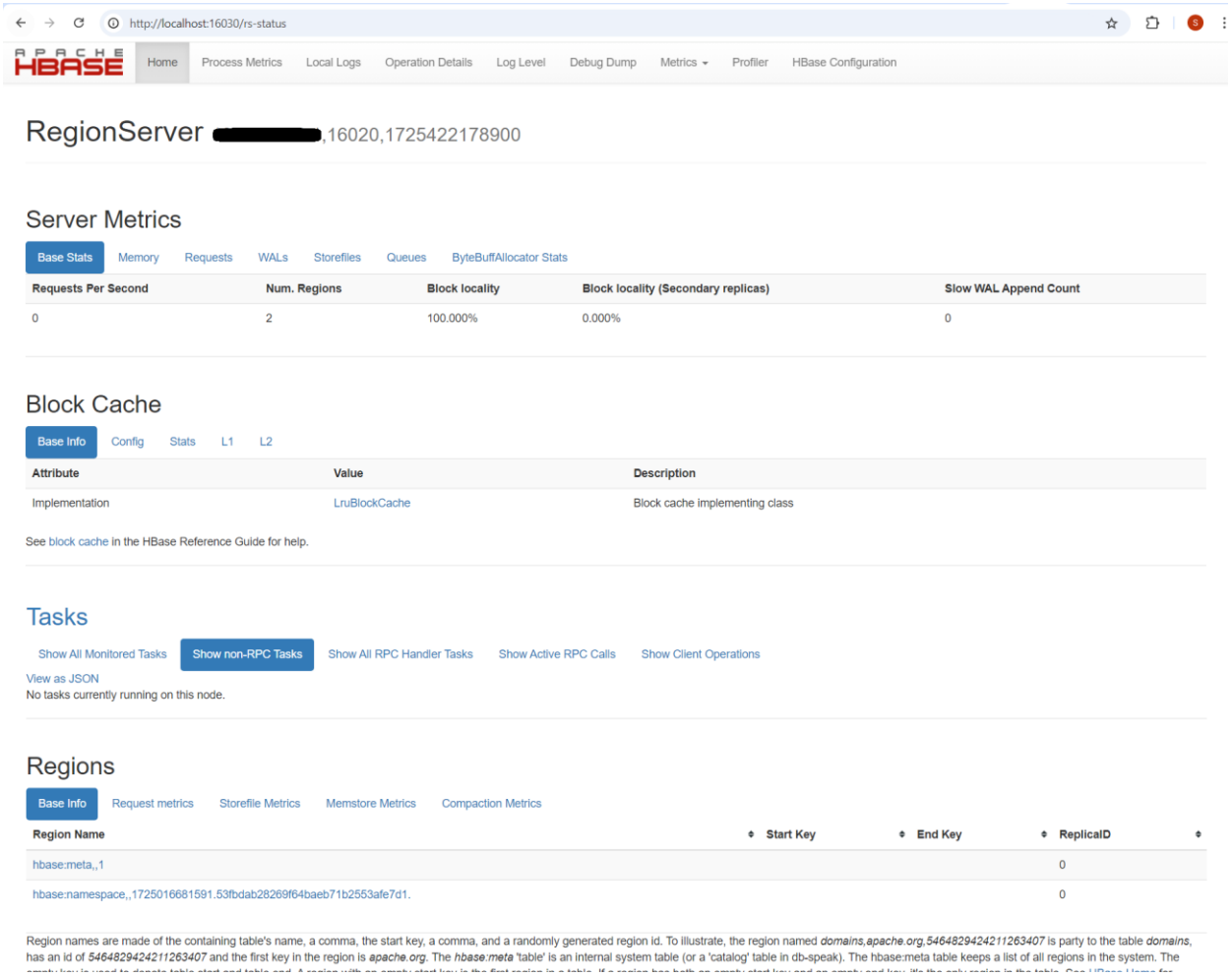
The Master server web UI helps to check the overall health of HBase system and displays the currently running Region servers, backup masters and HBase tables. By default, the Web UI runs on 16010 port and is accessible at http://localhost:16010/. If multiple masters are running, only the active master shows any information.



The Master web UI shows the created tables and their definition *(such as ColumnFamilies, blocksize, etc.)*. It also lists the summary of available RegionServers in the cluster stating how many regions each RegionServer is serving, how many requests per second they are handling, usedHeap, maxheap, etc.

5.2. Region Web UI:

HBase also provides a web UI for Region Server runs on 16030 port by default and is accessible at http://localhost:16030/. This web interface provides the detailed information about the performance of each Region server and data it is providing.



The **Server Metrics** section tells us how each Region server is working by showing how many regions it is hosting and how many requests per second it is serving. The **Tasks** sections displays the long-running tasks and their status.

6. Access HBase from Hive:

Since we are more comfortable on running SQL like queries on any database, we can leverage Apache Hive to access HBase and get the benefit of SQL syntax and run queries on HBase tables. To do so, we must use **HBaseStorageHandler** java class from `hive-hbase-handler.jar` file to register HBase tables with the Hive metastore.

First, make sure that Apache Hive is installed in your machine. If not already installed, follow this guide and install it.

6.1. Verify Hadoop Installation:

Open **Windows PowerShell** or **Command Prompt** and run the below command to verify if the Hadoop version:

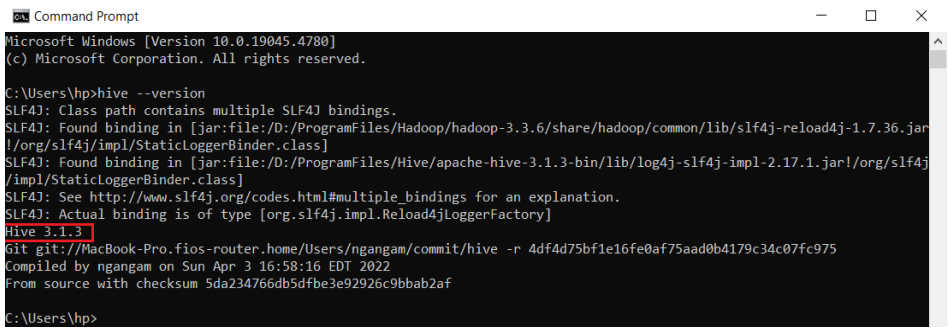
```
hadoop version
```

It displays **Hadoop 3.3.6** version that has been installed.

6.2. Verify Hive Installation:

Open **Windows PowerShell** or **Command Prompt** and run the below command to verify if the Hive version:

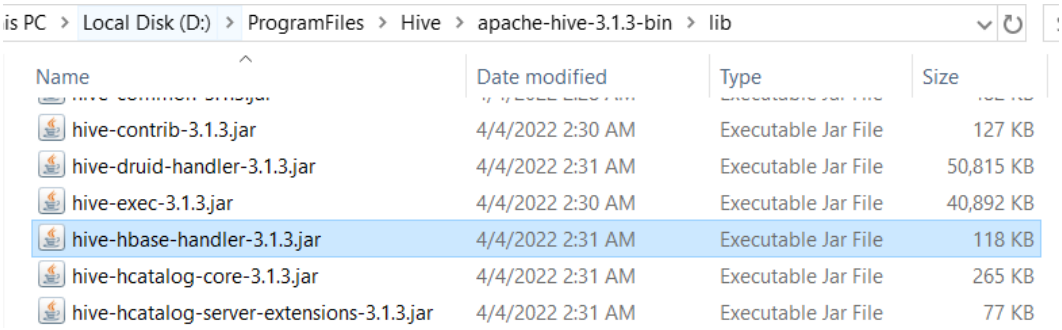
```
hive --version
```



This may take couple of minutes to complete and displays **Hive 3.1.3** version was installed.

6.3. Verify Hive HBase Handler:

Go to `%HIVE_HOME%\lib` directory and verify `hive-hbase-handler-x.x.x.jar` file is available.



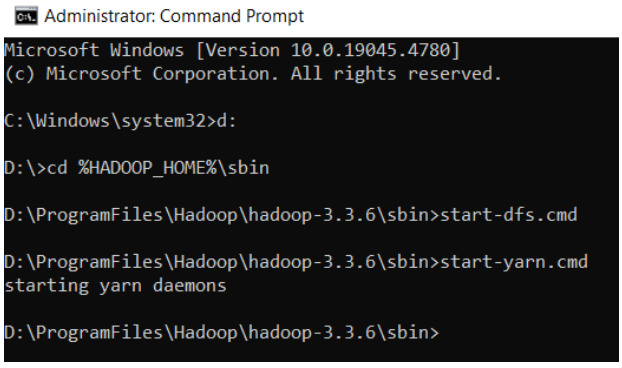
Note: If you can't find `hive-hbase-handler-x.x.x.jar` file in your Hive installation path, you can download it for the corresponding Hive version from Maven Repository and place it in `%HIVE_HOME%\lib` location.

