**Zach Adair – MSDS 610 – Week 3 Lab**

**Using Hbase on your Vagrant Cluster**

**Regis University – College of Computer & Information Sciences**

**Data Science – Data Engineering**

**January 31, 2019**

# Lab Overview

* Resume your Vagrant VMs to work with the cluster using Cmder/Termina/Putty/etc.
* Use the Ambari console to view the status of the cluster and HBase
* Open your HBase database shell to enter commands
* Create a few HBase column families (tables) and then add data
* Query data from the HBase database tables
* Disable and Drop a HBase family (table).
* Suspend your cluster and exit from Cmder/Termina/Putty/etc.

**Briefly (skim) review the sections from the online manual on the open source Apache HBase website:**

**http://hbase.apache.org/book.html#datamodel**

A basic understanding is needed for an understanding of what you are doing in this lab. **Note: there will be a new Data Engineering elective course that will focus exclusively on NoSQL Databases.** This lab is designed to give you some exposure to the concept of a NoSQL Database. Also, you should review the PPT slides in Worldclass about HBase schema design prior to beginning the lab.

[Data Model](http://hbase.apache.org/book.html#datamodel) - [19. Conceptual View](http://hbase.apache.org/book.html#conceptual.view)

[20. Physical View](http://hbase.apache.org/book.html#physical.view)

[21. Namespace](http://hbase.apache.org/book.html#_namespace)

[22. Table](http://hbase.apache.org/book.html#_table)

[23. Row](http://hbase.apache.org/book.html#_row)

[24. Column Family](http://hbase.apache.org/book.html#columnfamily)

[25. Cells](http://hbase.apache.org/book.html#_cells)

[26. Data Model Operations](http://hbase.apache.org/book.html#_data_model_operations)

[27. Versions](http://hbase.apache.org/book.html#versions)

[28. Sort Order](http://hbase.apache.org/book.html#dm.sort)

[29. Column Metadata](http://hbase.apache.org/book.html#dm.column.metadata)

[30. Joins](http://hbase.apache.org/book.html#_joins)

[31. ACID](http://hbase.apache.org/book.html#_acid)

[HBase and Schema Design](http://hbase.apache.org/book.html#schema) - [32. Schema Creation](http://hbase.apache.org/book.html#schema.creation)

[33. On the number of column families](http://hbase.apache.org/book.html#number.of.cfs)

[34. Rowkey Design](http://hbase.apache.org/book.html#rowkey.design)

[35. Number of Versions](http://hbase.apache.org/book.html#schema.versions)

[36. Supported Datatypes](http://hbase.apache.org/book.html#supported.datatypes)

[37. Joins](http://hbase.apache.org/book.html#schema.joins)

[38. Time To Live (TTL)](http://hbase.apache.org/book.html#ttl)

[39. Keeping Deleted Cells](http://hbase.apache.org/book.html#cf.keep.deleted)

[40. Secondary Indexes and Alternate Query Paths](http://hbase.apache.org/book.html#secondary.indexes)

[41. Constraints](http://hbase.apache.org/book.html#_constraints)

# Part 1

## Resume VMs

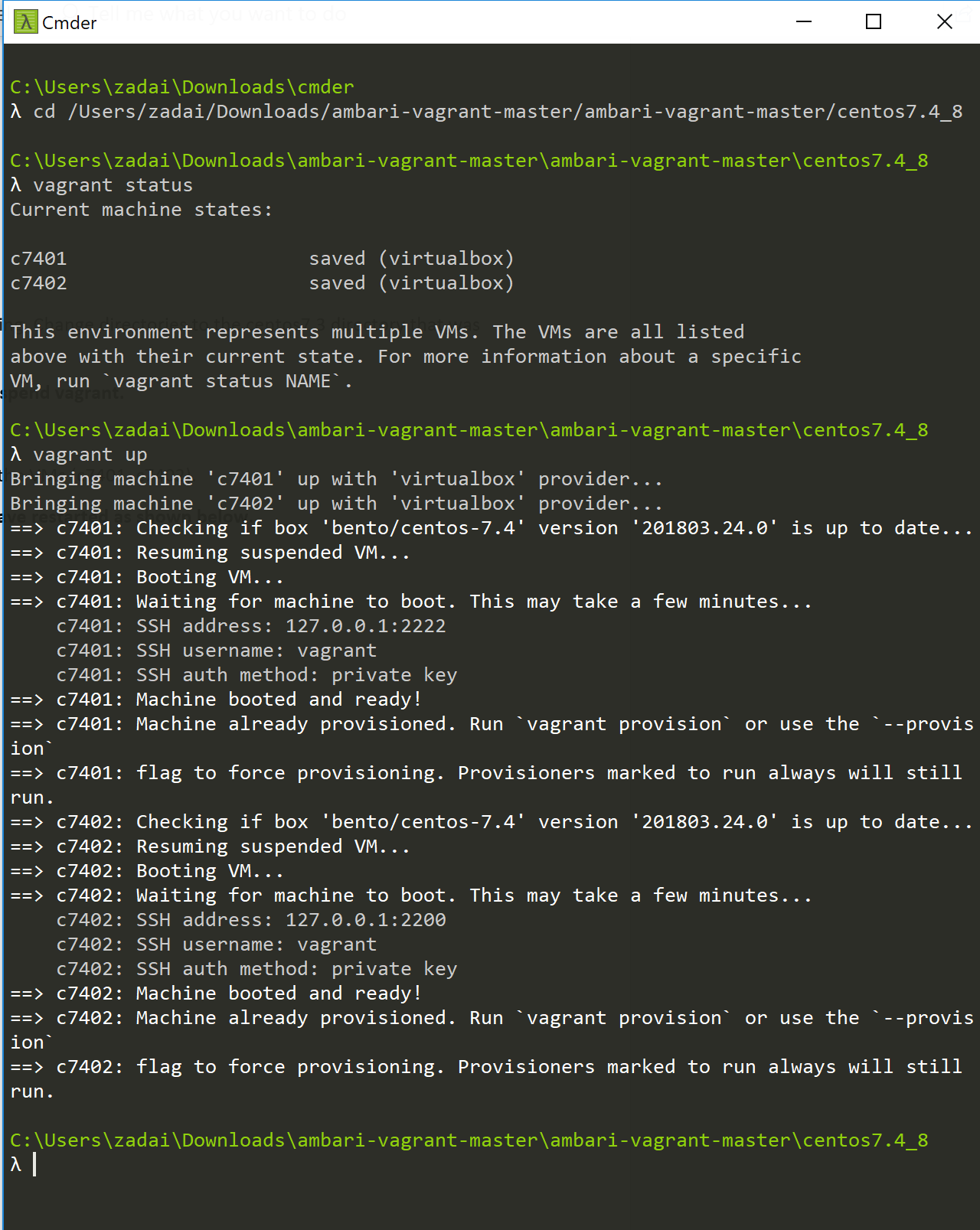
Start Cmder/Terminal/Putty, if it is not running. Change directories to the centos7.3 directory that was created in lab 2.

**Remember at the end of the last lab, we suspend vagrant.**

**Enter the command vagrant up**

This can take a few minutes to restart all of the VMs (c7401, c7402).

Eventually you will see that all of the **VMs have restarted as shown below.**



## Ambari Console

Once the VMs are restarted (booted) and running again, open up a Browser window to check on the status of the VMs using ambari.

