BIG DATA 210
Week 2 Assignment: Hadoop, Hive and More
Autumn 2018

To save time and make things run smoothly, we've provided each student with an Azure VM running Linux, that has Docker and the Hortonworks Sandbox pre-installed. We'll continue using this VM throughout the course.

Getting Started with your Azure VM

To get started with your Azure VM, perform the following steps which we demonstrated in class:

- Visit https://portal.azure.com and log in with your UW NetID
- Find the VM for your UW NetID in the "Virtual Machines" section
- Start your Azure VM and wait for the portal to confirm that it has booted up
- Set up your SSH login credentials as shown in class

The Hortonworks Sandbox

The Hortonworks Sandbox should start automatically when your Azure VM boots. Note that the sandbox in the Docker container can take between 10 and 20 minutes to finish starting all of the Hadoop services. You can verify that the sandbox's Docker container has started by logging in to your VM's bash shell and running:

```
sudo docker ps
```

Check that the output of the above command shows a pair of running containers with names like sandbox-proxy and sandbox-hdp.

If either container is missing, you can tell Docker to start them by invoking as necessary:

```
sudo docker start sandbox-hdp

sudo docker start sandbox-proxy
```

Once you're sure the containers are running, move on to the next section and check in on Ambari.

Getting Started with Ambari

Go to http://IP.ADD.RE.SS:8080 to see the Ambari Web UI. Replace “IP.ADD.RE.SS” with whatever the public IP address of your Azure VM is.

Once the sandbox is far enough along in its startup you’ll see the following in your browser. The

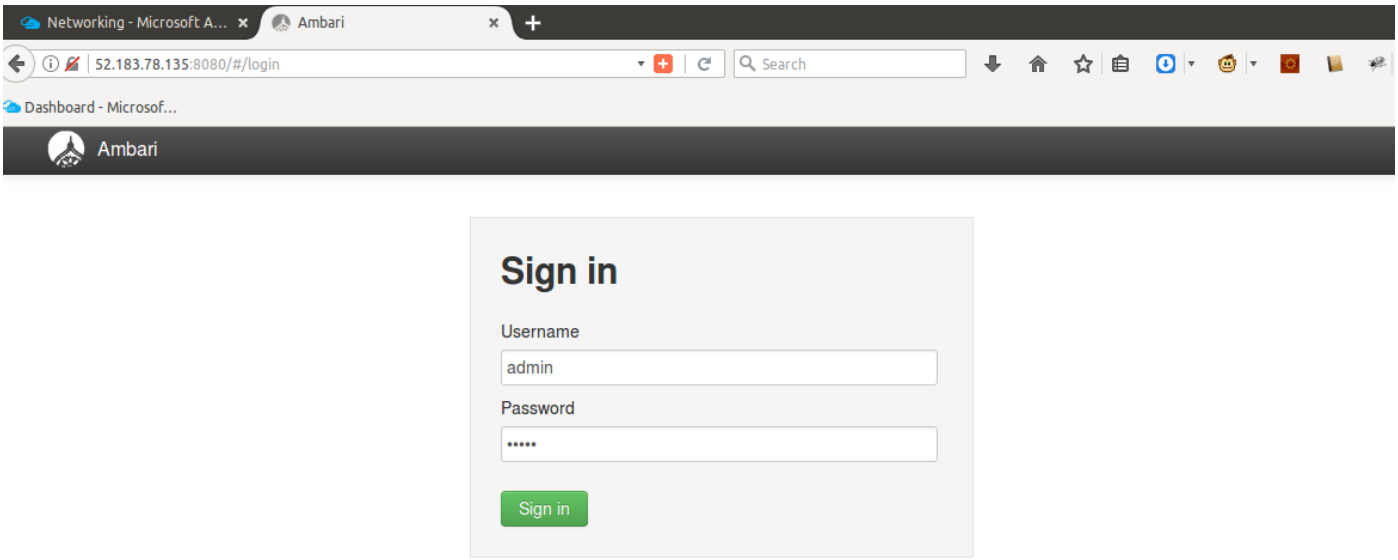

Figure: The Ambari Login Page

Ambari web UI comes up pretty early in the sequence so you can use it to monitor the rest of the startup.

