EECS E6893 Big Data Analytics
HW1: Clustering, Spark MLlib, and Hadoop

Tejasri Kurapati, tk2928@columbia.edu

Agenda

- HW1
  - Iterative K-means clustering
  - Hadoop
- Spark MLlib

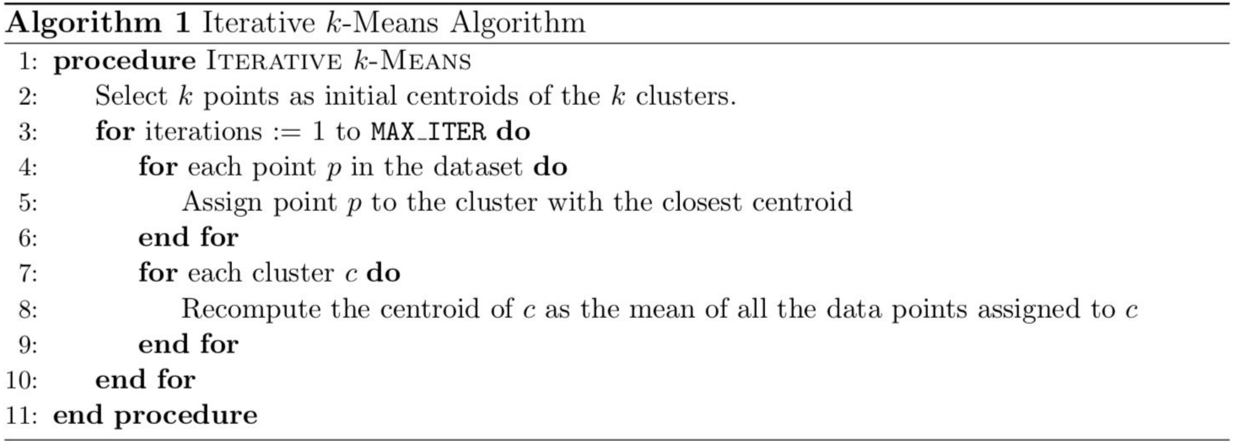
HW1

HW1

- Document clustering with K-means
  - “Implement” iterative K-means clustering in Spark
  - L1, L2 distance functions
  - Different initialization strategies
  - Plot the cluster assignment result with T-SNE dimensionality reduction
- Monitoring Hadoop metrics
  - Installing Hadoop in Pseudo Distributed Mode
  - Monitoring hadoop metrics through HTTP API

Iterative K-means

- In each iteration, k centroids are initialized, each point in the space is assigned to the nearest centroid, and the centroids are re-computed
- Pseudo code:

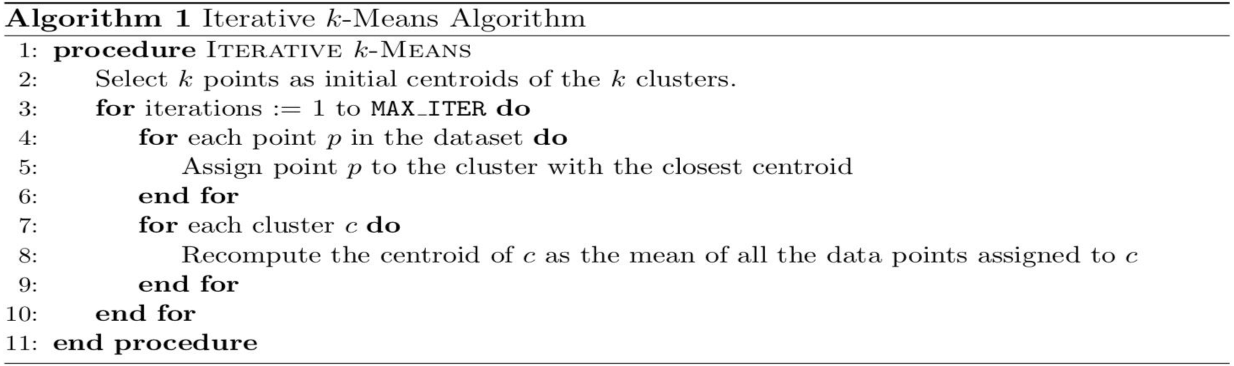


Algorithm 1 Iterative k-Means Algorithm

```
procedure ITERATIVE $k$-MEANS
    Select $k$ points as initial centroids of the $k$ clusters.
    for iterations := 1 to MAX_ITER do
        for each point $p$ in the dataset do
            Assign point $p$ to the cluster with the closest centroid
        end for
        for each cluster $c$ do
            Recompute the centroid of $c$ as the mean of all the data points assigned to $c$
        end for
    end for
end procedure
```