**In the Google Chrome Browser (or other browser) address box, enter**

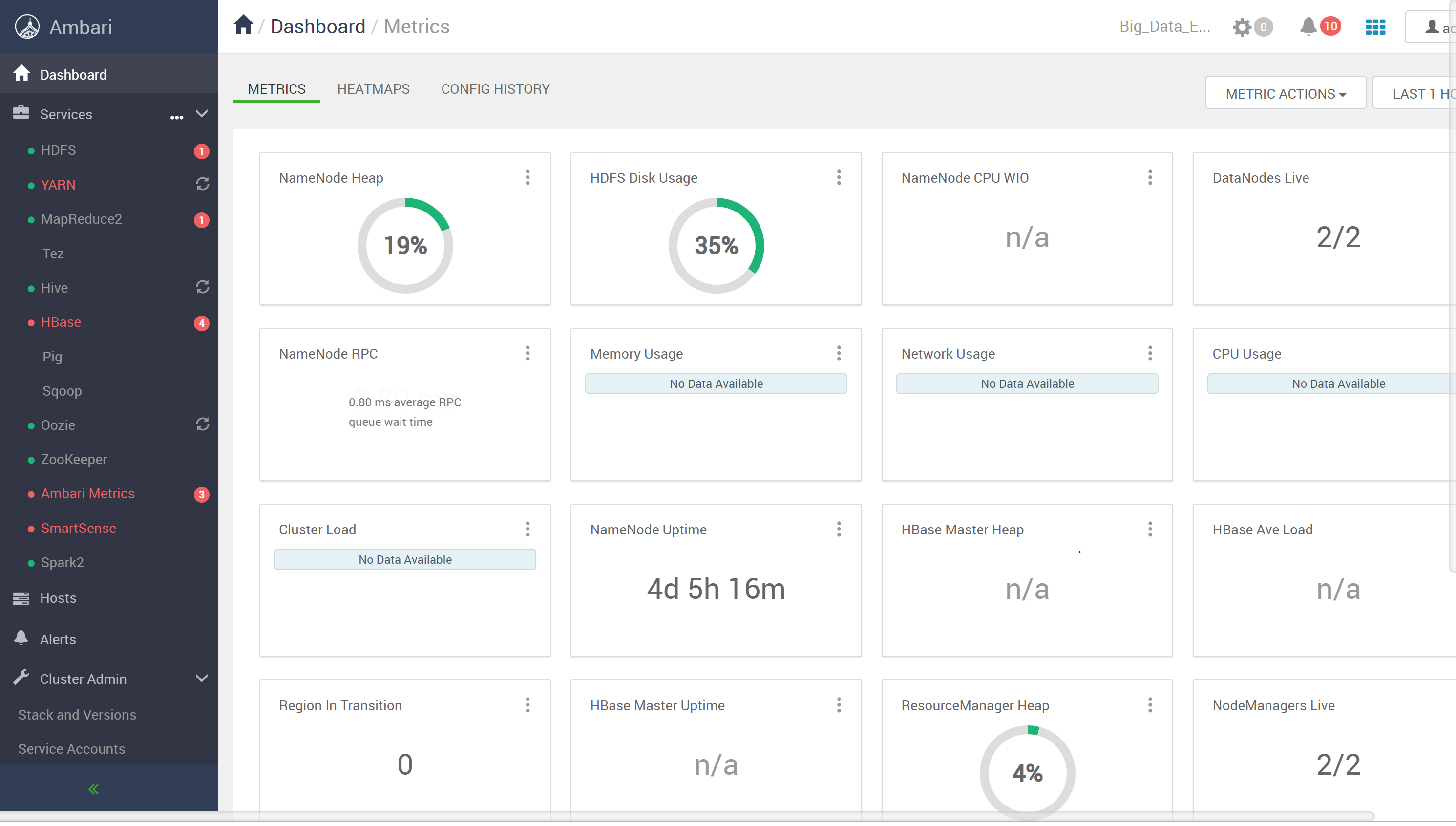
<http://c7401.ambari.apache.org:8080>

**For the ID and password enter admin. Click on the sign in button.**

Username: admin

Password: admin

Once logged in, your window may look similar to the following. Notice that wee have three datanodes that are live (running), provided that your created 3 VMs in the last lab. If you only created 2 datanodes, you will see 2 datanodes running.

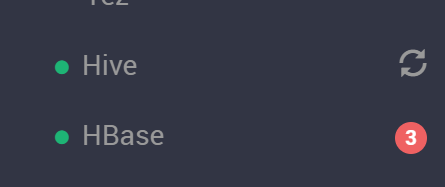


If your dashboard does not show all services up and running, perform the following as we did in lab 2.

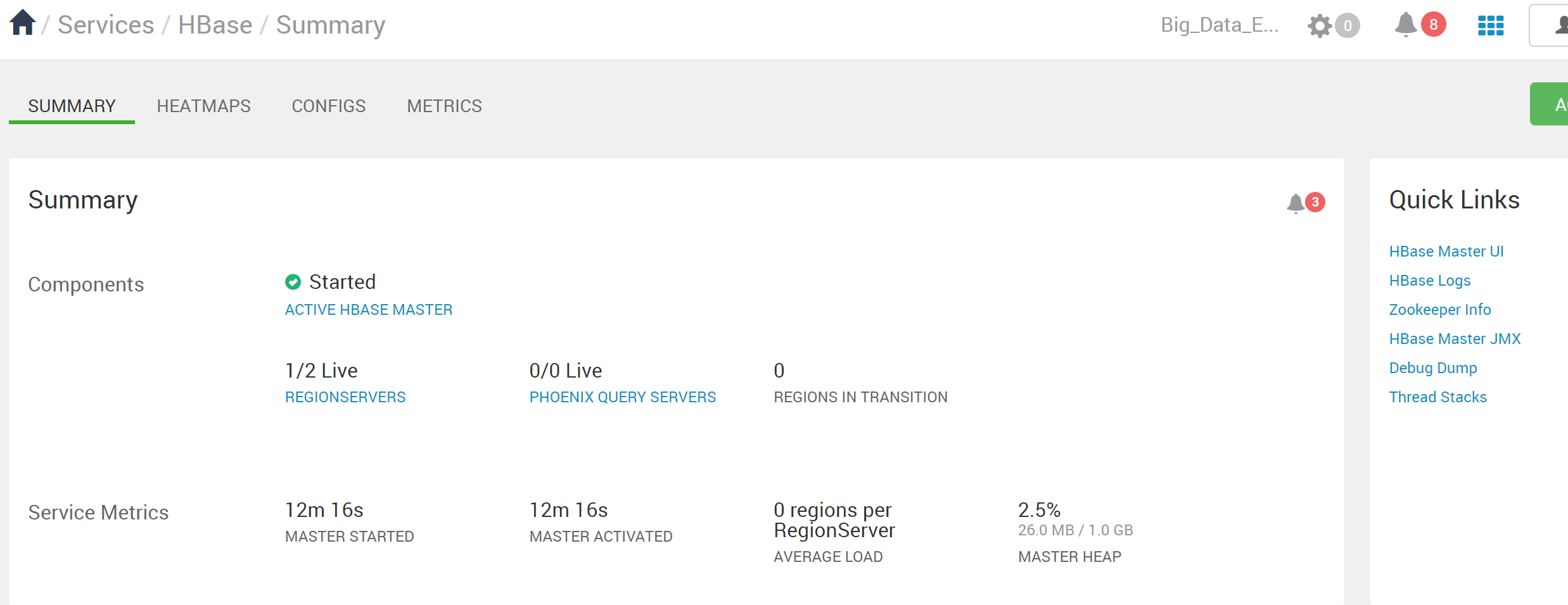


If some services still do not start, you may need to start them manually. I tend to click on Hosts, and then start the services on each hosts manually until everything is up and running successfully.

Click on Services and then HBase on the left side of the screen.



Click on the blue link called Active HBase Master



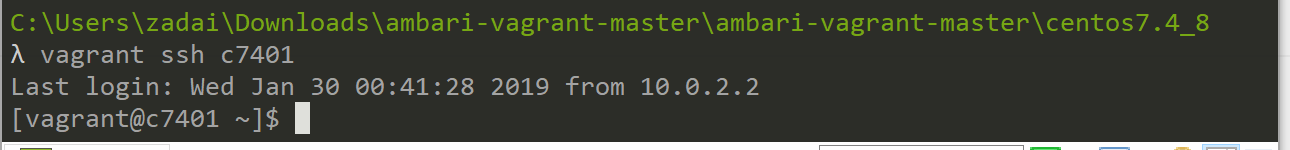
Now that you are satisfied that HBase is active and running. You can use HBase.

