Week 1 Updated Assignment

Introduction

Week 1 introduces you to some basic data engineering ecosystem tools. It also introduces Hadoop and shows you how to execute a simple program in the framework utilizing a mapper and a reducer file. If you don't have any experience in Linux, please review the Linux Primer or watch this video [https://www.youtube.com/watch?v=rvCU1E97pto] .
This week will also see you working with Git, GCP, and Hadoop. Docker is optional but very helpful in completing the assignment.
Please note there is a technical portion AND a written portion. Maximum of two pages for the written portion, and I do accept bullet points. If you'd like to format everything and make it presentable I'd recommend including it on your personal github for your candidate profile for job applications.

Assignment

Written Portion

Provide a report of up to 2 pages addressing the following points. The reports each week should include citations if needed; you can use APA format or any other clean-looking format -- it's recommended to use Zotero to export citations if citations are needed.
What is Linux, and why is it important for data engineers and data scientists?
Who are the main cloud services providers, and what sort of things do they provide?
Here is something to get you started: Google Cloud [https://cloud.google.com/free/docs/map-aws-google-cloud-platform]
What can we do with the cloud providers related to data engineering?
What is Docker, how is it useful, and why is it important to know?
This article may help [https://medium.freecodecamp.org/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b]

Technical portion

The assignment is to run the wordcount example. Please include screenshots that show your username, and if you do the

assignment locally please include some steps of how you set up your environment.
Please review the following document for some guidance. Week 1 lab supplemental rtm [../facdocs/Course%20Resources/Week1-lab-supplemental-RTM.docx?_&d2lSessionVal=ydyTWLjuCdSyKDrOPoky0uytl&ou=287325]
We will be using cloud resources during the course. Anytime you are using cloud resources, there is potential to incur costs. We have $50 in credit from GCP (Google Cloud Platform), and you have $75 from AWS (Amazon Web Services). You should also be able to get $300 of credit by signing up for GCP with a new account. Always be sure to shut down everything when you are done with a project -- double-check that everything is shut down every time. Leaving machines on can be an expensive mistake.

Technical Lab

Set up an account with GCP / Set up your own linux machine

Create an account for Google Cloud [https://www.youtube.com/watch?v=2BWx_lsleOw&feature=youtu.be] or use an existing Google account. Create a project by clicking on the upper left next to the Google Cloud Platform text. Then click the menu button in the upper left, then go to billing. You should also have $300 in credit if you just created your Google Cloud account. The $300 in credits are good for most things, but have limitations like no GPUs. The $50 credit may or may not work for GPUs. Unfortunately, you will also have to add a credit card or payment method in order to use the free credits.
Next, **create a budget** by choosing the 'Budgets & alerts' option on the left. Set some of the threshold rules at something like 25% and 50% (or even less) of the $50 value, so you will get an alert if you are using your credits too fast. Most of the resources we use will be pretty cheap (usually less than a few dollars if you use the resources for half a day), so $50 should be more than enough to complete the course and even experiment more on your own.
You can also create your own environment if you would like to.

Set up Google Cloud Products (GCP)

Once you've redeemed your credit and set up billing, we can get started. This assignment should only take a few dollars (or less) of the credits.
Create a project if you haven't already, then spin up a small compute engine with 2 vCPUs, 8GB memory, a 50GB HDD, and an Ubuntu 18.04 OS. See this video [https://youtu.be/2BWx_lsleOw] for an overview of how to get set up with GCP and get your first VM running for the assignment.

Hadoop Docker quickstart

Hadoop with quickstart Docker [https://www.youtube.com/watch?v=s9MWTjtnnvs&feature=youtu.be] video on how to run Hadoop with the quickstart docker Hadoop container.
If you don't want to burn any GCP credits, you can run this locally by installing docker on your computer (see below for "optional local version").
Use this Docker Container [https://hub.docker.com/r/ngeorge/ubuntu-hadoop-quickstart] to be able to run Hadoop easily.

First, we need to install docker. Ideally, this is completed by following the instructions Install Docker [https://www.linux.com/tutorials/how-install-and-use-docker-linux/] . However, by far the easiest way to install is `sudo snap install docker`, which uses the [Snappy package manager] Snappy Manager [https://en.wikipedia.org/wiki/Snappy]
You can also do `sudo apt-get install docker.io`, but this installs an older docker version.