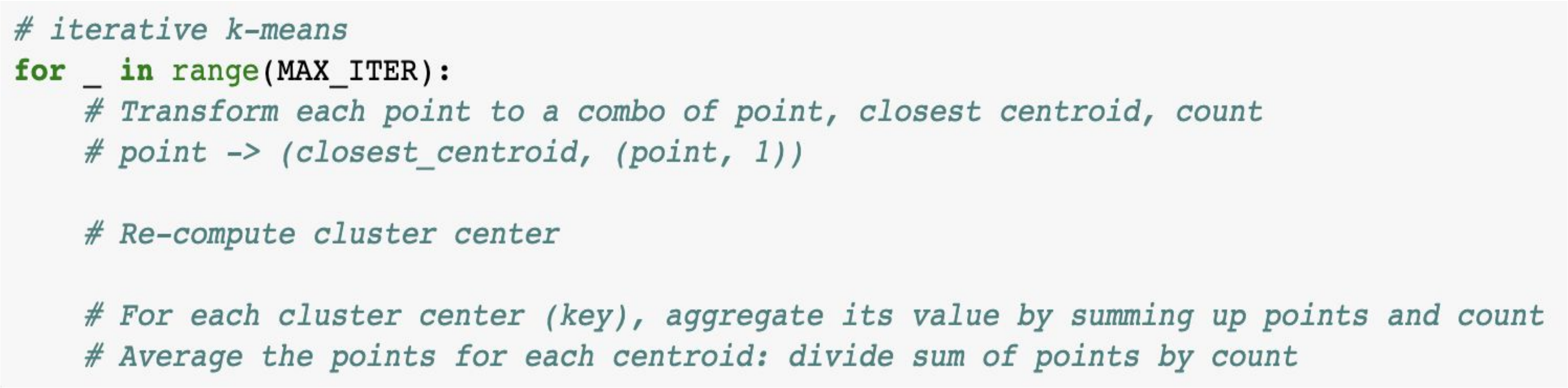
Iterative K-means in Spark

Hint:
Spark operations you might need:
map, reduceByKey, collect, keys



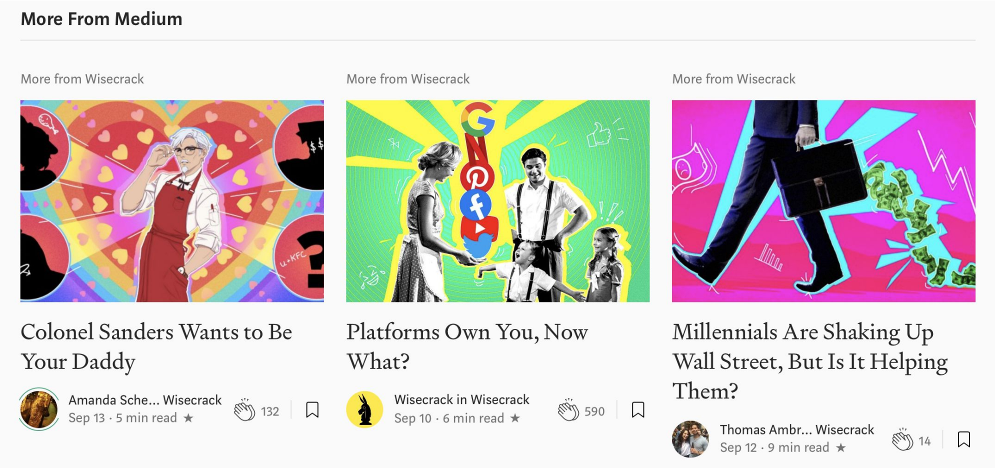
Algorithm 1 Iterative k-Means Algorithm

```
procedure ITERATIVE $k$-MEANS
    Select $k$ points as initial centroids of the $k$ clusters.
    for iterations := 1 to MAX_ITER do
        for each point $p$ in the dataset do
            Assign point $p$ to the cluster with the closest centroid
        end for
        for each cluster $c$ do
            Recompute the centroid of $c$ as the mean of all the data points assigned to $c$
        end for
    end for
end procedure
```



```
# iterative k-means
for _ in range(MAX_ITER):
    # Transform each point to a combo of point, closest centroid, count
    # point -> (closest_centroid, (point, 1))

    # Re-compute cluster center

    # For each cluster center (key), aggregate its value by summing up points and count
    # Average the points for each centroid: divide sum of points by count
```

Document clustering



More From Medium

More from Wisecrack

Colonel Sanders Wants to Be Your Daddy

Amanda Sche... Wisecrack
Sep 13 · 5 min read ★ 132

More from Wisecrack

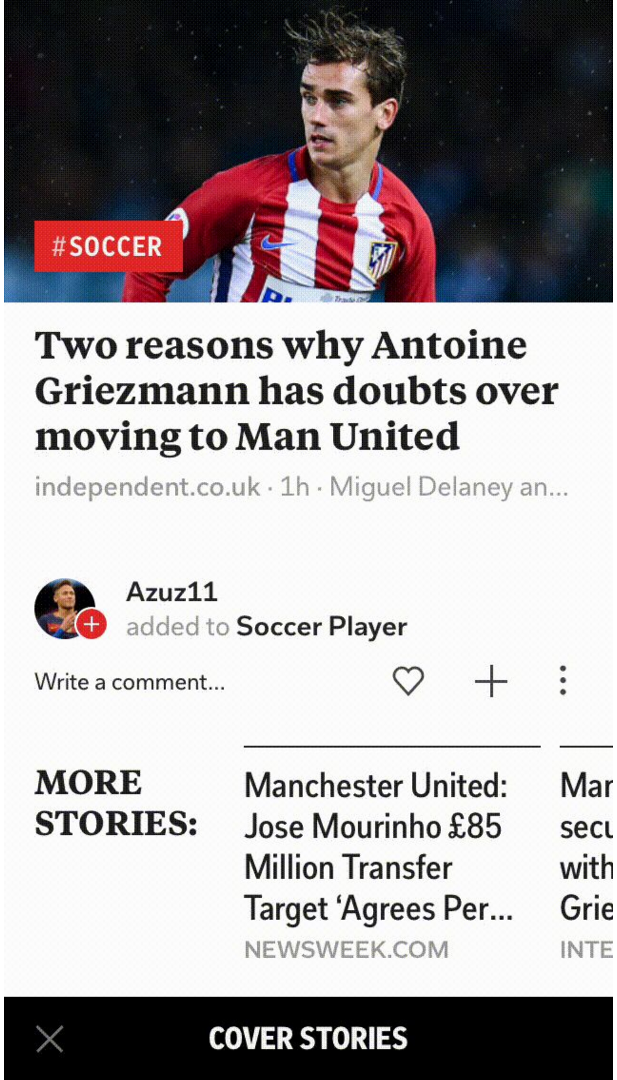
Platforms Own You, Now What?

Wisecrack in Wisecrack
Sep 10 · 6 min read ★ 590

More from Wisecrack

Millennials Are Shaking Up Wall Street, But Is It Helping Them?

Thomas Ambr... Wisecrack
Sep 12 · 9 min read ★ 14



#SOCCER

Two reasons why Antoine Griezmann has doubts over moving to Man United

independent.co.uk · 1h · Miguel Delaney an...

Azuz11
added to Soccer Player

Write a comment...