6.4. Start Hadoop Services:

Before starting Hive, Hadoop services must be running since Hive runs on top of HDFS.

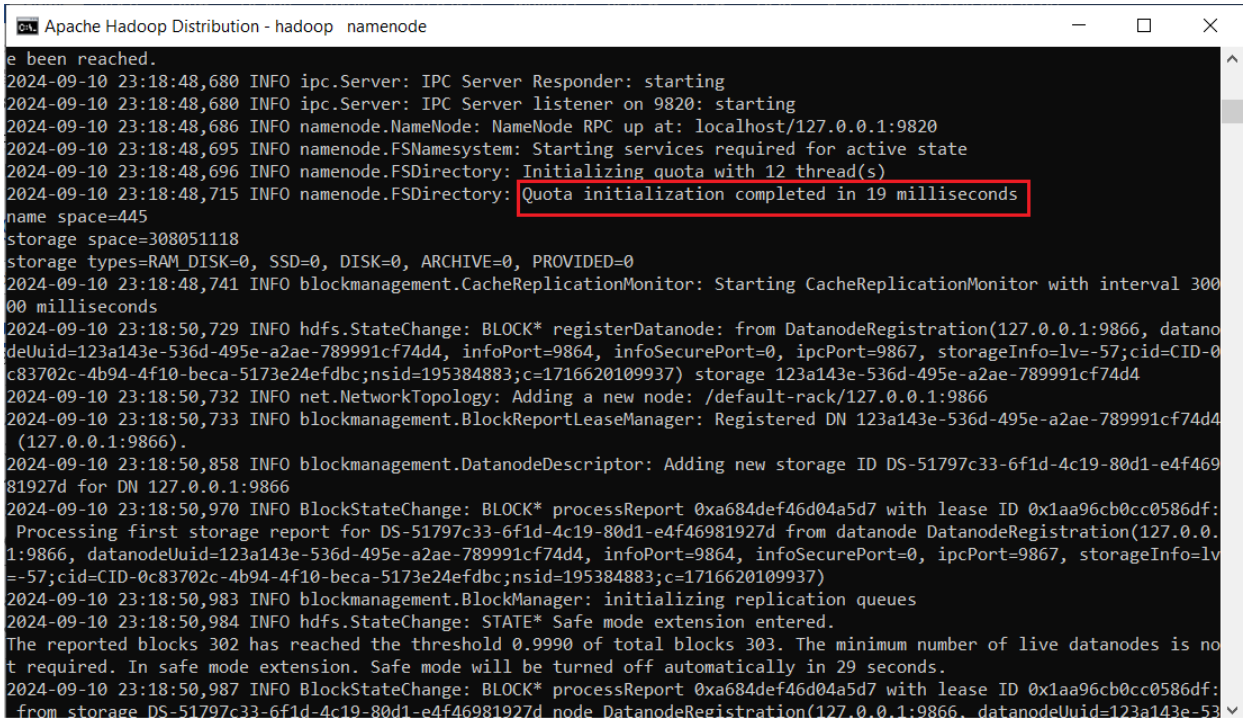
Open **Windows Command Prompt** or **Windows PowerShell** in **Administrator** mode and run the following commands to start Hadoop services.

```
D:
cd %HADOOP_HOME%\sbin
start-dfs.cmd
start-yarn.cmd
```

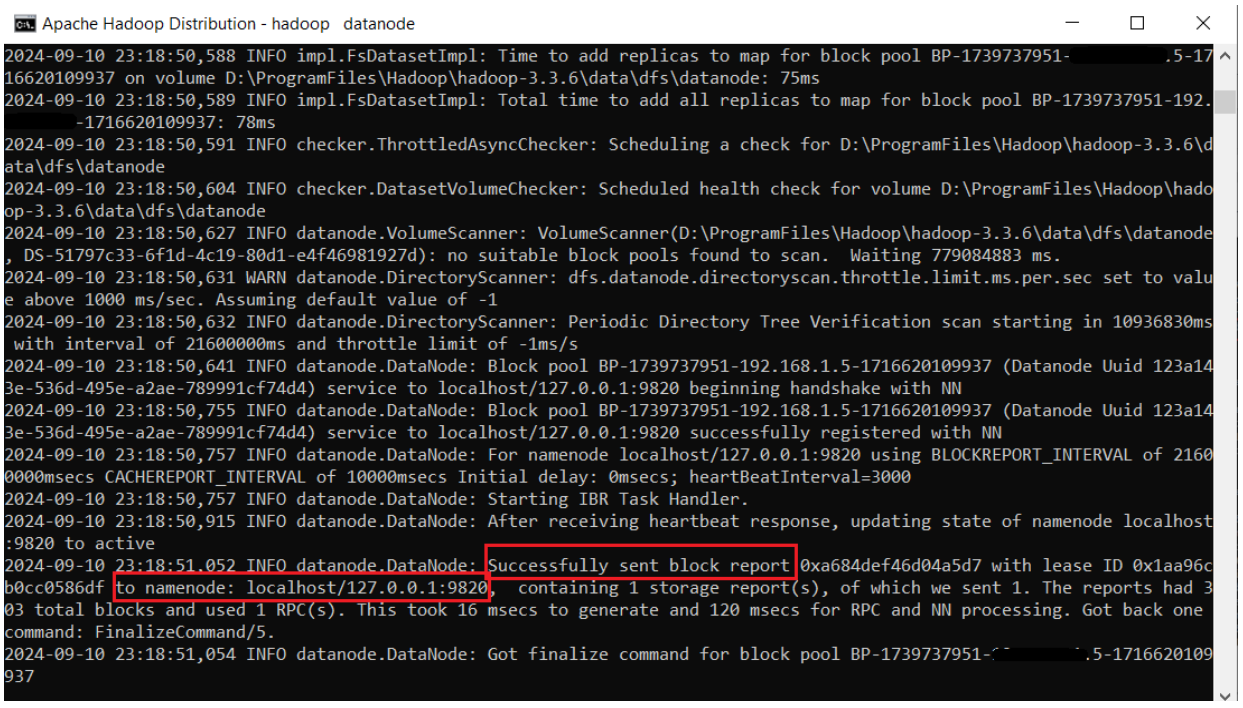


After executing the above commands, we can see four Windows command prompts opened for **namenode**, **datanode, resourcemanager and nodemanager** as below:

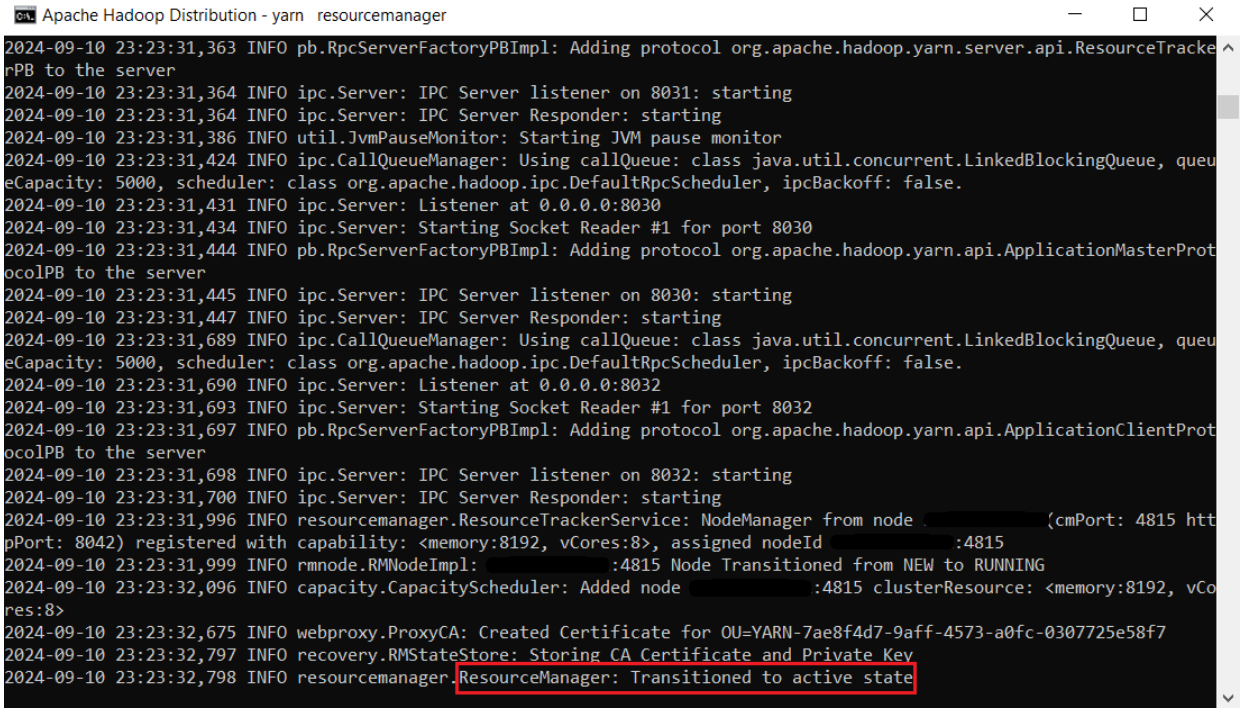
Wait for `namenode` service to say *"Quota initialization completed"*.



