Summary

This week, we will get started with some essential data engineering [https://www.stitchdata.com/resources/glossary/data-engineering/] tools. These are:

Linux

GCP (Google Cloud Products)

`tmux`

`Docker`

`Hadoop`

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?

Read through the 'Introduction to Linux' section of this document, and setup a GCP VM (explained in the lab 1 document). Try a few Linux commands in the VM, like `ls`, `ls -la`, and `cd`.

Please complete this assignment 1 (lab) and upload your MS Word document to the Worldclass Assignment drop box for labs. Include your name as part of the lab assignment (e.g. John-Smith-Week-1-Lab) and include screen shots from various points in the lab.

Also, as mentioned above, look at the lab 1 document with screen shots that can help you fill in some of the gaps and/or answer questions about the lab assignment.

If you have a lot of experience with Linux, then you can go through the Linux portion of the lab quickly. If not, then take some time to learn the Linux basics.

The week lab 1 is due at the end of the first week. 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.

Please download and review this lab document that will walk you through assignment 1.

Lab 1 using the Google Cloud [../Week1-lab-as-of-6-13-2022.docx?ou=350471]

Introduction to Linux

Linux is an operating system with many 'flavors', the most common of which are Ubuntu, Debian, and Red Hat distributions. A distribution is a specific type of Linux OS which has specific ways of installing software and running the OS.

Linux is used in many server applications, such as for big data, serving websites, and running critical software processes for organizations.

When using Linux, we usually use the command line, or terminal. This is because we may often be interacting with a remote server, and we can't easily plug a monitor in to it.

Basic Linux commands

Within the terminal, we have some basic Linux commands. Here are some:

Filesystem navigation and manipulation:

`pwd` -- print working directory -- prints the location in the filesystem

`ls` -- lists files/folders within the current directory

`ls -l` -- lists files with more details

`cd` -- change directory -- this moves use to a directory; using `cd ..` moves us up one directory

`mkdir` -- makes a directory (folder)

`touch`

`rm` -- removes a file. We can add the `-r` option to remove an entire directory

`sudo` -- super user do. This gives administrative privaledges to run commands

`man` -- brings up the help for a command, e.g. `man cd`

`which` -- locates a file for a command; e.g. to see where python is located, we would do `which python`. It shows either the location of Python, or a blank line if Python is not in our PATH (PATH is discussed in the next section).

`whereis` -- similar to `which`, but shows more locations. `which` just shows the first location

`echo` -- prints out whatever comes after `echo` -- we'll see why this is useful in the next section

`export` -- sets an environment variable

`source` -- runs a file, e.g. `source ~/.bashrc`

`logout` -- logs out of the current session

`reboot` -- restarts computer

`shutdown` -- shuts down computer

`-chmod` -- change permissions (read, write, execute) of files

`chown` -- change group and user owners of files/folders

`cat`

`head`

`tail`

Some other nice things to know:

`ctrl + c` will cancel a running process

`ctrl + d` usually quits things

`ctrl + arrows` (left and right arrows) usually skips text by word chunks

`ctrl + delete/backspace + arrows` (L/R arrows) usually deletes by word chunks at a time

pressing `tab` usually auto-completes things

Environment variables

Environment variables store information for the OS to use. A simple example is the home directory, or `$HOME`. We can examine the contents of this variable with `echo $HOME`. Note we need a dollar sign before the variable name. If we want to create a new environment variable, we can do `export NEW_VAR=testvar`. We can check that it worked with `echo $NEW_VAR`. If an environment variable doesn't exist or have a value, it'll print out a blank line.

Another important example is the `PATH` environment variable. This stores filepaths/directories where the OS will looks for commands when you type them. Often this includes `/usr/bin`, where many executable files are. We can see the `PATH` environment variable with `echo $PATH`. To use an environment variable in the terminal, we need to put a dollar sign before it. Environment variables are present in all OSs (Windows, Mac, Linux), and the PATH in particular is in all of them. You'll see it has filepaths separated by colons in Linux. We can add a new folder to our path like so:

`export PATH=/usr/bin:$PATH`

In that case, we added the /usr/bin folder to the PATH, and our OS will check the /usr/bin folder first any time it's looking for a command.

We can also store environment variables in a few files:

`~/.bashrc`

`~/.profile`

The '~' is the same as `$HOME`, our home directory. The ~/.bashrc file is loaded every time we create a new terminal, and the ~/.profile is loaded each time we log in. We can re-run these with `source ~/.bashrc` if we change them and want the changes loaded.

Users and file/folder ownership

There are users within our OS, just like with Mac and Windows. These users can be created and deleted. The "ultimate" user is 'root', which has full power to do anything on the system. When we run commands with `sudo` before them, we are running it as root.

When doing `ls -l`, we see the owners of files, as well as other information. We won't go into it now, but we can change ownership with `chown`, and change permissions (e.g. can a file be executed) with `chmod`.

Installing software

The different Linux distributions install software slightly differently. Ubuntu and Debian distros use the aptitude package manager with `.deb` files. These are like `.exe` files in Windows, or `.pkg` files in Mac. We can update and upgrade our current software in Ubuntu/Debian as shown below. The sudo command in linux gives you extended permissions to install software.

`sudo apt-get update` -- checks for updates

`sudo apt-get upgrade` -- upgrades current software

`sudo apt-get install` -- installs new software

Another Linux distribution, Red Hat (and a subversion of Red Hat, CentOS) uses the `yum` package manager and `.rpm` files. So if you find yourself with some unknown Linux distribution, then you can try the commands `apt` and `yum` to see which one works. This will at least tell you if it's an Ubuntu/Debian version or Red Hat / CentOS type Linux.

Updating software with yum works like:

`yum update`

[https://www.cyberciti.biz/faq/linux-update-all-packages-command/]

Important!

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). 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.

Set up GCP

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.

Cloudera quickstart

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_IsleOw] 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. This was created by Dr. George to save you time on the Hadoop setup.

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

These are covered in the lab 1 document, however they are re-iterated below as FYI.

The essential commands for install 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

All of the commands below are covered in the lab 1 document. Tmux is useful to know, however it is not essential for lab 1. Using docker is essential for lab 1.

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).

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. Lab 1 walks you through the word count example. However, the information is re-iterated below as an FYI.

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`.

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
```

An alternative (perhaps older) way to run `hdfs dfs` is `hadoop fs`. The lab 1 uses hadoop fs and you will notice this approach as you work through lab 1. However, you are welcome to use hdfs if you prefer.

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
```

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

Sharing data between docker and the host filesystem

FYI - 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 software installs in Ubuntu Linux
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):
Note: As mentioned previously, tmux is a useful tool, but is not required for lab 1.

```
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, these are both optional tools and not needed for lab 1. Students with advanced technical skills may want to try these tool.
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
```

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.

Bonuses (optional - total bonus up to 10 points) - For students with advanced technical skills only.

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.