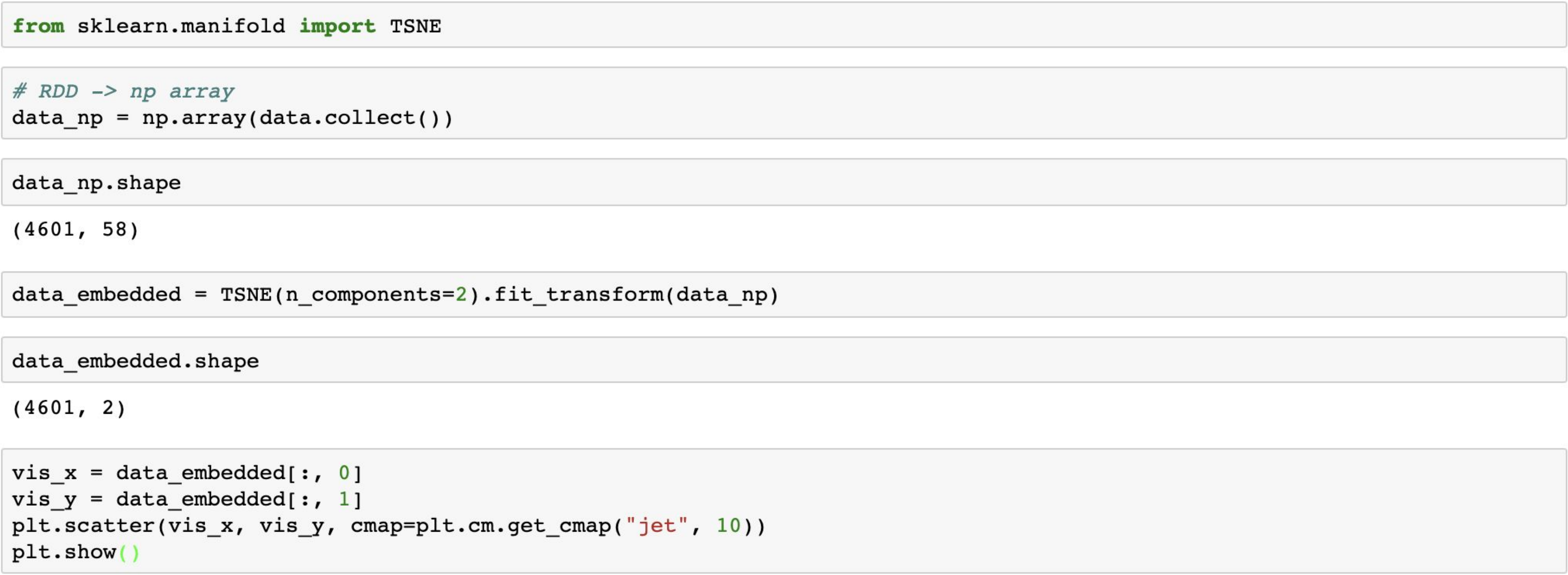
MORE STORIES:

Manchester United: Jose Mourinho £85 Million Transfer Target 'Agrees Per...
NEWSWEEK.COM

Mar secu with Grie
INTE

COVER STORIES

Plot the result with t-SNE



```python
from sklearn.manifold import TSNE
```

```python
# RDD -> np array
data_np = np.array(data.collect())
```

```python
data_np.shape
```

```
(4601, 58)
```

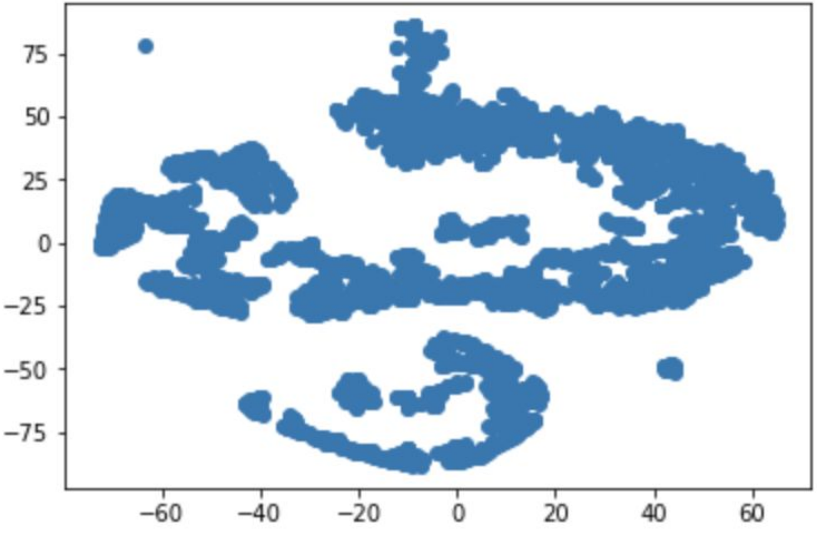
```python
data_embedded = TSNE(n_components=2).fit_transform(data_np)
```

```python
data_embedded.shape
```

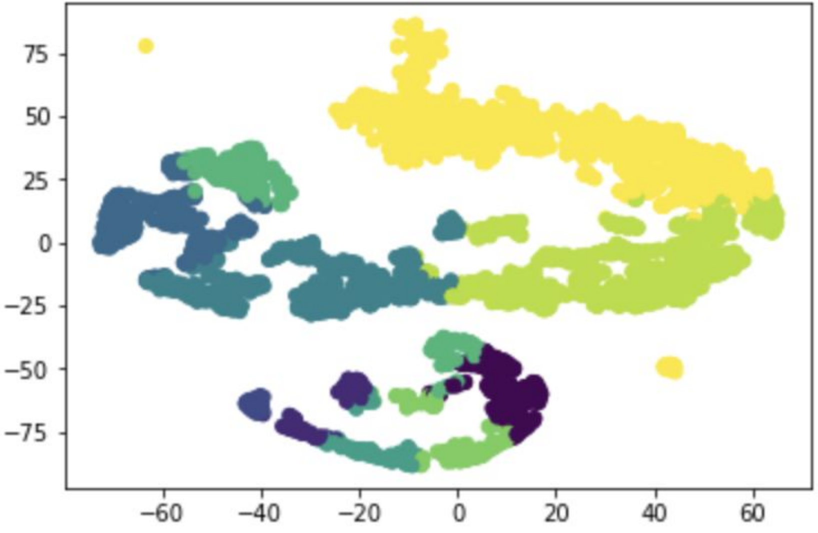
```
(4601, 2)
```

```python
vis_x = data_embedded[:, 0]
vis_y = data_embedded[:, 1]
plt.scatter(vis_x, vis_y, cmap=plt.cm.get_cmap("jet", 10))
plt.show()
```