# **Part 2**

To connect to the first VM c7401, at the terminal line type:

vagrant ssh c7401

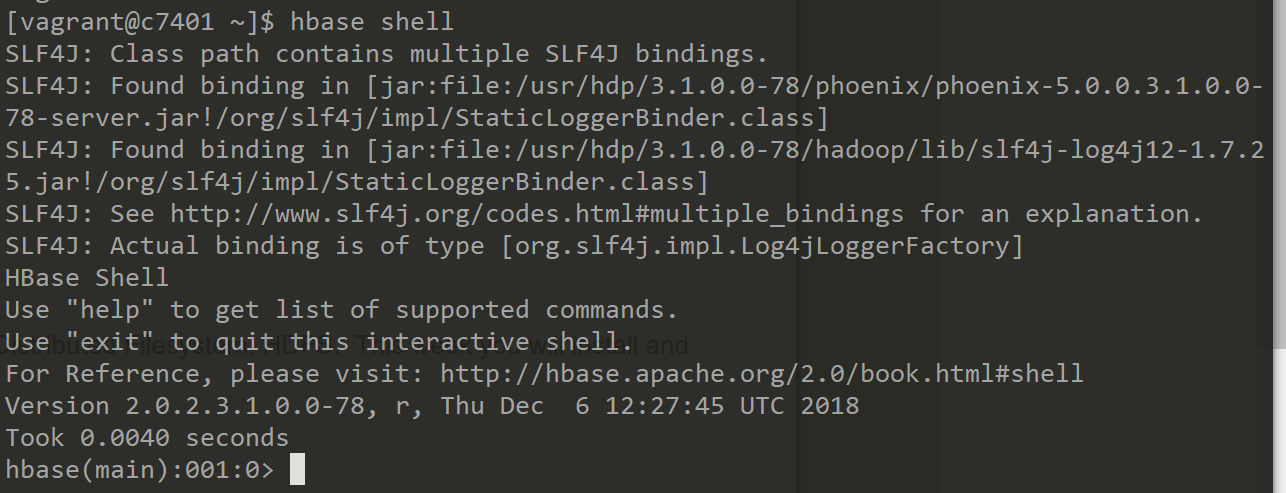
This logs you onto the c7401 VM. (Google ssh if you forgot what ssh is used for)



**From here I had several issues which made it so I could not complete the Lab for this week. So I will detail what I should have gotten as well as what I did get, discussing the errors and the ways I tried to correct those issues. I had issues at two different points. One time was while in HBase and then while backtracking and trying to solve my potential mistake I ran into another issue where I could no longer log into HBase.**

To start the HBase command line interface, type: hbase shell

Note: it may take a few minutes for the shell to start depending upon your PC’s memory.

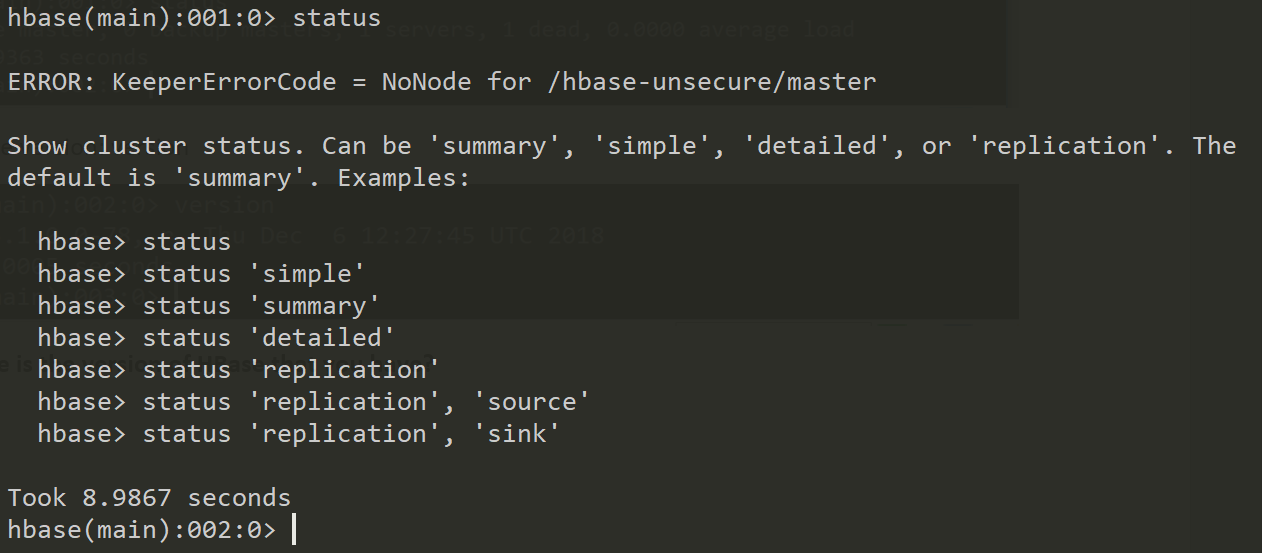


At this point I was able to get into the HBase shell and it looked just like what I should so I moved forward with access into the hbase(main).

**You are ready to enter HBase commands!**

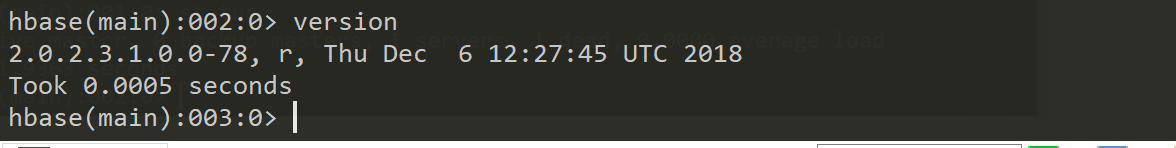
**Note: for future reference, to exit the HBase shell, type exit <return>**

--check the status: status



Checked the status, nothing out of the ordinary here and there were no errors at this time.

--check the version: version



The version was checked and I was able to move into section 2.2 of the Lab.

**What date is the version of HBase that you have?**

*The Date of my HBase is Thursday December 6, 2018.*

## 2.2 Create Tables, Insert Data and Retrieve Data

**Note:** As you issue different commands, the HBase prompt will increment. I didn’t show this in the examples below and I kept it as 000:0 since your prompt values may be different.

**Note:** Using the apostrophe ‘ on your keyboard can be a problem. Below we are using a tick ` instead of the ‘. Therefore, in this case, you will be better off if you copy the statements below and then paste the statements after the HBase prompt using the right mouse button, then press return. If you type the commands and use an apostrophe, the HBase shell will ‘barf’ (technical term) and kick you out of the shell. Then you’ll have to start the HBase shell again to enter the commands again.

**Here is where I ran into my problems running through the lab the first time. The wiki table that I was trying to create was outputting more than expected and I ended up having errors that pertained back to the HBase in my Ambari environment. So from here you’ll witness where I went through noticed the error and than when I decided to stop and back track and try to fix in my Ambari environment.**

-- create a table called wiki

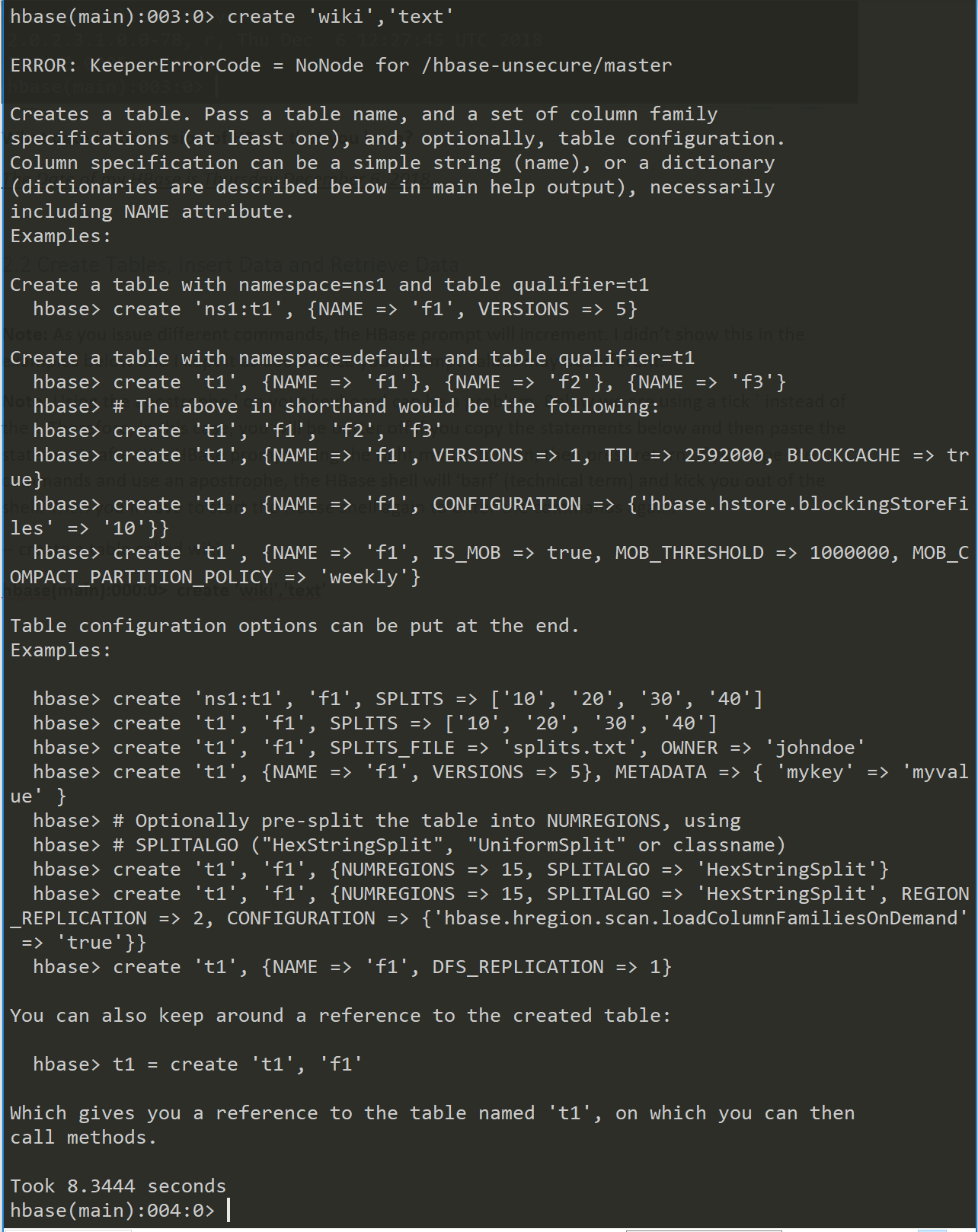
**hbase(main):000:0> create 'wiki','text'**