Log in as “admin” using password ”hadoop” (without quotes). Please change the admin password as soon as you log in by using the “admin” dropdown on the right side of the topmost Ambari toolbar. In the dropdown menu, select “Manage Ambari” and on the next page that appears go to “User + Group Management” and click on “Users”. You will then be presented a list of users that includes “admin”. Click on “admin” and change its password.

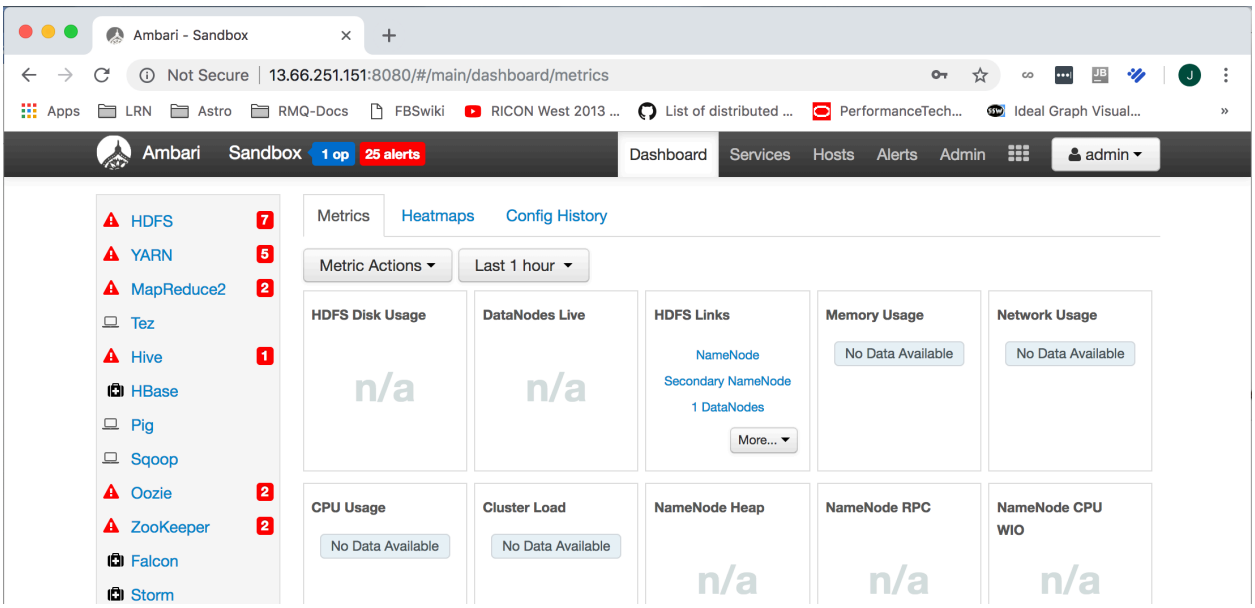

Figure: The Ambari Main Dashboard with "ops" and "alerts" at top

On the toolbar at the top of the Ambari web UI you'll notice a couple of badges for "ops" and "alerts" on the upper left next to the Ambari logo and the Sandbox label. If you click on "ops" you can see the currently running and recently finished background operations. If your sandbox is still starting up you'll see that "Start All Services" is incomplete. You can drill down using the arrow next to the associated progress bar to see how far along the startup process is and where in the sequence we are.