Plot the result with t-SNE

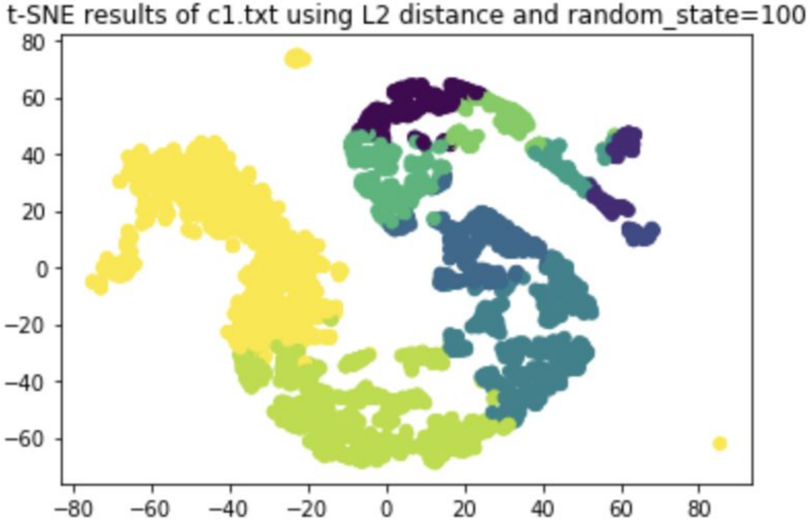


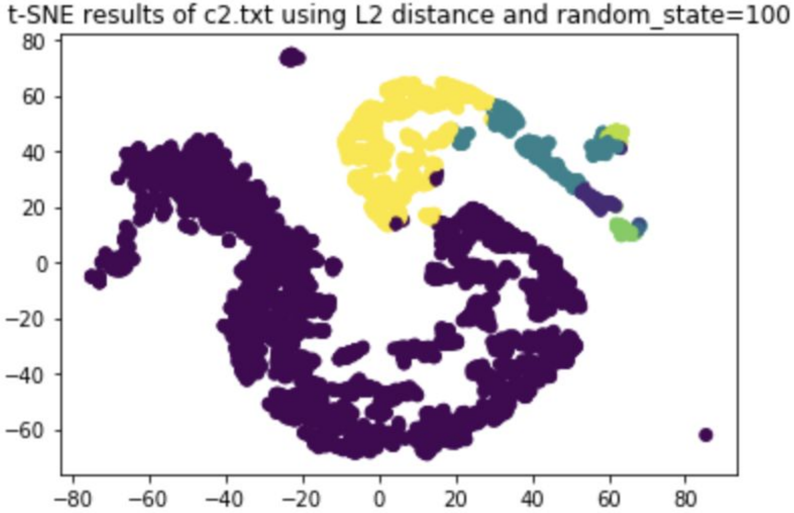
Before clustering



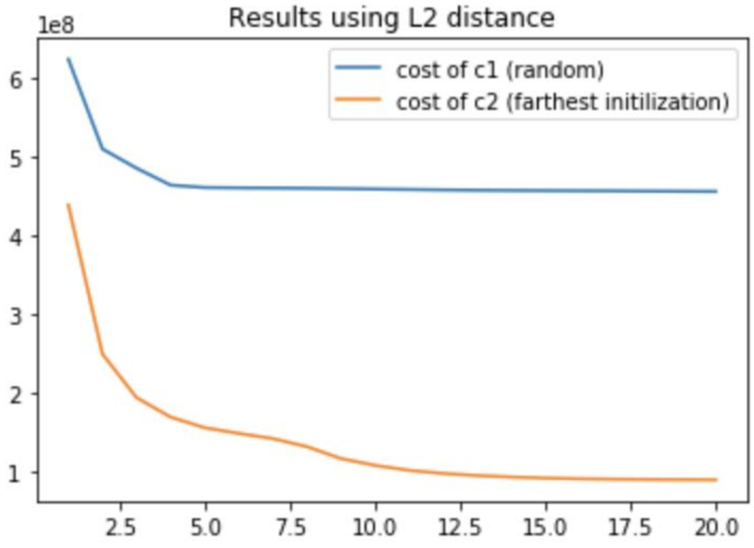
After clustering

Plot the result with t-SNE (set random state)

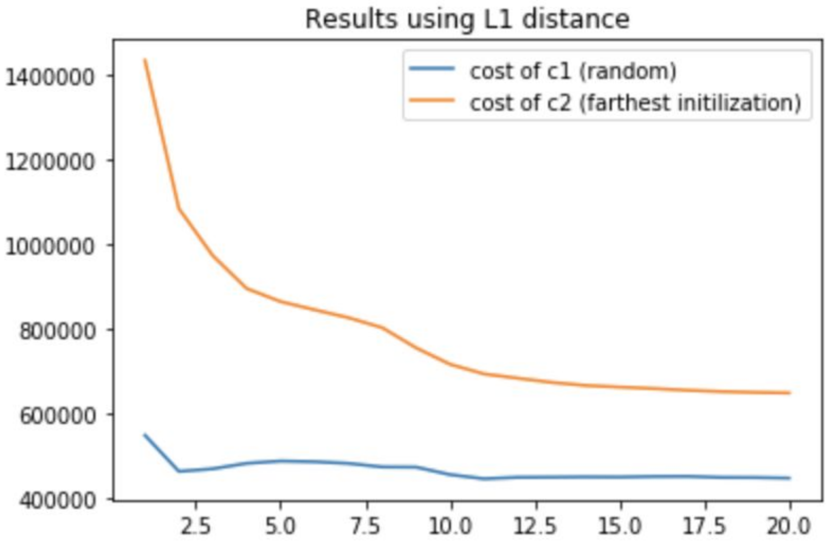




Plot the cost of each iteration



Results using L2 distance

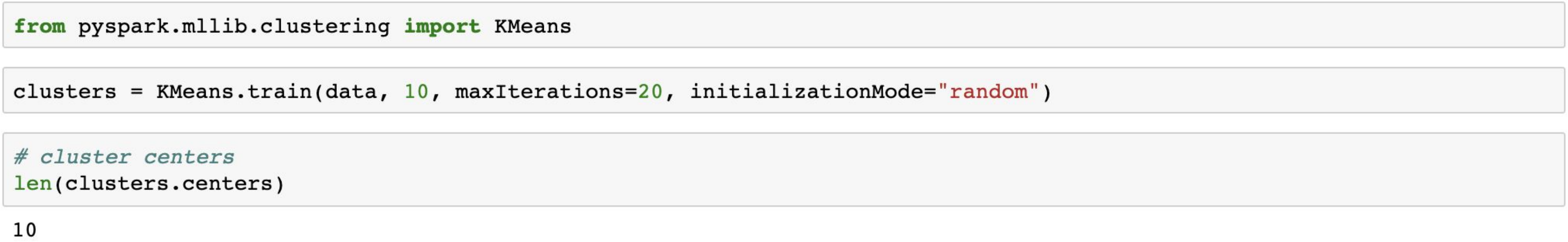


Results using L1 distance

Spark MLlib

- Spark's scalable machine learning library
- Tools:
  - ML Algorithms: classification, regression, clustering, and collaborative filtering
  - Featurization: feature extraction, transformation, dimensionality reduction, and selection
  - Pipelines: tools for constructing, evaluating, and tuning ML Pipelines
  - Persistence: saving and load algorithms, models, and Pipelines
  - Utilities: linear algebra, statistics, data handling, etc.