-- insert a row of data. Home is the (primary unique) key and Welcome is the value.

**hbase(main):000:0> put 'wiki', 'Home', 'text:', 'Welcome to the wiki!'**

-- retrieve data from our new table

**hbase(main):000:0> get 'wiki', 'Home', 'text:'**



**I initially noticed my output from running that get function was longer than I expected but I did try to move through it and see if it was nothing to worry about. Later I noticed that when I went back through I had an error line as the first part of my output stating: ERROR: KeeperErrorCode = NoNode for /hbase-unsecure/master**

**Let’s add more data:**

-- insert more data into a new row of data, Second is the key and second row is the value

**hbase(main):000:0> put 'wiki', 'second', 'text:', 'second row'**

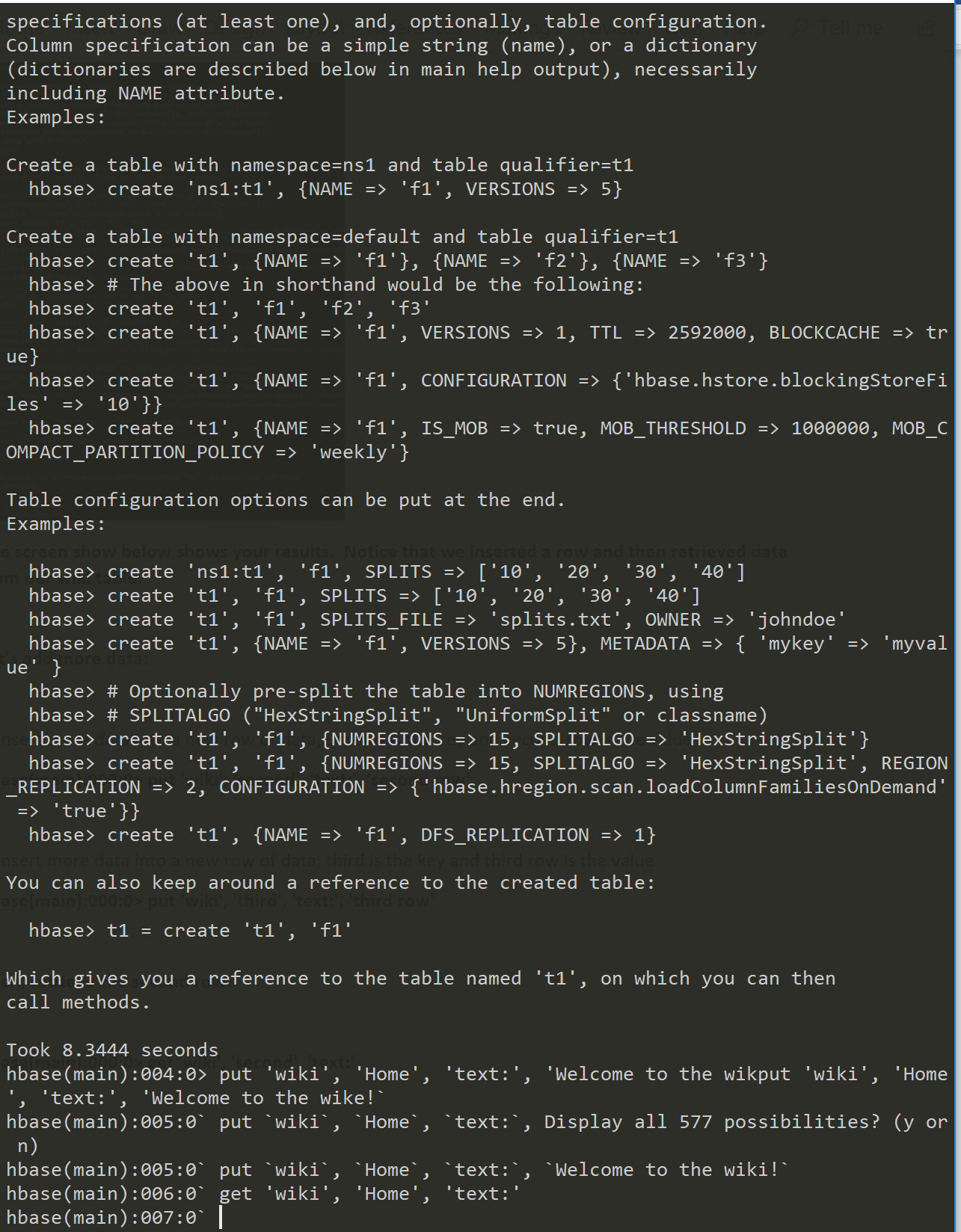
-- insert more data into a new row of data, third is the key and third row is the value

**hbase(main):000:0> put 'wiki', 'third', 'text:', 'third row'**

**Then, retrieve the second row:**

**hbase(main):000:0> get 'wiki', 'second', 'text:'**

**The screen show below shows your results. Notice that we inserted a row and then retrieved data from our wiki table.**



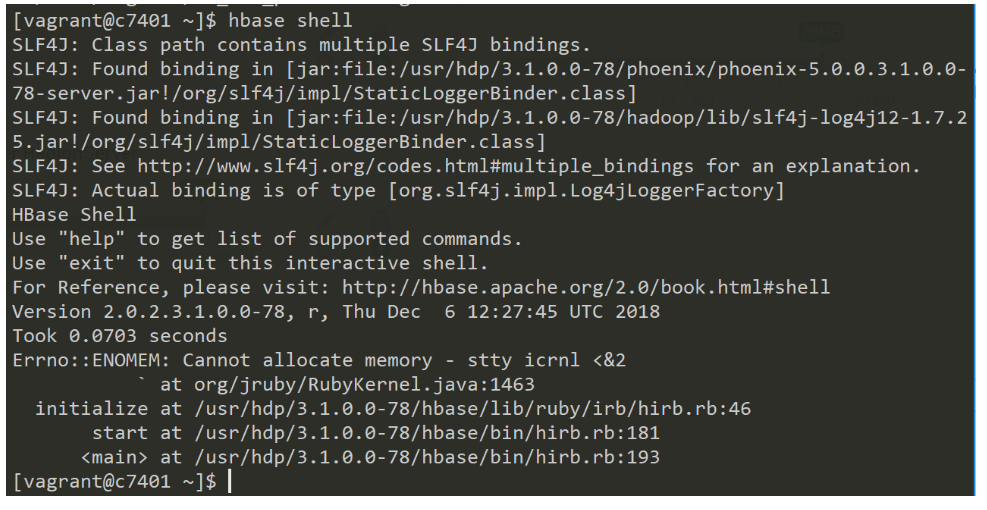
The steps coming up are designed to put data into the wiki table. Above I noticed an issue when my output was not what I expected, and then knew definitively there was a problem when I used the get function and got nothing back. From here I back tracked further an noticed my error from creating my wiki table and how it was tied back to a problem with HBase in Ambari.