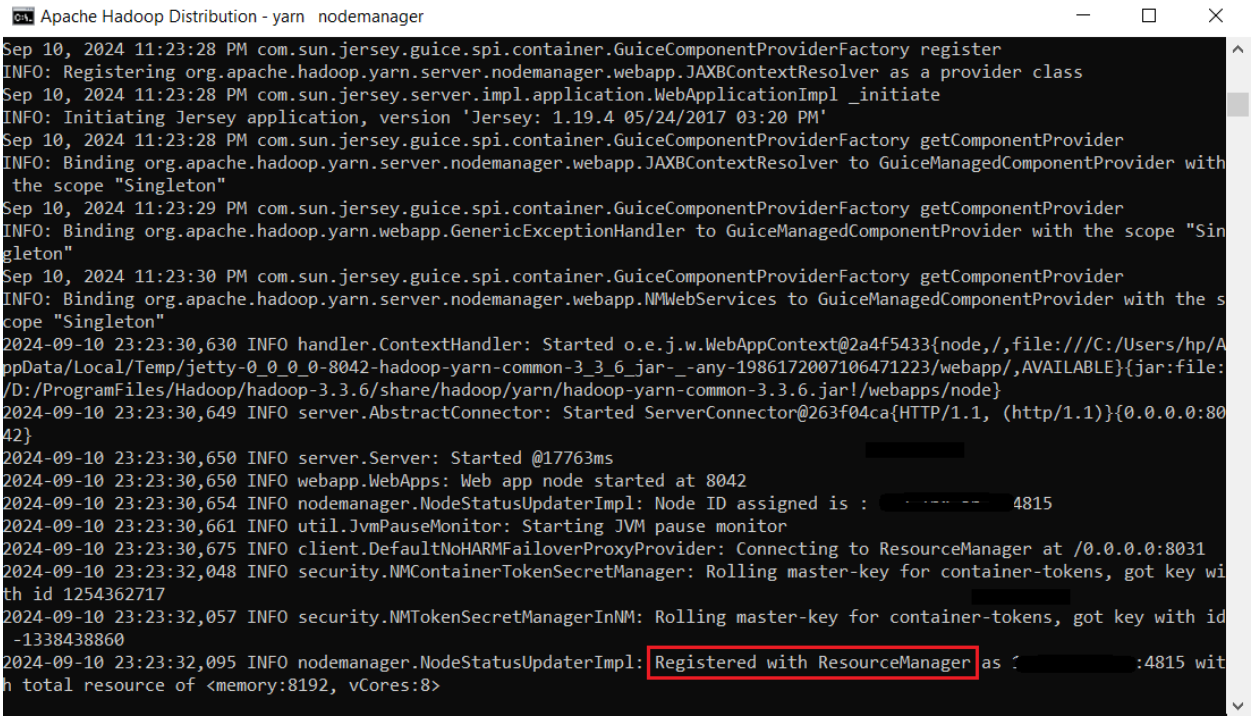
Wait for `datanode` service to say *"Successfully sent block report to namenode: localhost/127.0.0.1:9820"*.



Wait for `resourcemanager` service to say "*Transitioned to active state*".



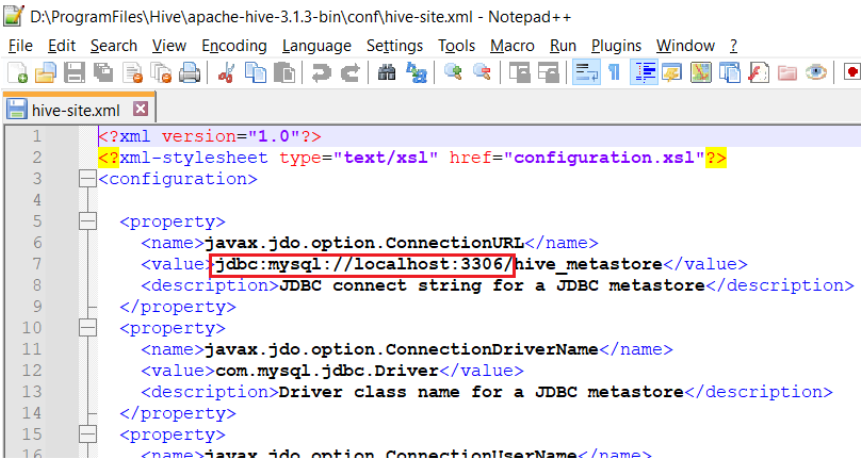
Wait for `nodemanager` service to say "*Registered with ResourceManager*".



6.5. Start Hive Metastore Service:

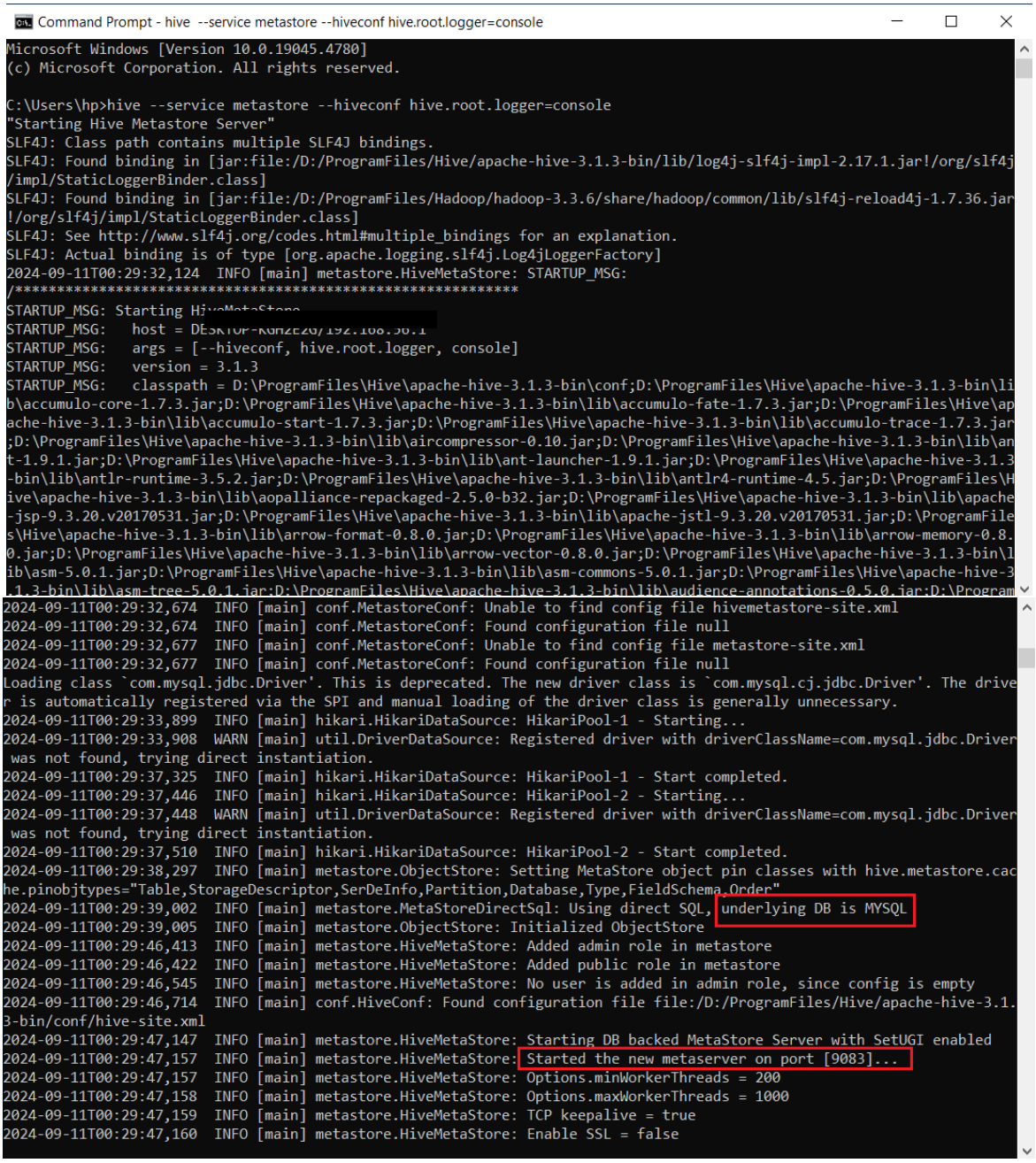
It is necessary to start Hive metastore service if your Hive metastore is configured in Remote database.