Example: K-means clustering with Spark MLlib



```
from pyspark.mllib.clustering import KMeans
```

```
clusters = KMeans.train(data, 10, maxIterations=20, initializationMode="random")
```

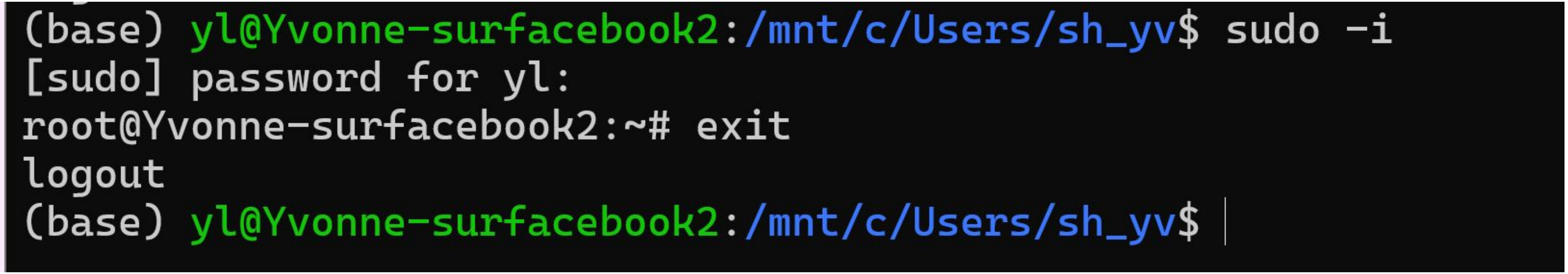
```
# cluster centers
len(clusters.centers)
```

```
10
```

Hadoop installation

Step 1: Pre-installation Setup

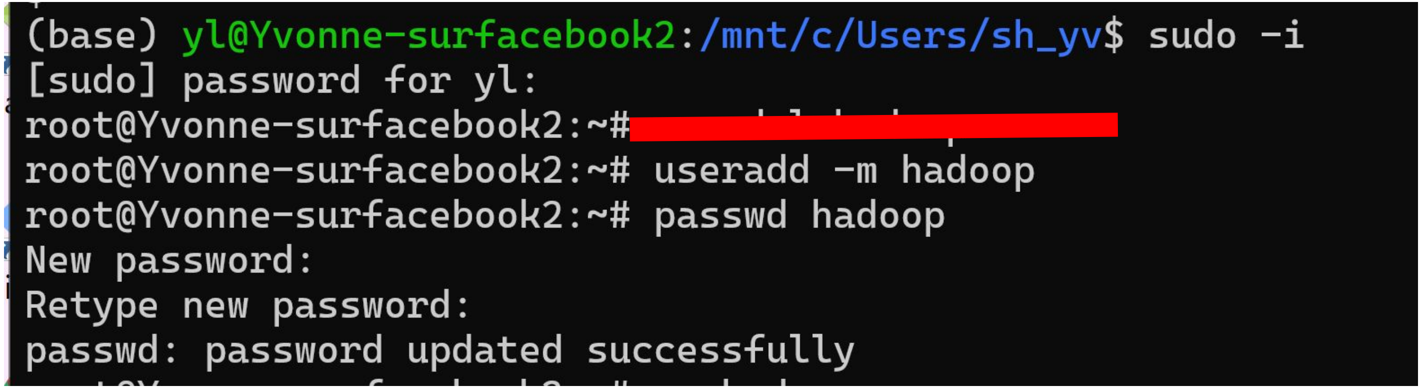
- Before the installation, learn how to login & exit the root account
  - Login: sudo -i
  - Exit: exit (or use ctrl+D)



```
(base) yl@Yvonne-surfacebook2:/mnt/c/Users/sh_yv$ sudo -i
[sudo] password for yl:
root@Yvonne-surfacebook2:~# exit
logout
(base) yl@Yvonne-surfacebook2:/mnt/c/Users/sh_yv$
```

Create a user

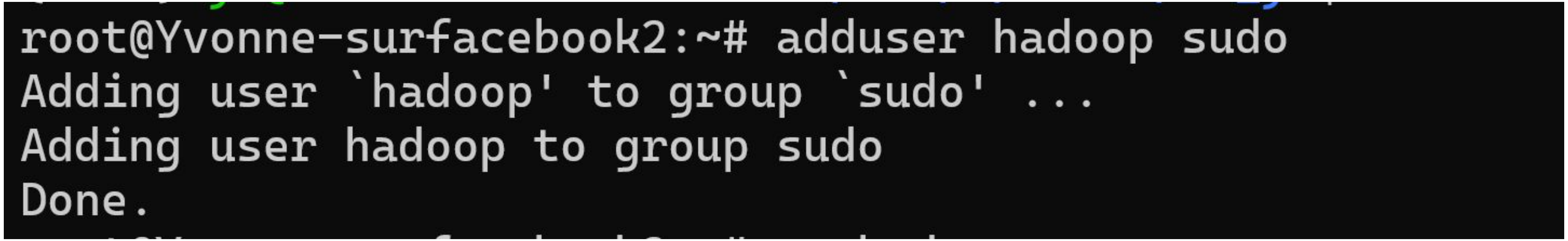
- Open the **root** using the command “sudo -i”.
- Create a user from the root account using the command “useradd -m username”.
- Set the password using the command “passwd username”
- Now you can open the new user account.
  - If you’re under root account, use the command “su username”
  - Otherwise, use “su - username”



```
(base) yl@Yvonne-surfacebook2:/mnt/c/Users/sh_yv$ sudo -i
[sudo] password for yl:
root@Yvonne-surfacebook2:~#[redacted]
root@Yvonne-surfacebook2:~# useradd -m hadoop
root@Yvonne-surfacebook2:~# passwd hadoop
New password:
Retype new password:
passwd: password updated successfully
```