Install with apt

The essential commands for installing docker in Ubuntu with `apt` are:

```
sudo apt-get update
```

sudo apt-get install \
apt-transport-https \
ca-certificates \
curl \
gnupg-agent \
software-properties-common

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) \ stable"

```
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Running docker

Next, add your user to the docker group: `sudo usermod -a -G docker $USER`

If you installed docker with snap, you will get an error that the docker group does not exist. You will need to first create the group:

```
sudo groupadd docker
```

Then log out and back in again by closing the SSH window or typing `logout`, then re-opening the SSH tunnel. If logging out and back in doesn't work, restart the machine with `sudo reboot` from the command line.

Now pull (download) the docker container for the cloudera quickstart:

```
docker pull ngeorge/ubuntu-hadoop-quickstart
```

Optionally run it within the tmux `hadoop` container:

```
tmux new -s hadoop
```

('ctrl + b', then 'd' will exit the tmux hadoop shell)

```
tmux attach -t hadoop
```

Run the docker container as interactive (-it option):

```
docker run -it ngeorge/ubuntu-hadoop-quickstart
```

`ctrl + d` will exit the docker container (stopping it).

Optional local docker version

If you have fast internet and a good computer (8-16 GB RAM and a decently fast processor), you can run it locally for extra practice. If your computer has struggled to run VMs in the past, you should stick with GCP.

Install docker CE (community edition, free, not EE, or enterprise edition, which is not free). Use docker toolbox, if your OS is too old. If you are using docker toolbox, your computer may be too slow to run this assignment locally.

Use this docker container to run hadoop (takes a while to download). The steps are (from a terminal, command prompt, or docker quickstart terminal):

download:

```
docker pull docker pull ngeorge/ubuntu-hadoop-quickstart
```

and run it:

```
docker run -it ngeorge/ubuntu-hadoop-quickstart
```

Run wordcount example

The "Word Count" example with Hadoop is the analog of the "Hello World" example used as an intro to most programming languages. One of its uses is to demonstrate that you have a working Hadoop installation.

First, get the code to run the wordcount problem from GitHub. The repository is HERE [https://github.com/Regis-University-Data-Science/simple_Hadoop_MapReduce_example]

Download it with `git clone https://github.com/Regis-University-Data-Science/simple_Hadoop_MapReduce_example.git`.

```
cd simple_Hadoop_MapReduce_example
```

The text data that we will use for this test is the works of Shakespeare. Use the `wget` command to get the data from the internet.

```
wget http://norvig.com/ngrams/shakespeare.txt
```

Next, create directories in HDFS for the data. This is done with the hadoop `fs` command.

The first command creates a directory, called shakespeare, in the home directory (shown below 4).

The second command creates a directory, called input, in the shakespeare directory.

The third command copies the data from the local file system to HDFS.

The fourth command lists the content of the HDFS directory `/shakespeare/input`

```
$ hdfs dfs -mkdir /shakespeare
$ hdfs dfs -mkdir /shakespeare/input
$ hdfs dfs -copyFromLocal shakespeare.txt /shakespeare/input
$ hdfs dfs -ls /shakespeare/input
```

hdfs dfs -mkdir /shakespeare hdfs dfs -mkdir /shakespeare/input hdfs dfs -copyFromLocal shakespeare.txt /shakespeare/input hdfs dfs -ls /shakespeare/input

An alternative (perhaps older) way to run `hdfs dfs` is `hadoop fs`.

NOTE: The slashes below indicate line continuation characters. Basically this is one command on multiple lines of parameters.

```
mapred streaming \
-mapper mapper.py \
-reducer reducer.py \
-input /shakespeare/input \
-output /shakespeare/output
```

The mapreduce streaming command used to be different as well; it has changed over time. An older version would look like:

```
hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
-files mapper.py,reducer.py \
-mapper mapper.py \
-reducer reducer.py \
-input /shakespeare/input \
-output /shakespeare/output
```

Check results with:

```
hdfs dfs -ls /shakespeare/output
hdfs dfs -cat /shakespeare/output/part-00000
```

and copy the results file to the local disk with:

```
hdfs dfs -copyToLocal /shakespeare/output/part-00000 result
```

View the sorted word count (e.g. to get top words) like so:

```
sort -gr -k 2 result | head
```

`g` and `r` are separate options, and the command could also be written:

```
sort -g -r -k 2 result | head
```

See the manual for more info on the sort and head commands. These manual pages are also available by typing `man sort` in the console/terminal.

Hadoop was originally made to run with Java, and can be run with Java JAR files like so:

```
hadoop jar /home/hadoop/hadoop-2.9.2/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.9.2.jar wordcount
/shakespeare/input /shakespeare/output
```

NOTE: If you run this word count more then one time, change the name of the output to output2, output3, etc. , otherwise you will get an error on the output directory name

Bonuses (optional - total bonus up to 10 points)

You can create a separate write up describing your results for the bonuses, and submit it to the same dropbox for week 1, or include the bonuses in a single write up with the assignment.

Download another text (perhaps from gutenberg.org, e.g., maybe the Bible) and use Hadoop to perform a wordcount. What are the top words from the count? What are the least-occuring words from the count (hint: use `tail` instead of `head`, or don't reverse the `sort`)?

Boot a VM directly using the docker image. This involves an extra setting when setting up the GCP VM.
Practice sharing files between docker and the host machine. Download some texts on to the host machine, then transfer them in to HDFS, and run MapReduce on them to get wordcounts.
Create your own docker container and upload it to docker's cloud hosting.
Read about why we still use terminals [https://ux.stackexchange.com/questions/101990/why-are-terminal-consoles-still-used] and not GUIs everywhere. Summarize your understanding in your writeup.

Optional Information

Sharing data between docker and the host filesystem
If you want to share files between docker and host system, the preferred method is volumes: Docker Storage Volumes [https://docs.docker.com/storage/volumes/]
It's also possible to use the -v option with `docker run` to share a directory between host and container:

```
docker run -v path_in:container_path -it container_name
```

Another resource on Docker [https://forums.docker.com/t/how-to-access-containers-data-from-the-host/45453/11]
Optional: Install tmux
We will use `tmux` so we can leave processes running in the background even if we get disconnected from the server. First, update our package manager (software manager in Ubuntu/Debian Linux):

```
sudo apt update
```

`tmux` may already be installed, but if it's not:

```
sudo apt install tmux
```

Then we will create a tmux session and attach to it:

```
tmux new -s hadoop
```

Press `ctrl + b` then `d` to exit the session. List the running sessions with

```
tmux ls
```

You can attach back to the session with:

```
tmux attach -t had
```

You only need to type enough of the session name so that tmux knows it's unique from any other existing session. If you have another session called `had-hadoop`, you could do `tmux attach -t had-` to attach to it. To kill a session, there are multiple ways, such as `tmux kill-session`, but one good way is to enter the session, stop anything running in it (with ctrl+c for example), then press `ctrl + d` or type `exit`. Try killing the hadoop session and recreating it.
Optional: Install a point-and-click text editor for the terminal
Options include:
Micro
Slap
Slap
Slap is an IDE like Sublime, but can be run from within the terminal. Install the NodeJS package manager first:

```
sudo apt install npm
```

Then install slap:

```
curl -sL https://raw.githubusercontent.com/slap-editor/slap/master/install.sh [https://raw.githubusercontent.com/slap-editor/slap/master/install.sh] | sh
```

```
editor/slap/master/install.sh) | sh
```

To use slap with a new file, first create the file:

```
touch test.py
```

then open with slap and play around: `slap test.py`. You can point-and-click; ctrl+s is save and ctrl+q is quit. It's got other features too, but this should be sufficient for us.

Micro

Install with `sudo apt install snapd` then `snap install micro --classic`. You should then log out of the SSH session and log back in (refreshing your PATH environment variable). Use `micro test.py` to open a file, hotkeys like `ctrl+s` work as expected. One issue is copy-pasting in and out of micro doesn't seem to work well (or at all in some cases).

Alternative install for micro

```
curl https://getmic.ro | bash
```

Then copy the executable to `/usr/bin`:

```
sudo cp ./micro /usr/bin
```

`/usr/bin` is in the `PATH`, and any command we try to run will search through our `PATH` environment variable. So putting `micro` in a folder in `PATH` makes it so we can run `micro` from anywhere on our system.

Reflect in ePortfolio Download Print

Activity Details

Task: View this topic

If you have a lot of experience with Linux, then you can go through the Linux portion of the lab 1 quickly. If not, I'd highly recommend taking some time to learn the Linux basics. Linux will be used in other Data Engineering courses that you may take in the MS Data Science program.

Please complete the lab through the word count example. The word count example is the "hello world" of Hadoop processing. An alternate assignment is mentioned in the lab, however you can ignore it.