Verify your Hive metastore configuration in `%HIVE_HOME%\conf\hive-site.xml` file in which you can look for `javax.jdo.option.ConnectionURL` property which tells the metastore connectivity. In my case, Hive metastore is connected to MySQL running locally.



Since Hive is able to connect to remote metastore using Thrift URIs, start the `metastore` service using the below command:

```
hive --service metastore --hiveconf hive.root.logger=console
```



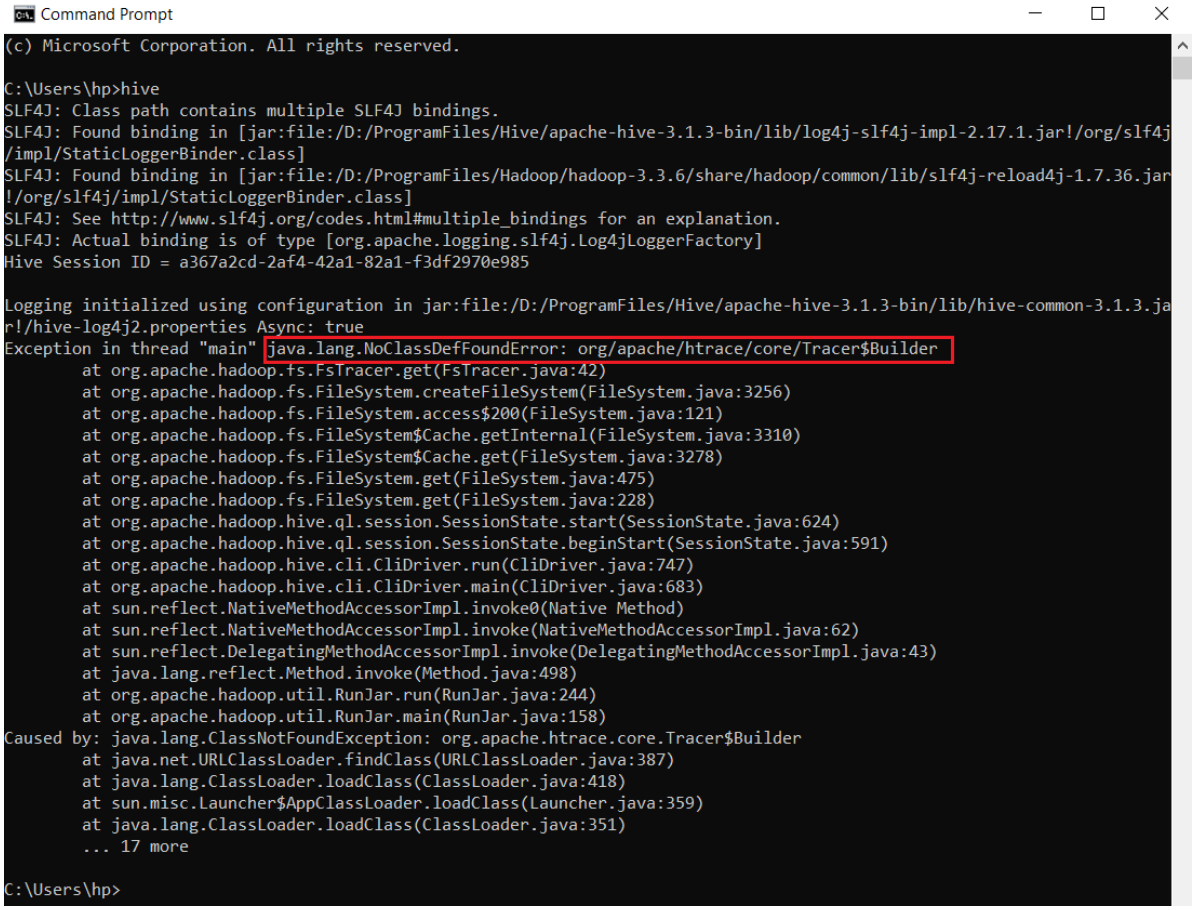
On the console, we can see that **metaserver** started on port **9083** and connected to underlying MySQL database.

6.6. Launch Hive Shell:

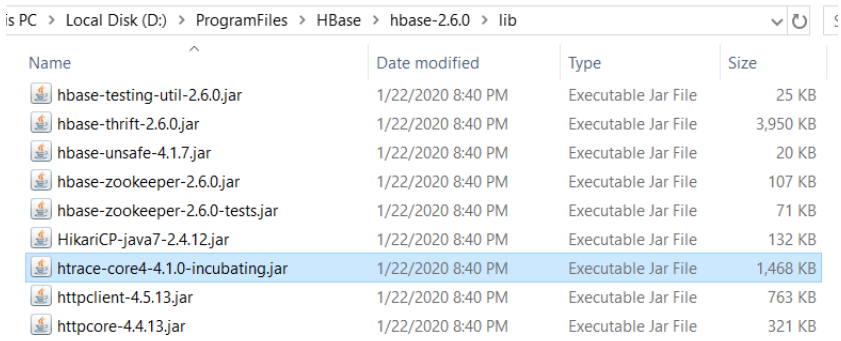
Now, open **Command Prompt** or **Windows PowerShell** and run the following command to launch Hive shell.

```
hive
```

Note: Sometimes, you might face an error *java.lang.NoClassDefFoundError: org/apache/htrace/core/Tracer$Builder* as below:



This happens when `htrace-core.jar` file is missing in `%HBASE_HOME%\lib` directory. To resolve this, go to `%HBASE_HOME%\lib\client-facing-thirdparty` location and copy `htrace-core4-x.x.x-incubating.jar` file to `%HBASE_HOME%\lib` directory.

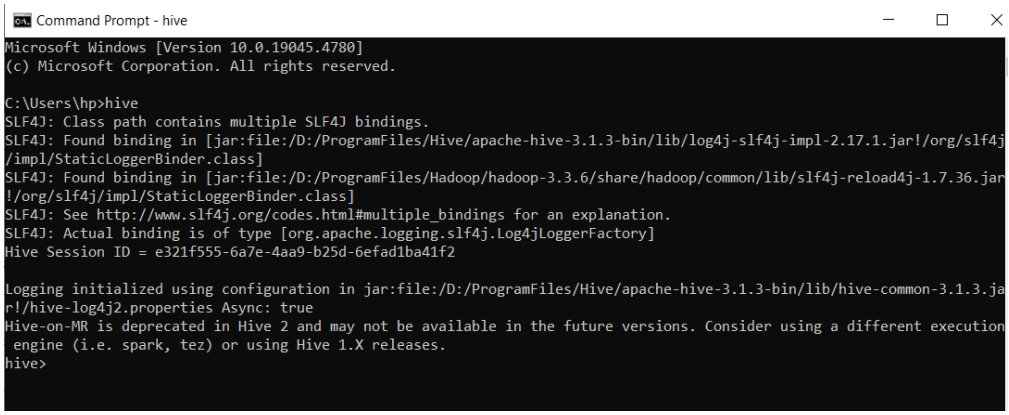


Additionally, I would suggest you to set the following properties which are helpful to disable unnecessary logging on the Hive console:

- Set environment variable `HADOOP_USER_CLASSPATH_FIRST` to `true`.
- Open `hive-config.cmd` file in `%HIVE_HOME%\bin` directory and add the below line at the end of the file.

```
set HADOOP_CLIENT_OPTS=%HADOOP_CLIENT_OPTS% -Dlog4j.configurationFile=hive-log4j2.properties
```

Now, we should be able to launch the hive shell successfully



6.7. Create HBase Table:

Use the following syntax to create a new HBase table managed by Hive using the `STORED BY` clause pointed to `HBaseStorageHandler` class.

```
CREATE TABLE hive_table_name (key datatype, hive_column datatype)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,
hbase_column_family:hbase_column")
TBLPROPERTIES ("hbase.table.name" = "hbase_table_name");
```

Here, the `hbase.columns.mapping` property is required and is used to map the column names between HBase and Hive tables. The `hbase.table.name` property is optional and it controls the name of the table as known by HBase, and allows the Hive table to have a different name. If not specified, then the Hive and HBase table names will be identical.