- # Create a user

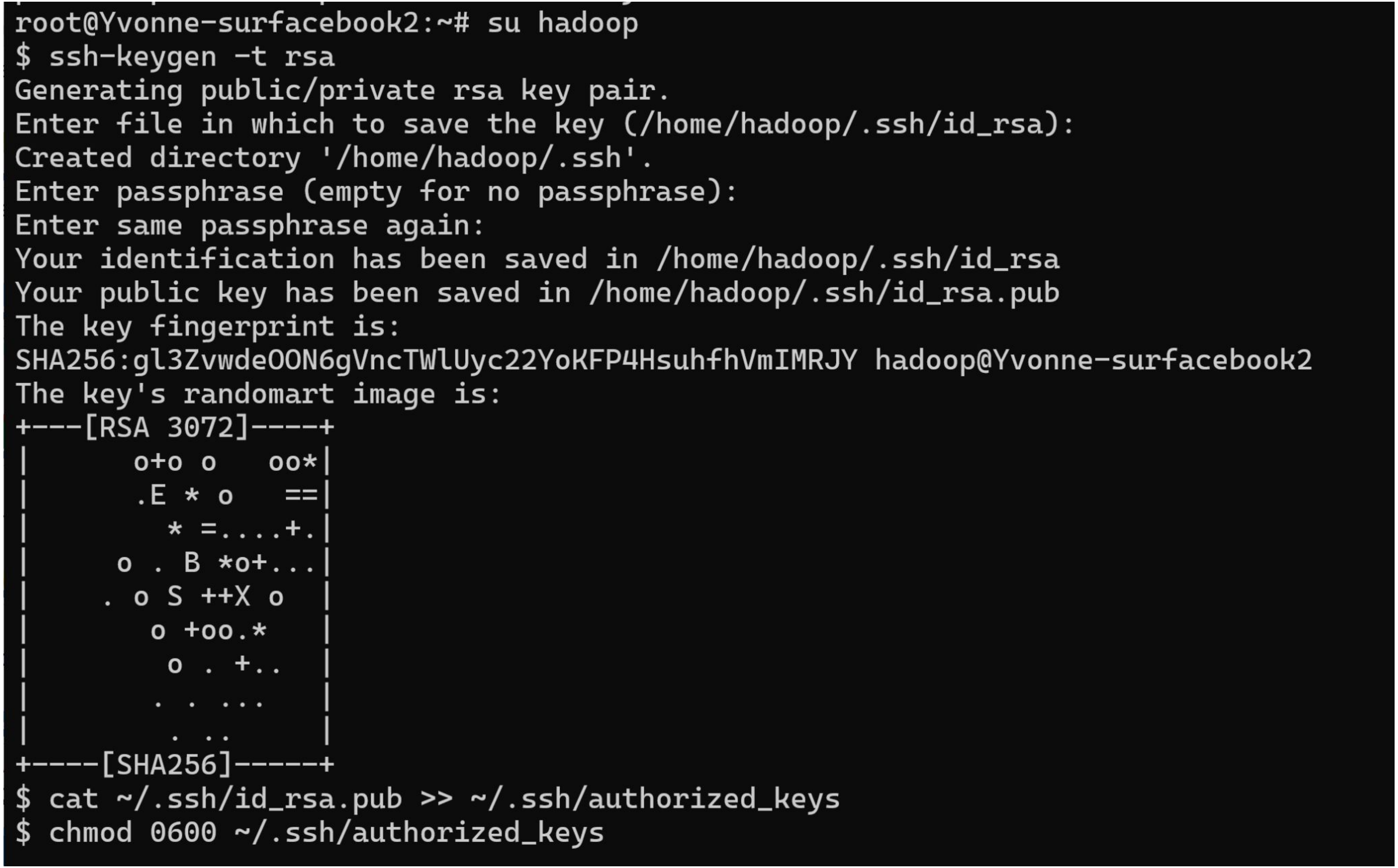
- Add user to sudo group



```
root@Yvonne-surfacebook2:~# adduser hadoop sudo
Adding user `hadoop' to group `sudo' ...
Adding user hadoop to group sudo
Done.
```