My next step was to go into Ambari and try and figure out what was going wrong with HBase.



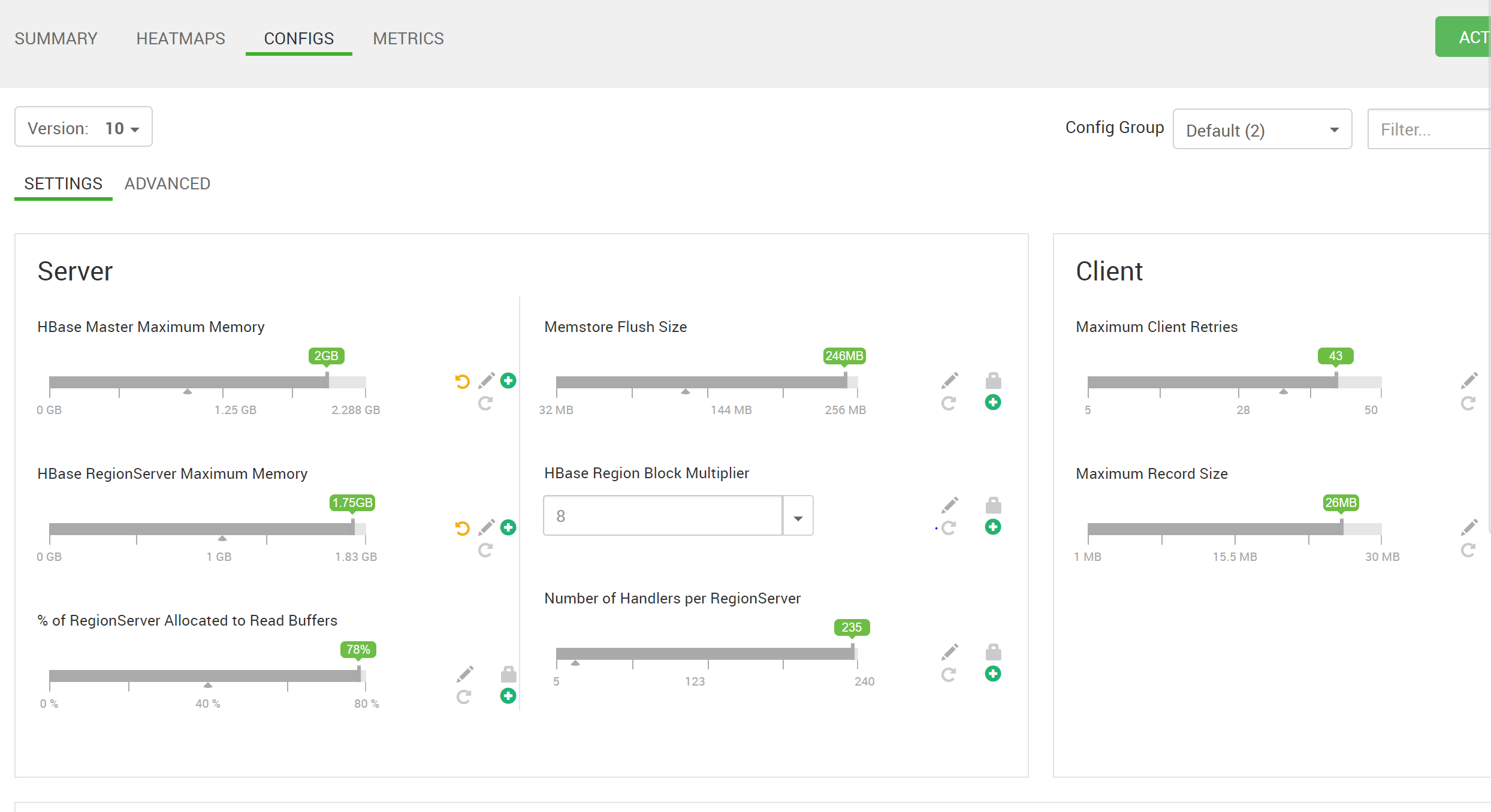
I looked at the summary page and immediately noticed that HBase had a red dot next to it. So my initial reaction was to change that to a green dot by starting all services again. And seeing if that would change it.

First time I did that and it worked so I went back into Cmder and tried to get going again. Before I left Cmder the last time, I made sure to sign out of hbase and end my wiki table to start as fresh and new as possible. But when I tried getting back into HBase I kept getting denied and getting what you see below.



From here I see an issue at the bottom stating: **Erron: : ENOMEM: Cannot allocate memory -stty icrnl <&2**

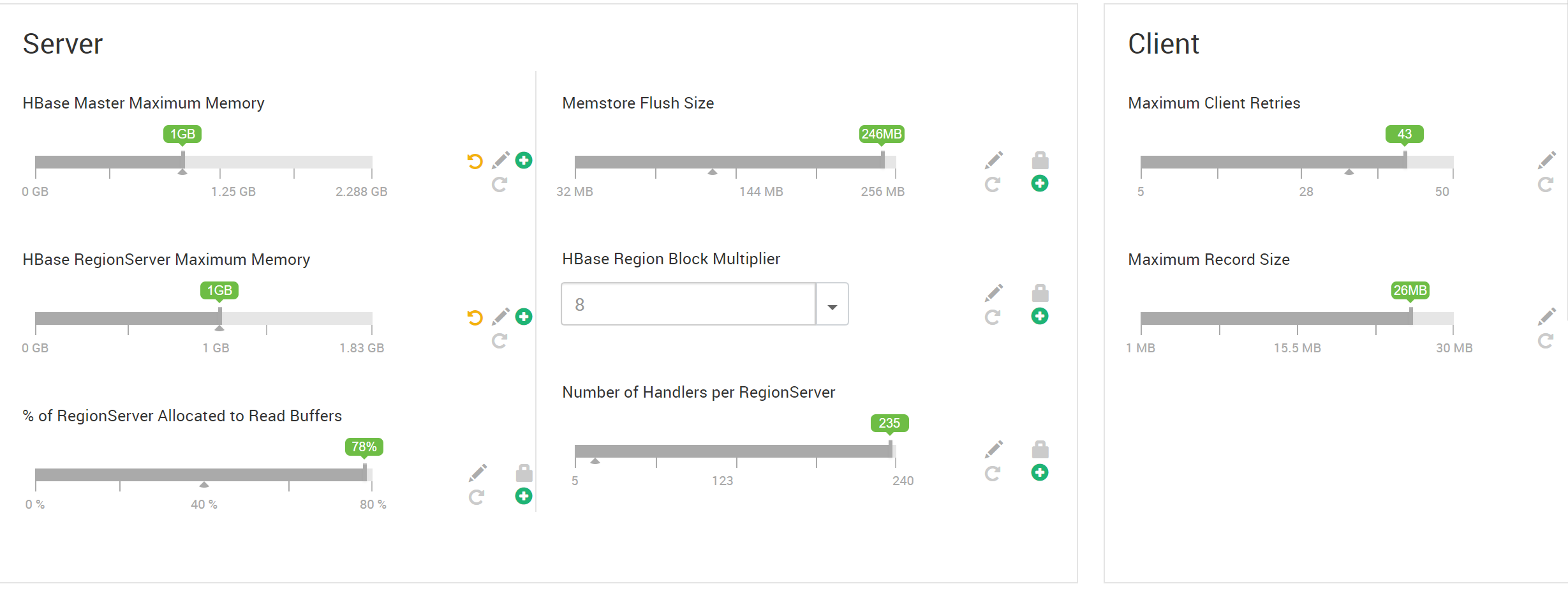
My eyes immediately caught sight of the inability to allocate memory, so I thought I needed to boost my memory for HBase and then restart it and try again to get into hbase shell.