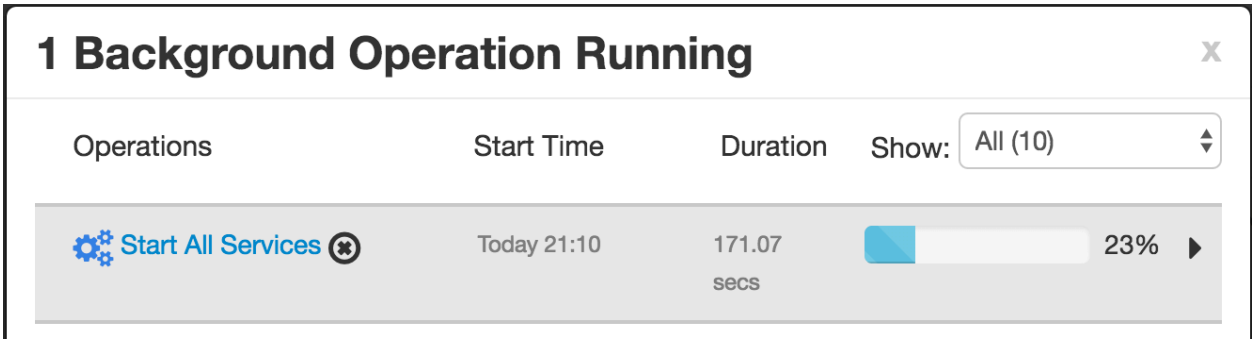

Figure: The Background Operations Pane from Pressing the "ops" badge, click the arrow for more details

Fun With Hive Lab

Now, we'll get started with Hive.

We'll need to bring up the Hive view and create a table. Only instead of loading from HDFS you will simply upload from a CSV file from your local machine for this assignment. Please feel free

to explore HDFS and get a handle on the Ambari HDFS “Files View” and the “hadoop fs” command line tool as shown in class.

Download the sample data from Canvas to your local machine. The Canvas URL of the file is:

https://canvas.uw.edu/courses/1243280/modules/items/8839033

In Ambari, bring up the “Hive View 2.0” from the 3x3 grid icon dropdown near the top right of the Ambari web UI’s top toolbar.

Create a new Hive table using the green “+ NEW TABLE” button. Press the green “UPLOAD TABLE” button and provide the file you just downloaded (make sure the box to use “first row as header” is checked---Hive will use this to try to infer column names). A green “Create” button may lie off screen to the bottom. Scroll down to find it and press it.

While things churn along you’ll see a dialog that mentions “insertion of rows from temporary table” and the “deletion of temporary table”. Eventually these will go away and you’ll have a Hive table based on the CSV text file you selected.

Now switch to the query view. Run the following query as a sanity check:

```
select count(*) from home_data
```

You should get a result of **21613** if everything is working.

Assignment 2: Exercises to Submit

Now that we have the VM and the sandbox working, and some data imported into Hive, it’s time to do some Hive SQL finger exercises.

Use the sample data provided to answer the following questions. Please submit your answers to the following questions along with the query you executed to get that answer. For question 5 save the results file using the “Save as” option from the Ambari UI and attach to the assignment submission.

1) What are the most and least expensive houses in the data set?
2) What is the most expensive zipcode in the data set, defined as highest average sales price?
3) How many houses were built prior to 1979?

Using the same process as before, download the CSV file from the following Canvas link and create a Hive table from it:

https://canvas.uw.edu/courses/1243280/modules/items/8840048

You've imported a zip code lookup table for WA, King County. Use the zipcode table table to join with the previous table to solve the next questions:

4) How many homes were sold with a zipcode defined as being in "Seattle"?
5) Output a report showing the most expensive house in each city. Include at a minimum the price, zipcode and city. NOTE: You might find this question's query interesting to run in the *beeline* command line Hive shell so you can see an example of a query that compiles down into a non-trivial MapReduce job that has two map stages and two reduce stages.

Bonus Exercise 1:

Price per sq. ft. is a common metric for valuing housing. Calculate the avg. price per sq. ft for the houses sold in this data set. You do not need to get the answer using one query i.e. can use two or more queries to get the data you need to solve.

Bonus Exercise 2:

"date" is a reserved keyword in Hive. The sample data has a column named "date". How can you use this column in a query without an error? Generate a list of all the unique date values represented in this data set.