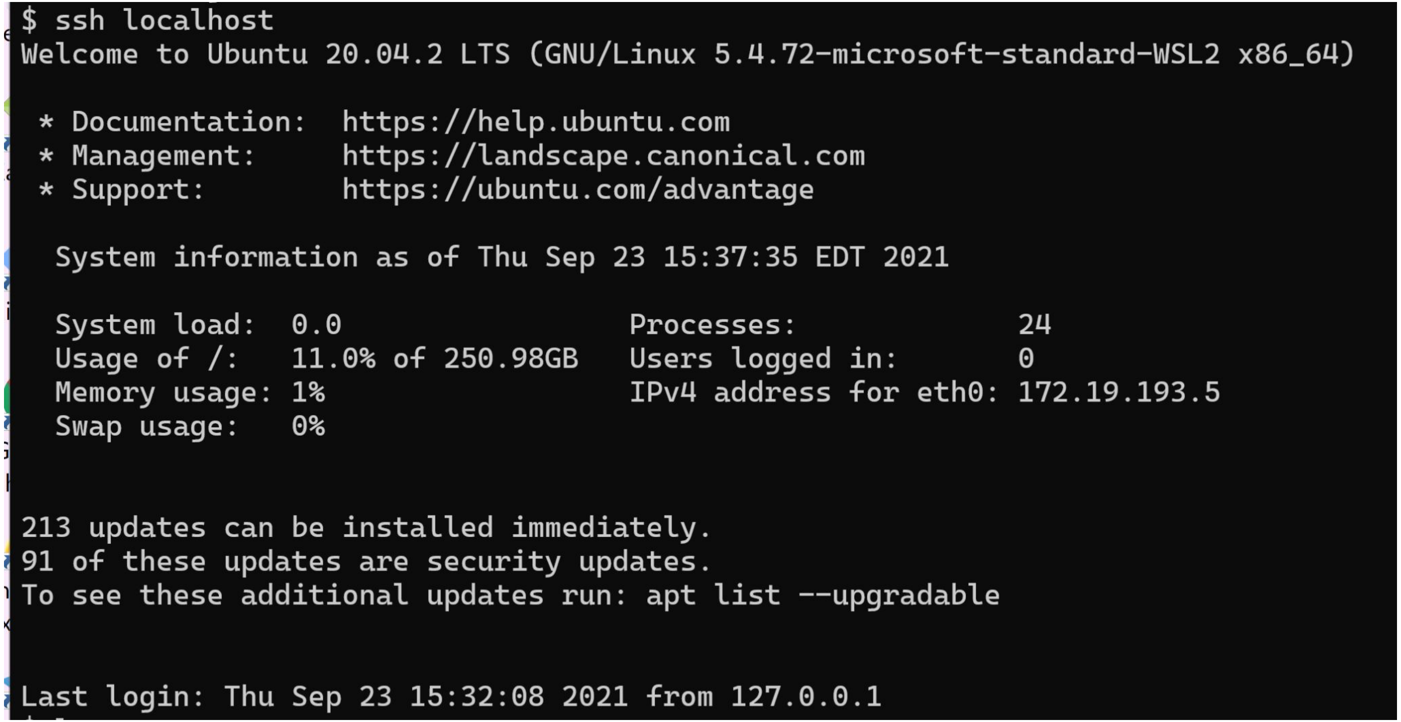
SSH Setup and Key Generation

- Open the account you created, using
  - su hadoop
- Generate generating a key value pair using SSH, using
  - ssh-keygen -t rsa (press “enter” directly where you’re asked to enter)
- Copy the public keys from id_rsa.pub to authorized_keys, using
  - cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
- Provide the owner with read and write permissions to authorized_keys file respectively
  - chmod 0600 ~/.ssh/authorized_keys
- Test SSH setup
  - ssh localhost



```
root@Yvonne-surfacebook2:~# su hadoop
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hadoop/.ssh/id_rsa):
Created directory '/home/hadoop/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/hadoop/.ssh/id_rsa
Your public key has been saved in /home/hadoop/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:gl3ZvwdeOON6gVncTWlUyc22YoKFP4HsuhfhVmIMRJY hadoop@Yvonne-surfacebook2
The key's randomart image is:
+---[RSA 3072]----+
|      o+o o   oo*|
|      .E * o   ==|
|        * =....+.|
|     o . B *o+...|
|    . o S ++X o  |
|       o +oo.*   |
|        o . +..  |
|       . . ...   |
|        . ..     |
+----[SHA256]-----+
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ chmod 0600 ~/.ssh/authorized_keys
```

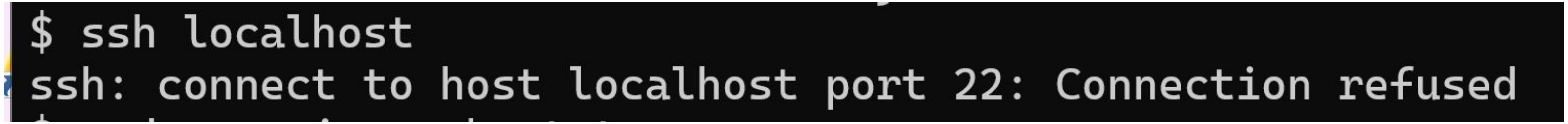
- Test ssh setup. Use “logout” command to log out



```
$ ssh localhost
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.72-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu Sep 23 15:37:35 EDT 2021

  System load:  0.0                Processes:             24
  Usage of /:   11.0% of 250.98GB   Users logged in:       0
  Memory usage: 1%                 IPv4 address for eth0: 172.19.193.5
  Swap usage:   0%


213 updates can be installed immediately.
91 of these updates are security updates.
To see these additional updates run: apt list --upgradable


Last login: Thu Sep 23 15:32:08 2021 from 127.0.0.1
```

SSH Setup (for Debugging)

- If ssh localhost doesn’t work

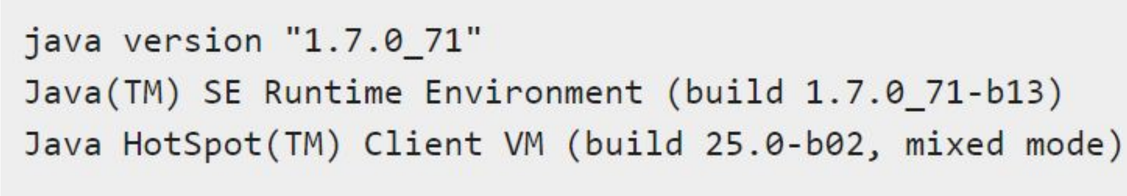


```
$ ssh localhost
ssh: connect to host localhost port 22: Connection refused
```

- Try reinstall some packages:
  - sudo apt-get remove openssh-client openssh-server
  - sudo apt-get install openssh-client openssh-server
- If still doesn’t work, check the following
  - sudo service ssh start
  - ssh localhost

- # Installing Java

- Verify the existence of Java in your system
  - java -version
  - If you’ve installed Java, it will give you the following output, and you can skip the java installing steps, continuing to the next section.

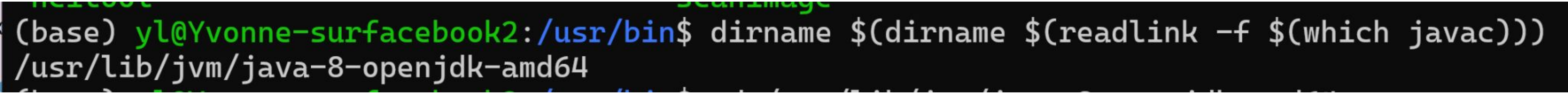


```
java version "1.7.0_71"
Java(TM) SE Runtime Environment (build 1.7.0_71-b13)
Java HotSpot(TM) Client VM (build 25.0-b02, mixed mode)
```

- If you did not install Java, you need to follow the next steps to install Java.

Installing Java

- Install java
  - sudo apt-get install openjdk-8-jre openjdk-8-jdk
- Then check Java version to see if you have installed java
  - java -version
- To find where you have installed java
  - dirname $(dirname $(readlink -f $(which javac)))



```
(base) yl@Yvonne-surfacebook2:/usr/bin$ dirname $(dirname $(readlink -f $(which javac)))
/usr/lib/jvm/java-8-openjdk-amd64
```