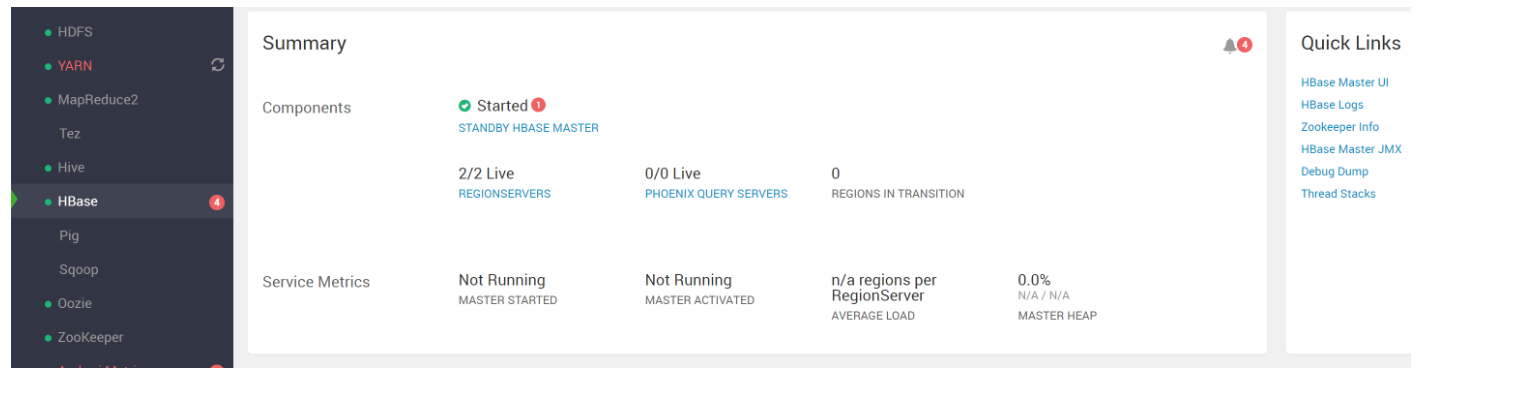


I figured the problem was that I didn’t have my memory config settings maximized for HBase so I did that and restarted HBase. After that I got the red dot to go back to green again and proceeded back to Cmder to try to get back into the hbase shell.

When I tried to get back into the hhbase shell again I got the same error and was not able to get back in. After this frustration I decided to put my issue on our class discussion page in the hopes someone can help me find my error.

After about an hour I got a response from Professor Cathie Wilson and she had told me to check two things. First, check the region servers and make sure they are up and two, to decrease the memory settings on HBase Master Maximum Memory, and the HBase RegionServer Maximum Memory both to 768M, and that should lessen the memory load on my PC.



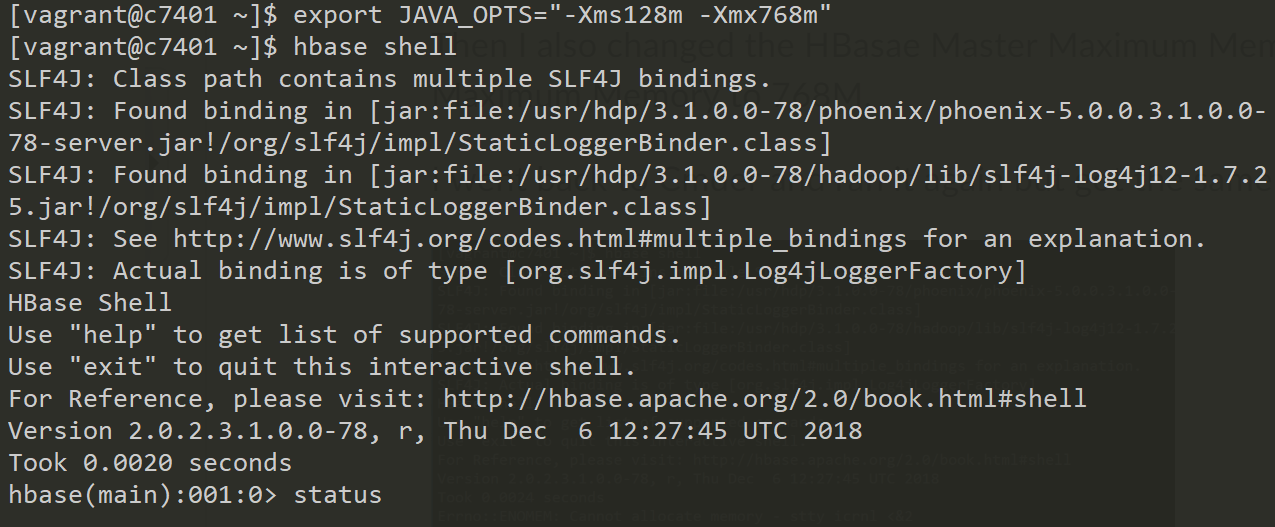


I did both things suggested in my Ambari server, restarted HBase services and went back to try to get into HBase again. But unfortunately I didn’t get the desired result.

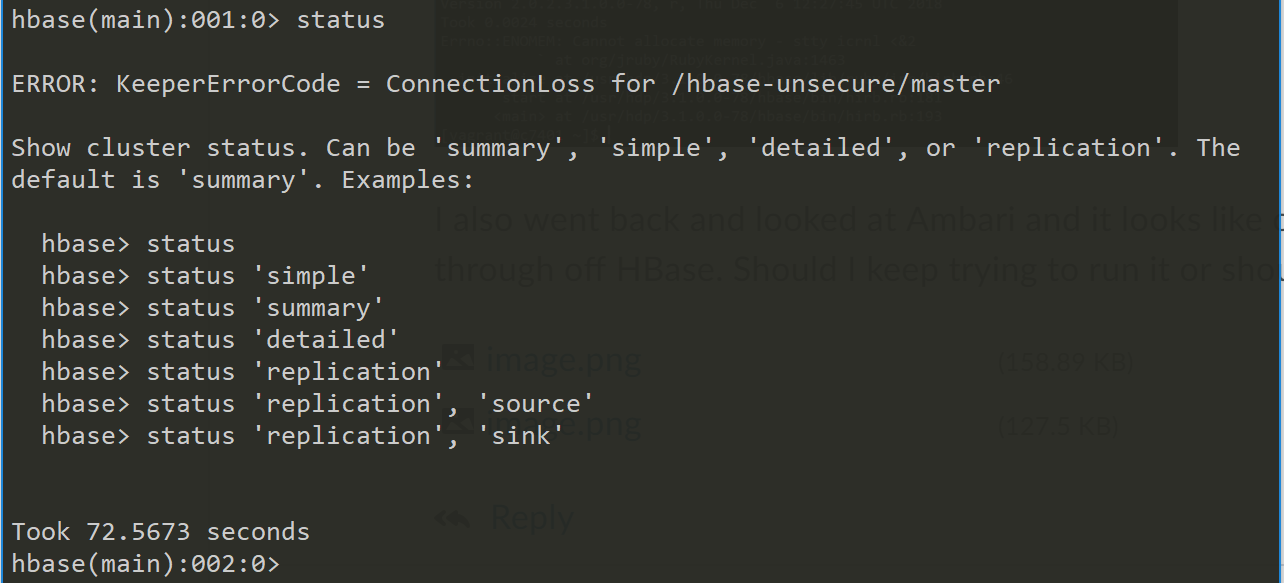
My next option was to get on a call with Professor Cathie Wilson to hopefully solve what my issue was together. I did that Saturday morning. By this time I had shut down my Ambari server and Vagrant machine and needed to bring them back up. Both were uncommonly slow to bring back up so we came up with a plan to hopefully speed up the processing time as well as create a way to work through the rest of the assignment. The plan went like this:

1. Run ambari-server stop
2. Run vagrant halt
3. Reboot my computer
4. When the computer starts back up, only open commander, run through the code instructions after bringing back up vagrant and the ambari server
5. Before going into the hbase shell run the code: export JAVA\_OPTS=”-Xms128m -Xmx768m”
6. Try to get back into hbase shell and continue with the assignment.

From steps 1-4 there was a lot of promise. Shut everything down and brought everything back without issue. I then ran vagrant up commands and brought started the ambari server as well. I got to the point to where I was able to use the command given to me for my Java and I inserted that without issue. So after I went and ran my hbase shell.



So from here I was confident that I had corrected my issue and was able to start building my own wiki table. But after this is where I ran into the same issue I had run into before.



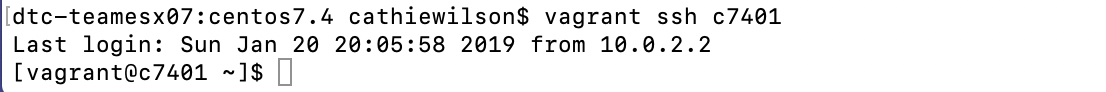
I ran the status of hbase but somehow my connection with hbase had disconnected. So I went into the ambari-server environment to have a look, and sure enough I had the same issues that I had seen before, red dot next to HBase and 0/2 RegionServers were connected. I tried to connect the servers again but it was no use, I was in the same situation I was prior with my errors. So from here on I will run through what to expect starting from when the user is getting into the hbase environment.