On the Hive shell, run the following command to create `hbase_departments` table in Hive with HBase table name as `departments`:

```
CREATE TABLE hbase_departments (dept_id int, name string, manager
string, city string, state string, country string, pincode int)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH
SERDEPROPERTIES ("hbase.columns.mapping" = ":key, dept:name,
```

```
dept:manager, location:city, location:state, location:country,
location:pincode") TBLPROPERTIES ("hbase.table.name" =
"departments");
```

```
hive> CREATE TABLE hbase_departments (dept_id int, name string, manager string, city string, state string, country strin
g, pincode int) STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES ("hbase.columns.mappin
g" = ":key, dept:name, dept:manager, location:city, location:state, location:country, location:pincode") TBLPROPERTIES (
"hbase.table.name" = "departments");
OK
Time taken: 7.618 seconds
hive>
```

Describe the newly created table with this hive command:

```
describe formatted hbase_departments;
```

```
hive> describe formatted hbase_departments;
OK
# col_name              data_type               comment
dept_id                 int
name                    string
manager                 string
city                    string
state                   string
country                 string
pincode                 int

# Detailed Table Information
Database:               default
OwnerType:              USER
Owner:                  hp
CreateTime:             Wed Sep 11 19:54:06 IST 2024
LastAccessTime:         UNKNOWN
Retention:              0
Location:               hdfs://localhost:9820/user/hive/warehouse/hbase_departments
Table Type:             MANAGED_TABLE
Table Parameters:
        COLUMN_STATS_ACCURATE   {\"BASIC_STATS\":\"true\",\"COLUMN_STATS\":{\"city\":\"true\",\"country\":\"true\",\"dep
t_id\":\"true\",\"manager\":\"true\",\"name\":\"true\",\"pincode\":\"true\",\"state\":\"true\"}}
        bucketing_version       2
        hbase.table.name        departments
        numFiles                0
        numRows                 0
        rawDataSize             0
        storage_handler         org.apache.hadoop.hive.hbase.HBaseStorageHandler
        totalSize               0
```

6.8. Insert Data into HBase Table:

Run the following command to insert record into `hbase_departments` table.

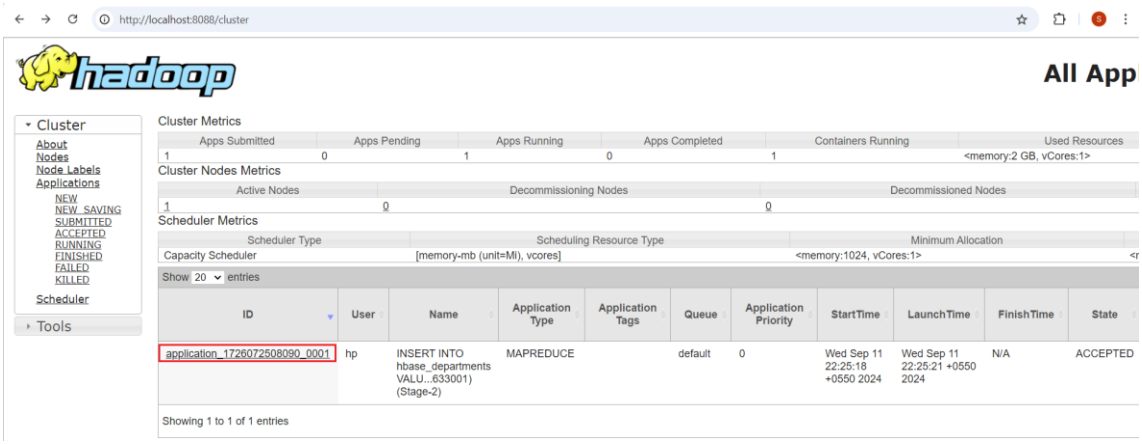
```
INSERT INTO hbase_departments VALUES (10, 'Finance', 'Steve',
'Houston', 'Texas', 'USA', 633001);
```

This Insert statement triggers a MapReduce job that will get executed on the Hadoop cluster. Here, you might get an **Execution error** as shown below

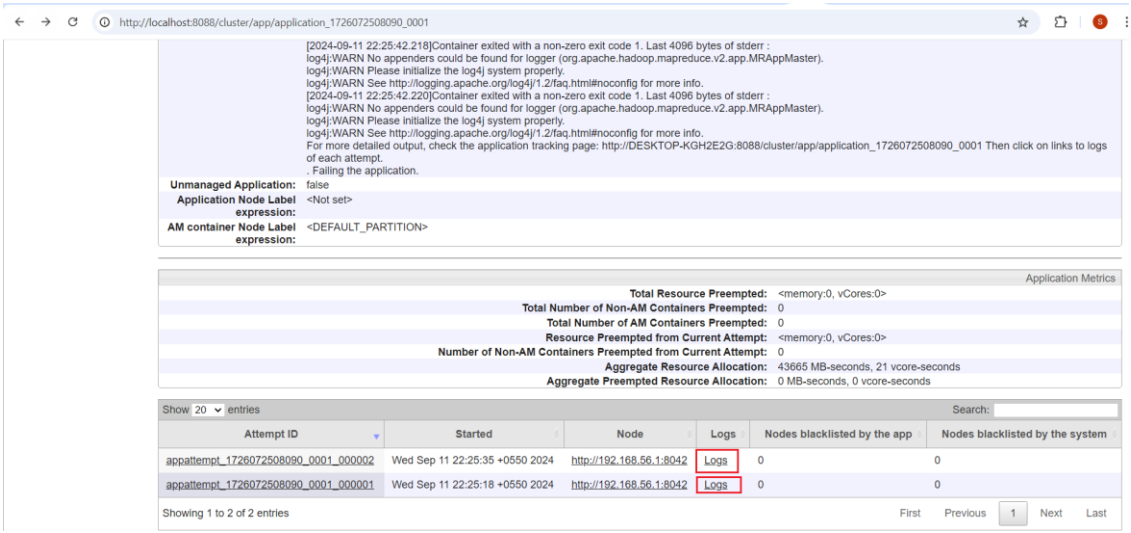
```
hive> INSERT INTO hbase_departments VALUES (10, 'Finance', 'Steve', 'Houston', 'Texas', 'USA', 633001);
Query ID = hp_20240911222452_f2fceb02-40fe-4b59-9f82-3ee69a01ff97
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1726072508090_0001, Tracking URL = http://DESKTOP-KGH2E2G:8088/proxy/application_1726072508090_0001/
Kill Command = D:\ProgramFiles\Hadoop\hadoop-3.3.6\bin\mapred job  -kill job_1726072508090_0001
Hadoop job information for Stage-2: number of mappers: 0; number of reducers: 0
2024-09-11 22:25:43,431 Stage-2 map = 0%,  reduce = 0%
Ended Job = job_1726072508090_0001 with errors
Error during job, obtaining debugging information...
FAILED: Execution Error, return code 2 from org.apache.hadoop.hive.ql.exec.mr.MapRedTask
MapReduce Jobs Launched:
Stage-Stage-2:  HDFS Read: 0 HDFS Write: 0 FAIL
Total MapReduce CPU Time Spent: 0 msec
hive>
```

The details of the error can be seen on the Hadoop YARN UI at http://localhost:8088/cluster

- On the YARN UI, click on the Application ID.



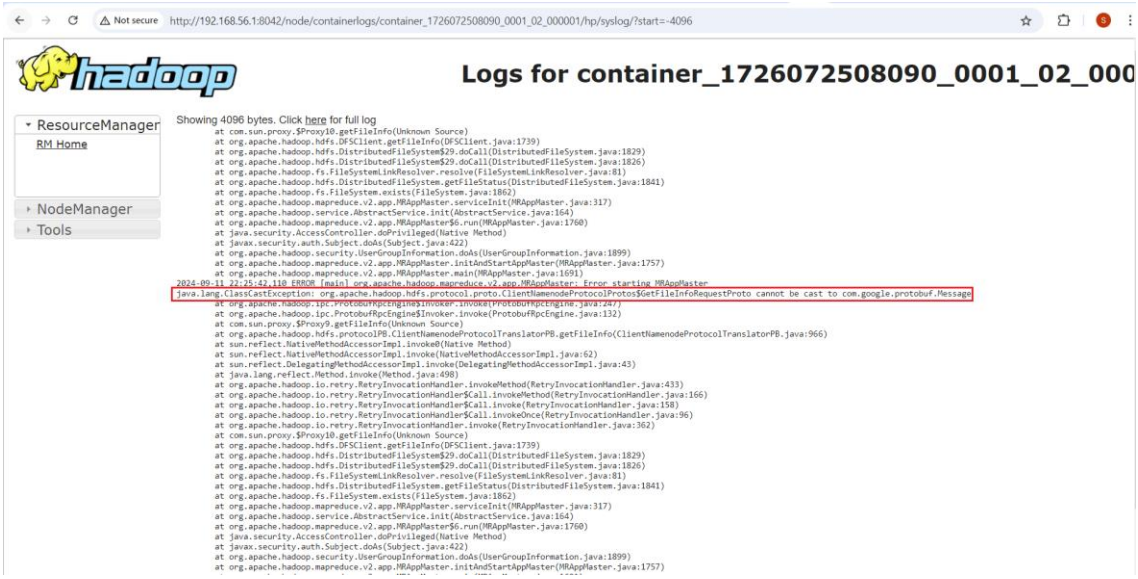
- Click on **Logs** link available for one of the attempts.