## 2.1 Start HBase

To connect to the first VM c7301, at the terminal command line type:

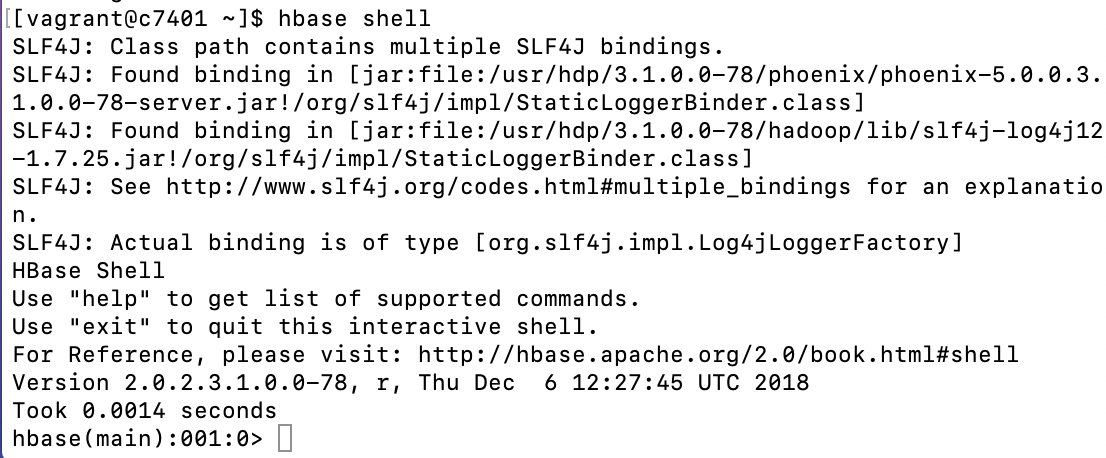
vagrant ssh c7401

This logs you onto the c7401 VM. (*Google ssh if you forgot what ssh is used for)*



To start the Hbase command line interface, type: hbase shell

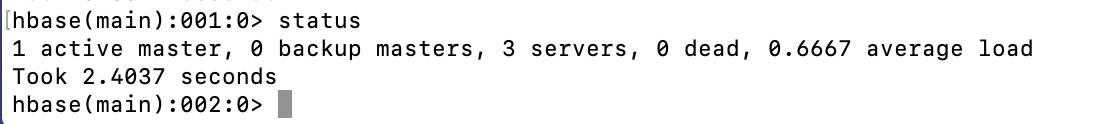
Note: it may take a few minutes for the shell to start depending upon your PC’s memory.



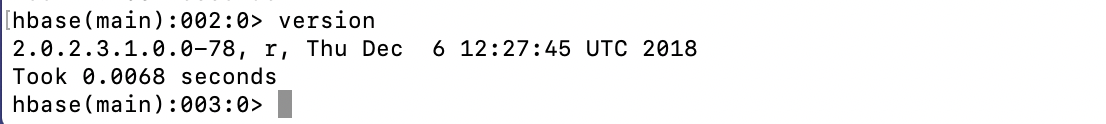
**From here we are into the hbase environment and able to move on from here. Next we will be building the wiki table and inserting the requisite data into it.**

***Note: for future reference, to exit the hbase shell, type exit <return>***

-- check the status: status



-- check the version: version



Answer this question in your lab response: What date is the version of HBase that you have?

**My answer is the same as from the instructions and you can see it above, but my version date is also Thursday December 6, 2018.**

## **2.2 Create Tables, Insert Data and Retrieve Data**

***Note:*** As you issue different commands, the HBase prompt will increment. I didn’t show this in the examples below and I kept it as 000:0 since your prompt values may be different.

**Note:** using the apostrophe ‘ on your keyboard can be a problem. Below we are using a tick ' instead of the ‘. Therefore, in this case, you will be better off if you copy the statements below and then paste the statements after the Hbase prompt using the right mouse button, then press return. If you type the commands and use an apostrophe, the hbase shell will ‘barf’ (technical term) and kick you out of the shell. Then, you’ll have to start the hbase shell again to enter commands again.

-- create a table called wiki

**hbase(main):000:0> create 'wiki','text'**

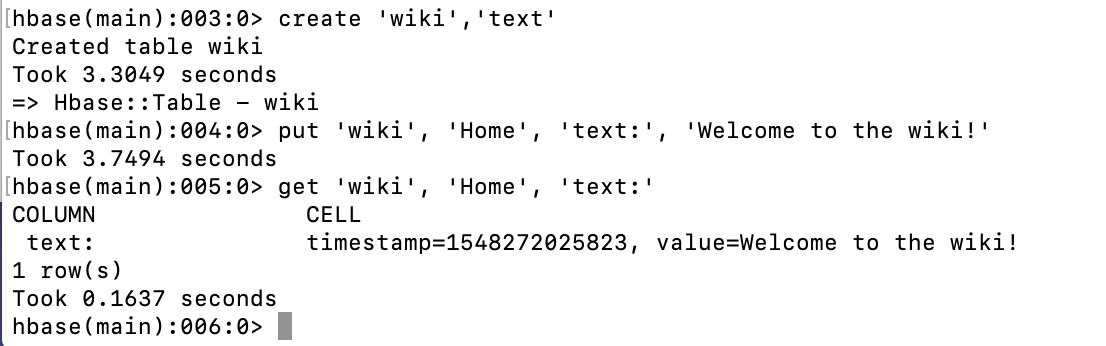
-- insert a row of data. Home is the (primary unique) key and Welcome is the value.

**hbase(main):000:0> put 'wiki', 'Home', 'text:', 'Welcome to the wiki!'**

-- retrieve data from our new table

**hbase(main):000:0> get 'wiki', 'Home', 'text:'**

**The screen show below shows your results. Notice that we inserted a row and then retrieved data from our wiki table.**

****

**We have now created, inserted and grabbed data from the wiki table. The only values in there currently are ‘Home’ which is the primary key and ‘Welcome’ which is the value of the row. So simple table to create and from here there is more data to insert.**

**Let’s add more data:**

-- insert more data into a new row of data, Second is the key and second row is the value

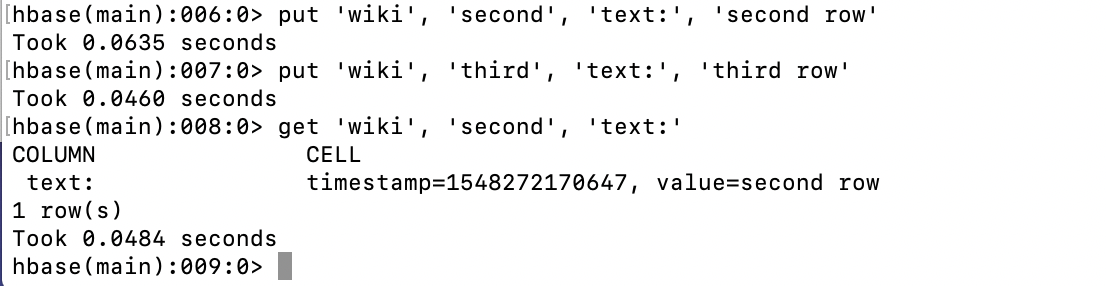
**hbase(main):000:0> put 'wiki', 'second', 'text:', 'second row'**

-- insert more data into a new row of data, third is the key and third row is the value

**hbase(main):000:0> put 'wiki', 'third', 'text:', 'third row'**

**Then, retrieve the second row:**

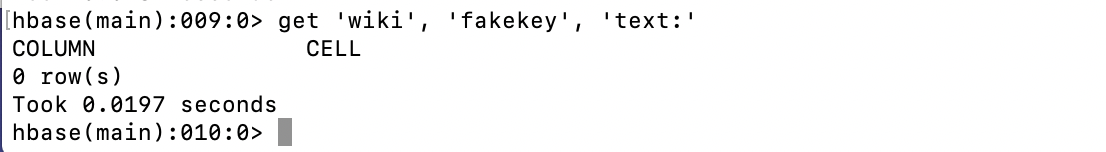
**hbase(main):000:0> get 'wiki', 'second', 'text:'**

****

**In this section. There were two more rows added, ‘second’ and ‘third’ went in as primary keys and ‘second row’ and ‘third row’ are the values. Starting each with the value of ‘wiki’ designates the table the rows are going into.**

-- try retrieving data with a fake key (not valid). You’ll notice that no data is returned.