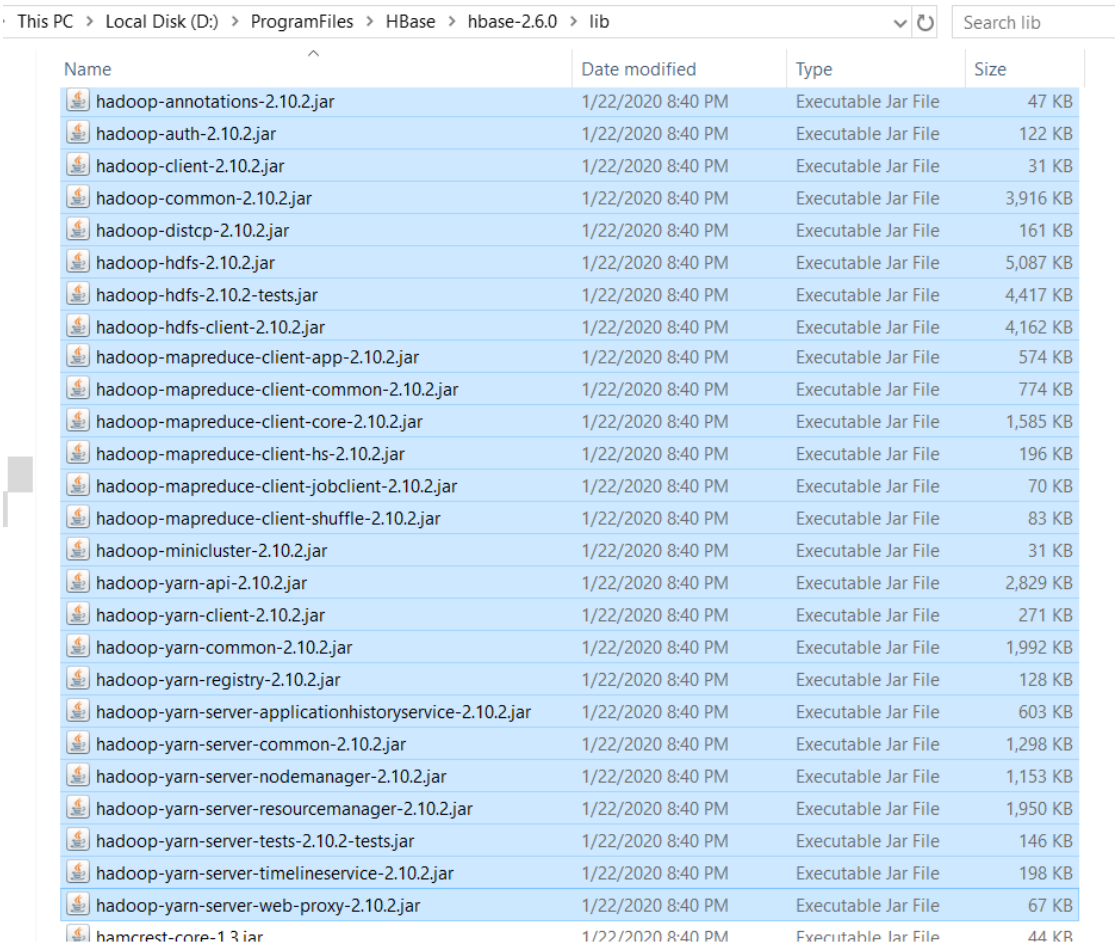
- Select **syslog** link.

- You can see a detailed error message here

```
java.lang.ClassCastException:
org.apache.hadoop.hdfs.protocol.proto.ClientNamenodeProtocolProtos$GetF
ileInfoRequestProto cannot be cast to com.google.protobuf.Message
```



The above `ClassCastException` occurs when HBase is referring to different version of Hadoop libraries compared to libraries available on the existing Hadoop cluster. Go to `%HBASE_HOME%\lib` location where you can see all `hadoop` jars are of **2.10.2** version while our Hadoop is actually running on **3.3.6** version**.**



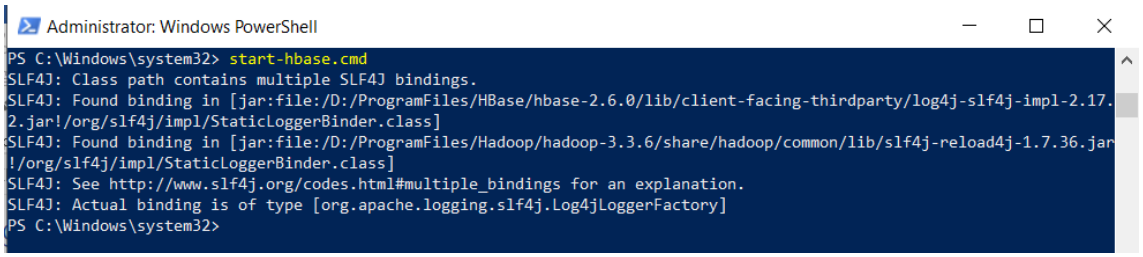
Follow the below steps to resolve the `ClassCastException` issue:

- Close **hive terminal** window, **Hive metastore service** window and **hbase-master start** window.

- Remove or rename all `hadoop*.jar` files in `%HBASE_HOME%\lib` location *(here, I renamed all haddop*.jar files instead of deleting them)*

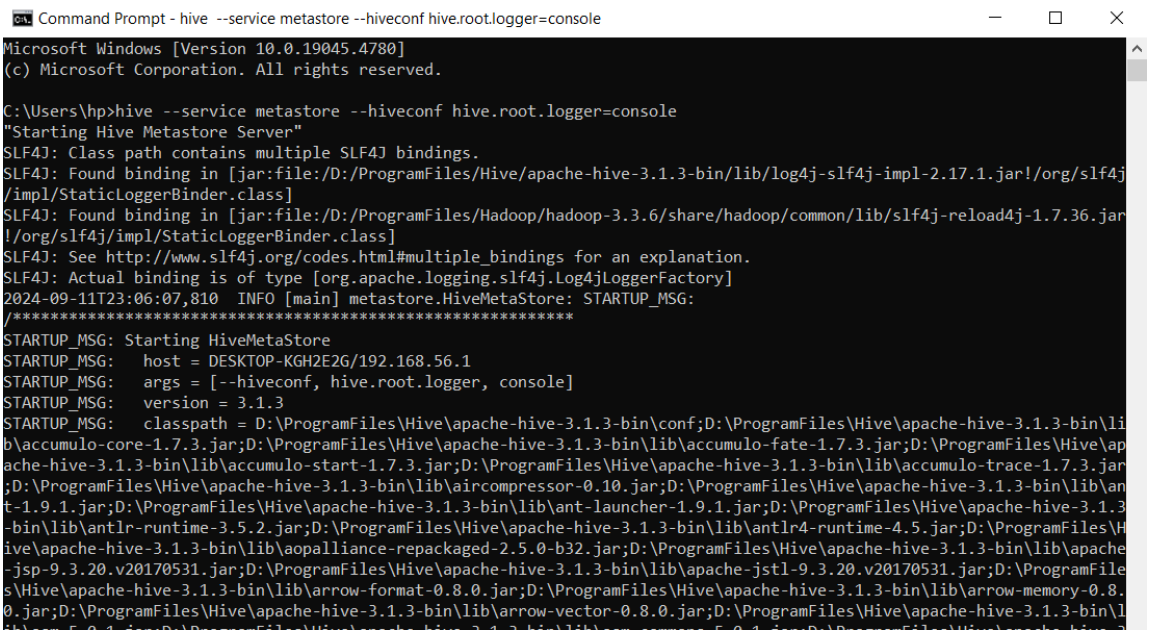
- Open **Windows Powershell** as **Administrator** and run the following command to start HBase cluster.

```
start-hbase.cmd
```



- Open a new Command Prompt and start the `metastore` service using the below command:

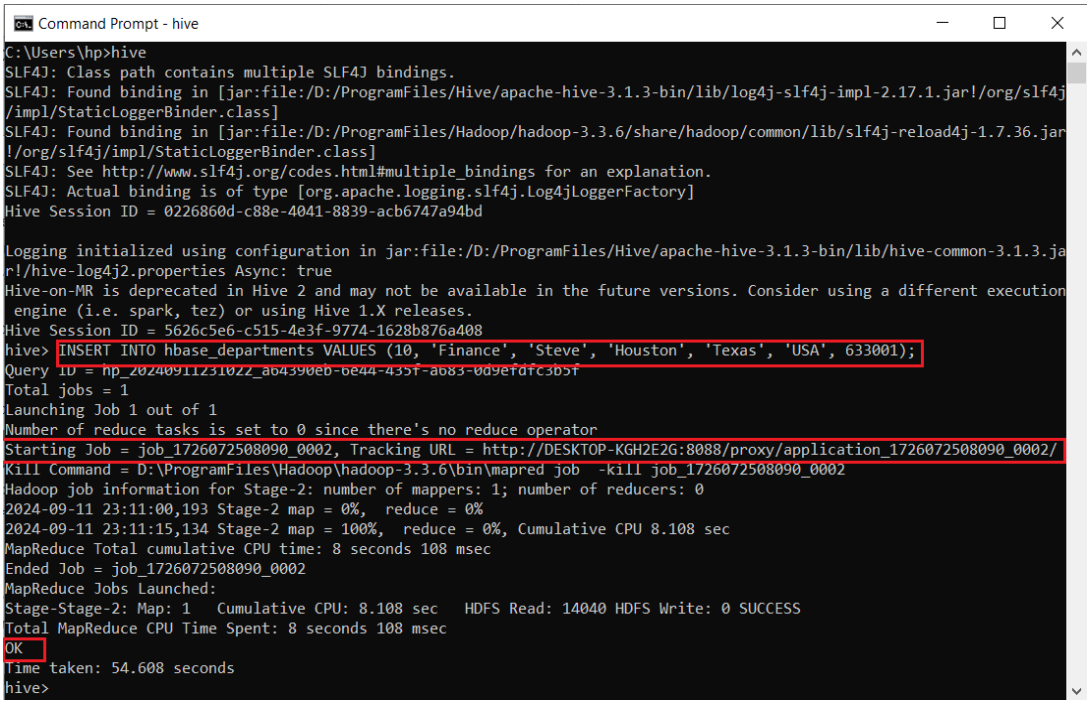
```
hive --service metastore --hiveconf hive.root.logger=console
```



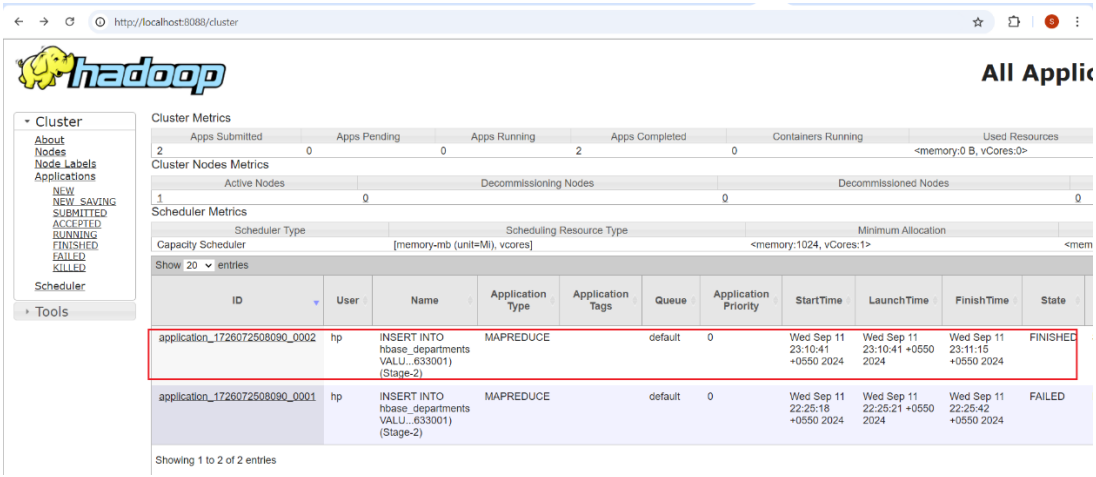
- Open a new Command Prompt, launch Hive shell and run the following insert statement:

```
INSERT INTO hbase_departments VALUES (10, 'Finance', 'Steve',
'Houston', 'Texas', 'USA', 633001);
```

It submitted a new application to the Hadoop YARN and provided application URL which is successful and inserted record into table.



You can notice the same on the YARN UI at http://localhost:8088/cluster where the latest application status is FINISHED successfully.



Let us quickly verify the data in the table by running the following `select` statement.

```
select * from hbase_departments;
```

```
hive> select * from hbase_departments;
OK
10      Finance Steve   Houston Texas   USA     633001
Time taken: 1.09 seconds, Fetched: 1 row(s)
hive>
```

Note: To view the column names in the above result, set `hive.cli.print.header` property to `true` and run the `select` statement.

```
set hive.cli.print.header=true;
select * from hbase_departments;
```

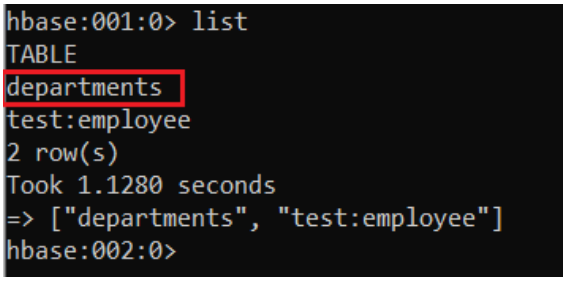
6.9. Validate Newly Created Table in HBase:

Open new Command Prompt and start hbase shell.

```
D:
cd %HBASE_HOME%\bin
hbase shell
```

On `hbase>` prompt, list the available tables

```
list
```



Here, you can see that `departments` table is available which got created from Hive.

Describe the `departments` table:

```
describe "departments"
```

```
hbase:002:0> describe "departments"
Table departments is ENABLED

departments, {TABLE_ATTRIBUTES => {METADATA => {'hbase.store.file-tracker.impl' => 'DEFAULT'}}}

COLUMN FAMILIES DESCRIPTION

{NAME => 'dept', INDEX_BLOCK_ENCODING => 'NONE', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING =>

'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW', IN_MEMORY => 'false', COM

PRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536 B (64KB)', METADATA => {'EVICT_BLOCKS_ON_CLOSE' => 'false'

, 'NEW_VERSION_BEHAVIOR' => 'false', 'CACHE_DATA_ON_WRITE' => 'false', 'CACHE_INDEX_ON_WRITE' => 'false', 'CACHE_BLOOMS_

ON_WRITE' => 'false', 'PREFETCH_BLOCKS_ON_OPEN' => 'false', 'CACHE_DATA_IN_L1' => 'false'}}


{NAME => 'location', INDEX_BLOCK_ENCODING => 'NONE', VERSIONS => '1', KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING

 => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW', IN_MEMORY => 'false',

 COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536 B (64KB)', METADATA => {'EVICT_BLOCKS_ON_CLOSE' => 'fa

lse', 'NEW_VERSION_BEHAVIOR' => 'false', 'CACHE_DATA_ON_WRITE' => 'false', 'CACHE_INDEX_ON_WRITE' => 'false', 'CACHE_BLO

OMS_ON_WRITE' => 'false', 'PREFETCH_BLOCKS_ON_OPEN' => 'false', 'CACHE_DATA_IN_L1' => 'false'}}


2 row(s)
Quota is disabled
Took 0.6860 seconds

hbase:003:0>
```

Scan the `departments` table to see its contents:

```
scan "departments"
```

```
hbase:003:0> scan "departments"
ROW                             COLUMN+CELL

 10                             column=dept:manager, timestamp=2024-09-11T23:11:14.519, value=Steve

 10                             column=dept:name, timestamp=2024-09-11T23:11:14.519, value=Finance

 10                             column=location:city, timestamp=2024-09-11T23:11:14.519, value=Houston

 10                             column=location:country, timestamp=2024-09-11T23:11:14.519, value=USA

 10                             column=location:pincode, timestamp=2024-09-11T23:11:14.519, value=633001

 10                             column=location:state, timestamp=2024-09-11T23:11:14.519, value=Texas

1 row(s)
Took 0.1750 seconds

hbase:004:0>
```

Here, you can see that one record of **row key 10** is available in `departments` table.

6.10. Create External HBase Table:

If you want to give Hive access to an existing HBase table, then use `CREATE EXTERAL TABLE` command on Hive. The syntax is similar to `CREATE TABLE` command:

```
CREATE EXTERNAL TABLE hive_table_name (key datatype, hive_column
datatype)
```

```
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,
hbase_column_family:hbase_column")
TBLPROPERTIES ("hbase.table.name" = "hbase_table_name");
```

Go to hive terminal and run the following command to create `hbase_external_employees` table in Hive:

```
CREATE EXTERNAL TABLE hbase_external_employees (emp_id int, name
string, gender string, address string, city string, state string,
country string) STORED BY
'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH
SERDEPROPERTIES ("hbase.columns.mapping" = ":key, emp:name,
emp:gender, office:address, office:city, office:state,
office:country") TBLPROPERTIES ("hbase.table.name" =
"test:employee");
```

```
hive> CREATE EXTERNAL TABLE hbase_external_employees (emp_id int, name string, gender string, address string, city strin
g, state string, country string) STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES ("hba
se.columns.mapping" = ":key, emp:name, emp:gender, office:address, office:city, office:state, office:country") TBLPROPER
TIES ("hbase.table.name" = "test:employee");
OK
Time taken: 0.43 seconds
hive>
```

External table has been created successfully which means this table is pointed to the existing HBase table "`test:employee`"

6.11. Fetch HBase Data in Hive:

Now, let us run fetch records from `hbase_external_employees` table:

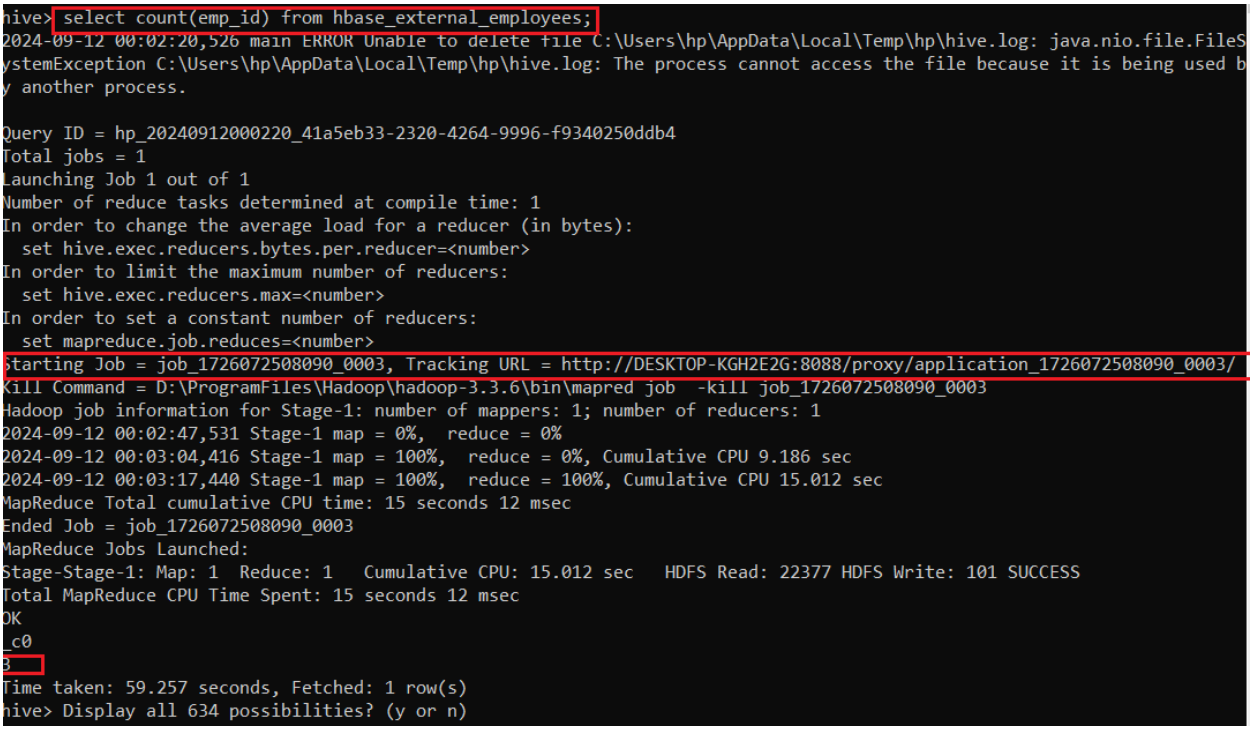
```
select * from hbase_external_employees;
```

```
hive> select * from hbase_external_employees;
OK
hbase_external_employees.emp_id hbase_external_employees.name   hbase_external_employees.gender hbase_external_employees
.address        hbase_external_employees.city   hbase_external_employees.state  hbase_external_employees.country
101     Scott   M       NULL    NULL    NULL    NULL
102     Mark    NULL    101 BayHill Drive       Bentonville     NULL    USA
103     Linda   F       NULL    Jacksonville    Florida USA
Time taken: 0.48 seconds, Fetched: 3 row(s)
hive>
```

It displays 3 records which are available in the existing HBase "`test:employee`" table.

Similarly, we can run SQL queries on Hive which pulls data from HBase.

```
select count(emp_id) from hbase_external_employees;
```



It submits a MapReduce job and up on successful, it displays the count of records in the table.

7. Shutdown Standalone HBase:

We can shutdown the standalone Hbase cluster in the Windows environment.

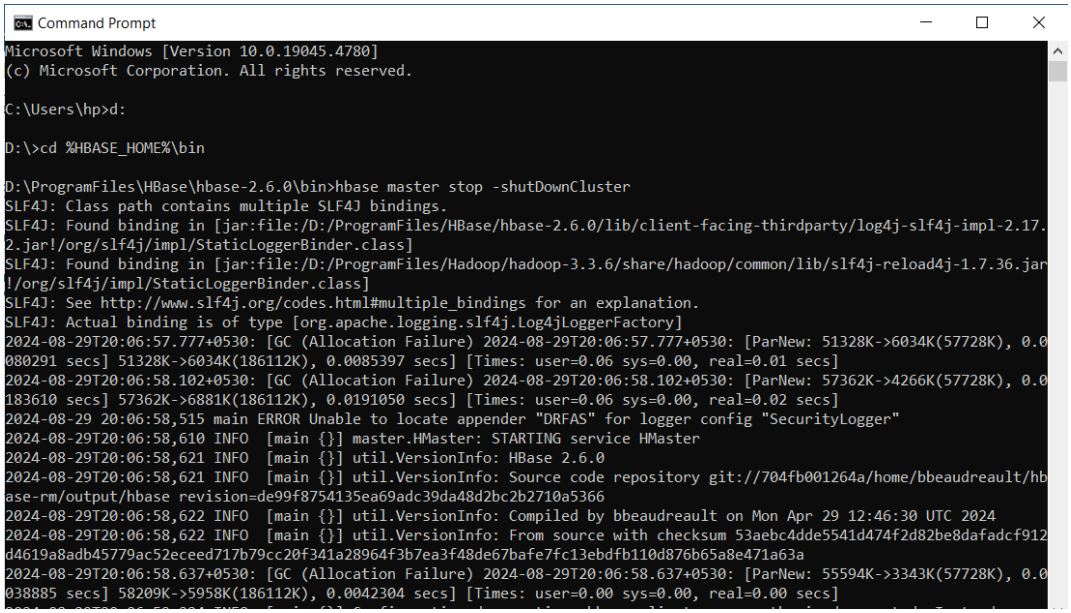
Open **Windows PowerShell** as **Administrator** and shutdown the entire HBase cluster using the below command:

```
hbase master stop -shutDownCluster
```

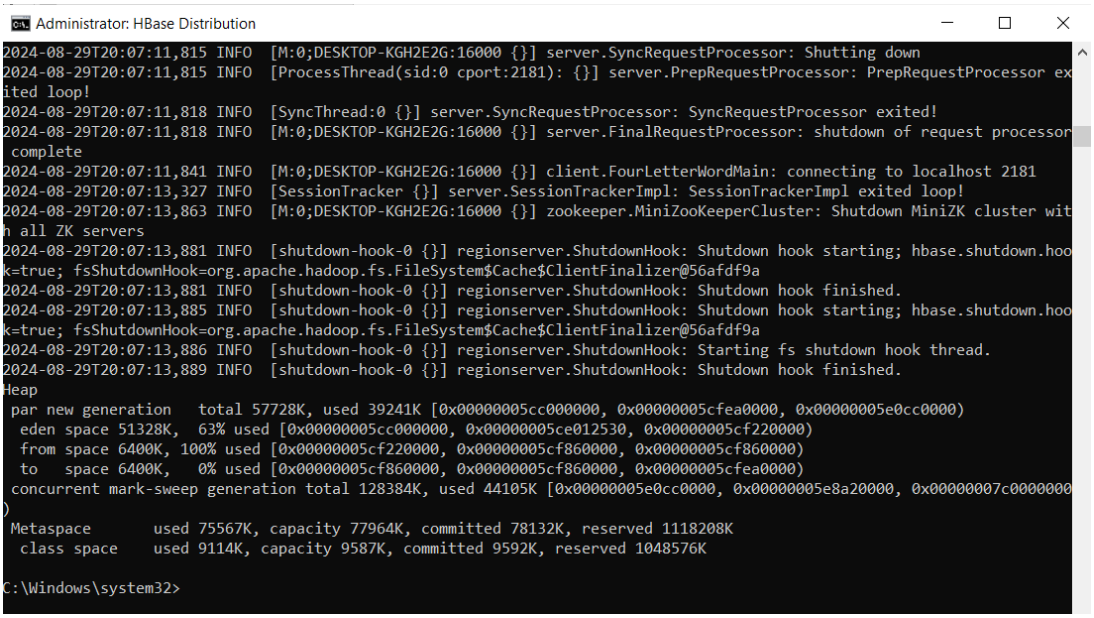
Note:

If you would like to stop hbase using **Command Prompt**, then you must first navigate to the location where HBase is installed as below.

```
D:
cd %HBASE_HOME%\bin
hbase master stop -shutDownCluster
```



This will shut down three daemon services `HRegion`, `HMaster` and `Zookeeper` *(you will see the same in the another Command Prompt where HBase Distribution was running)*



Congratulations!! You have successfully installed Standalone HBase 2.6 in Windows operating system and executed few HBase commands. You also understood how to access HBase tables from Hive and run SQL type queries on HBase.