**hbase(main):000:0> get 'wiki', 'fakekey', 'text:'**

****

**What is of note here is when trying to get data that is not in the wiki table hbase recognizes that and doesn’t retrieve any data. So it was able to recognize that it was told a fake key and thus didn’t grab any of the information.**

## **2.3 Lab Questions**

**Answer these questions in your lab results:**

How many rows were retrieved for the fakekey?

**There is zero because it is a fake key.**

Besides the value for the row, what else is retrieved from HBase?

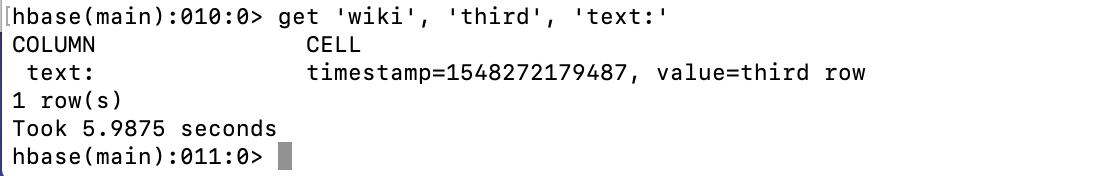
**When using the get command, you get a timestamp.**

How might you use a timestamp?

**The timestamp can be used to inform when the data value was created and to log when a table has its data inputted and seen by users.**

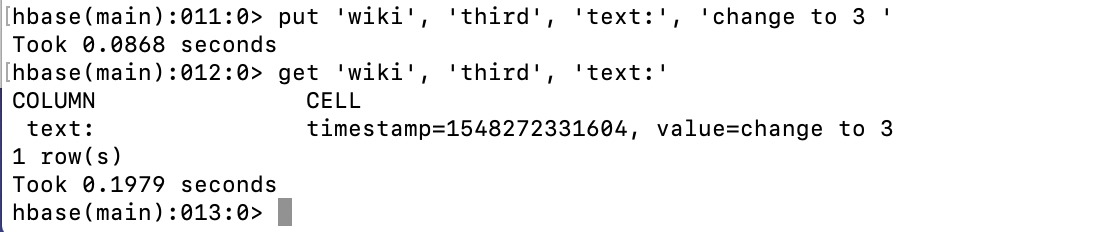
**Now retrieve the third row, put a new value (update the text: value) in the row and retrieve it again as shown below.**

**hbase(main):000:0> get 'wiki', 'third', 'text:'**

****

**hbase(main):000:0> put 'wiki', 'third', 'text:', 'change to 3 '**

**hbase(main):000:0> get 'wiki', 'third', 'text:'**

****

**The third row has now been changed in the wiki table with the primary key of ‘third’ and a value changed to ‘change to 3’.**

Answer these questions in the lab response:

**Why did the value change? Is the timestamp different?**

**The value changed from ‘third row’ to ‘change to 3’ because we used the same primary key and we used a put command to put ‘change to 3’ as its value. Then the timestamp is a little different but that is because the data is new so the timestamp has changed with the value.**

-- Issue the following command to create a ‘students’ table with two column families

**hbase(main):000:0> create 'students','account','address'**

-- add data to the table using multiple values for street, zipcode, state

**hbase(main):000:0> put 'students','student1','account:name','Alice'**

**hbase(main):000:0> put 'students','student1','address:street','123 Ballmer Av'**

**hbase(main):000:0> put 'students','student1','address:zipcode','12345'**

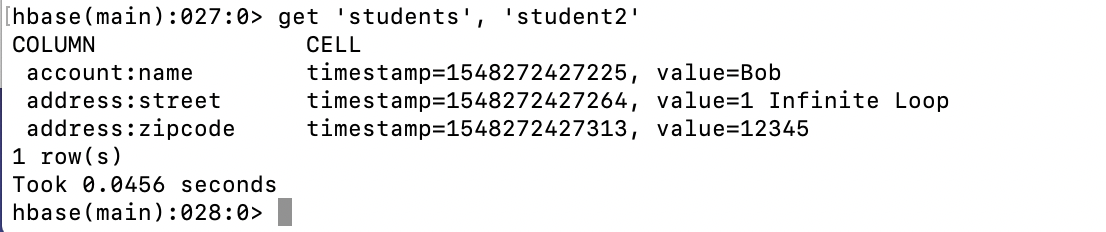
**hbase(main):000:0> put 'students','student1','address:state','CA'**

**hbase(main):000:0> put 'students','student2','account:name','Bob'**

**hbase(main):000:0> put 'students','student2','address:street','1 Infinite Loop'**

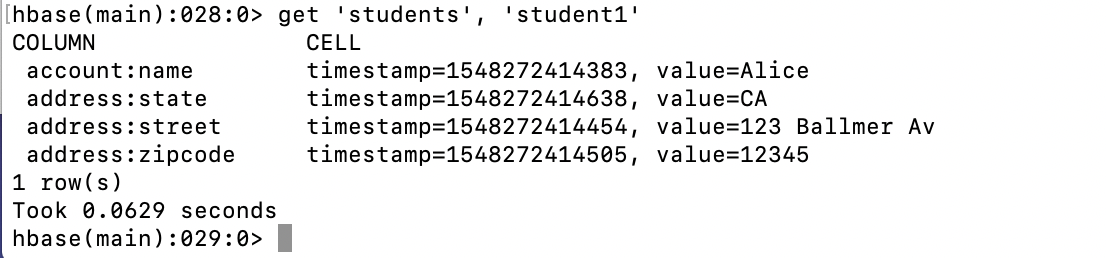
**hbase(main):000:0> put 'students','student2','address:zipcode','12345'**

Retrieve the data the values for using the get command as shown below in the screen shot. **get 'students', 'student2'**

****

**Now…**

**get 'students', 'student1'**

****

**Three new tables were created, students, accounts and addresses. The new data for each was input each record one at a time, and by the end we had two students and had their addresses with separate entries to place everything in their proper fields per table.**

Answer the questions below in your lab response:

**Based on the results from student1 and student2, what does it mean when NoSQL databases are described as ‘sparsely populated’?**

**It means that in NoSQL databases that data records can be either fully populated or they can be filled with nulls and it can vary based on the record and how much data has been collected for the specific unique record.**

**How is sparse data shown in the examples above?  
It is shown above between student1 and student2. Student1 had all the entries filled in that we’re assigned but student2 didn’t have state populated for it leaving it as a null value for that student.**

**Why did the one get statement return all of the values for one key which is different from the get wiki (above)?**

**The reason it returned all those values is because all those values made up the entire record for student1.**

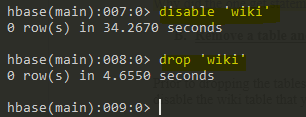
## **2.4 Remove a table and shutdown HBase**

Prior to dropping the tables, we need to disable them. Use the following commands to disable the wiki table that you created. We will keep the students table for later use.

**hbase(main):000:0> disable 'wiki'**

Then drop the ‘wiki’ table

**hbase(main):000:0> drop 'wiki'**



**This is how to stop and close out of your wiki table. From here the next steps will be to exit hbase shell and close out of Cmder.**

The last command basically wraps up lab 3. Exit from the HBase Shell using exit.

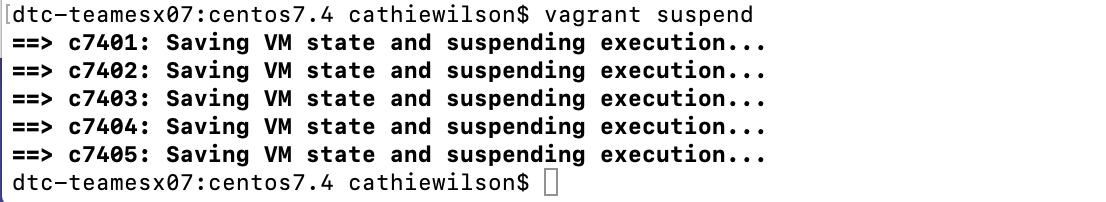
**hbase(main):000:0> exit**



Exit now from the ssh (shell that you’ve been using) on the c7301 VM to return to the terminal prompt (as shown below).



Enter the command: vagrant suspend



**Reflection:**

Even though I was not able to complete the HBase lab it was very informative to see, from a surface level, the capabilities of HBase. Even with the small tables created in this lab, it is apparent how well HBase can handle unstructured data and that it has capabilities that relational databases don’t have, proving their use and why they’re really starting to take over the Big Data landscape.