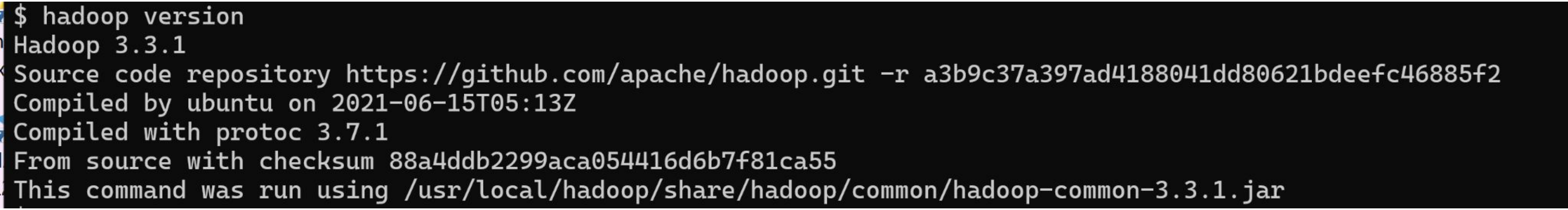
- Set up PATH and JAVA_HOME variables
  - export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64 (the path from last step)
  - export PATH=$PATH:$JAVA_HOME/bin
- Now apply all the changes into the current running system.
  - exec bash

Step 2: Downloading Hadoop

- Change to root and change directory
  - sudo -i
  - cd /usr/local/
- Download and extract Hadoop
  - wget http://apache.claz.org/hadoop/common/hadoop-3.3.1/hadoop-3.3.1.tar.gz
  - tar xzf hadoop-3.3.1.tar.gz
  - mv hadoop-3.3.1 hadoop
- Change owner
  - sudo chown -R hadoop:hadoop ./hadoop
- Set Hadoop environment variables
  - su hadoop
  - export HADOOP_HOME=/usr/local/hadoop
  - export PATH=$PATH:/usr/local/hadoop/bin
  - exec bash

- Test the Hadoop setup

- Type the following command
  - hadoop version
  - If everything is fine, you’ll see the following



```
$ hadoop version
Hadoop 3.3.1
Source code repository https://github.com/apache/hadoop.git -r a3b9c37a397ad4188041dd80621bdeefc46885f2
Compiled by ubuntu on 2021-06-15T05:13Z
Compiled with protoc 3.7.1
From source with checksum 88a4ddb2299aca054416d6b7f81ca55
This command was run using /usr/local/hadoop/share/hadoop/common/hadoop-common-3.3.1.jar
```

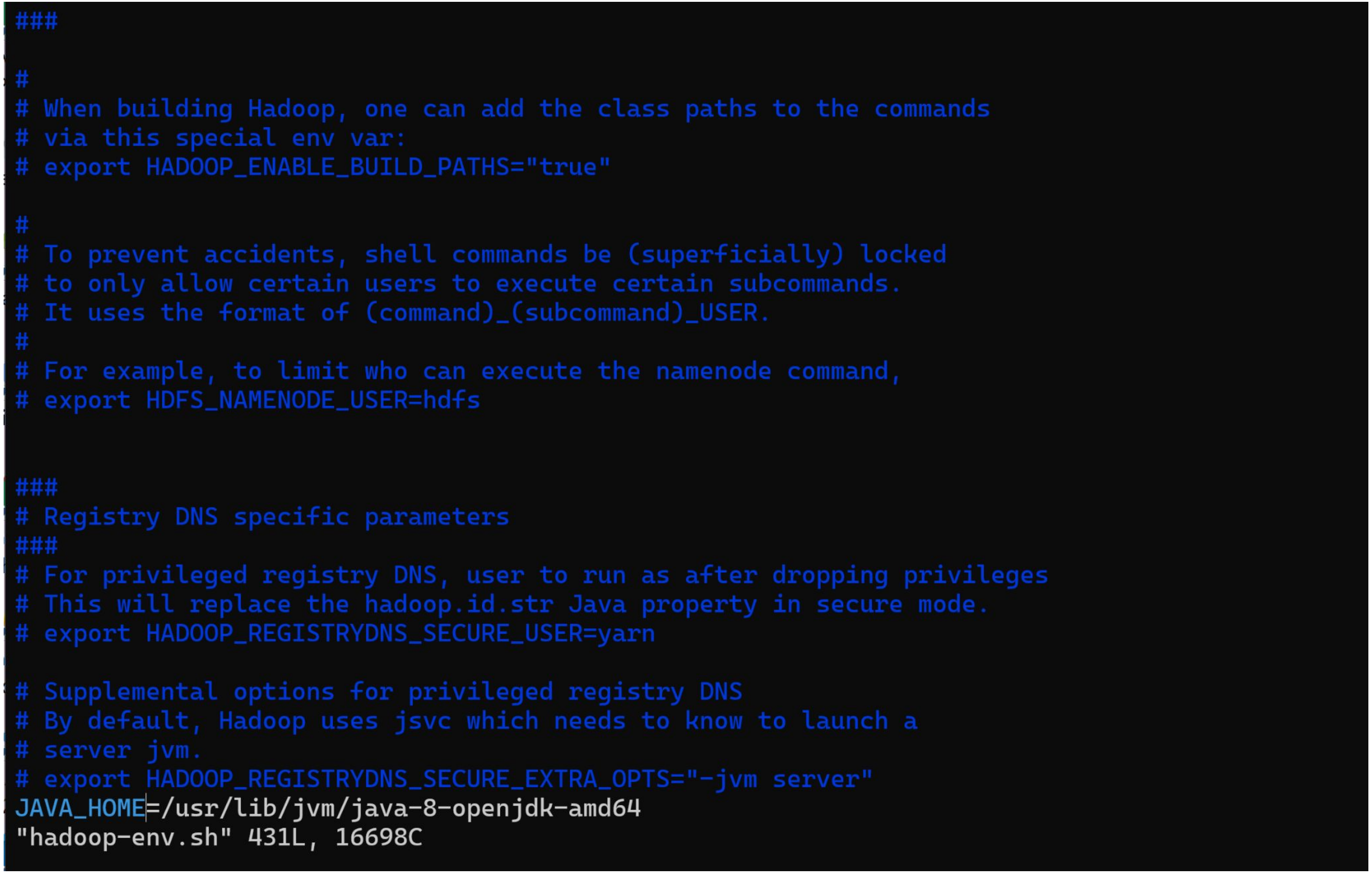
- Now you have successfully set up the Hadoop’s standalone mode

Installing Hadoop in Pseudo Distributed Mode

- Set the Hadoop environment variables
  - export HADOOP_HOME=/usr/local/hadoop
  - export HADOOP_MAPRED_HOME=$HADOOP_HOME
  - export HADOOP_COMMON_HOME=$HADOOP_HOME
  -
  - export HADOOP_HDFS_HOME=$HADOOP_HOME
  - export YARN_HOME=$HADOOP_HOME
  - export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
  - export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
  - export HADOOP_INSTALL=$HADOOP_HOME
  - exec bash

Hadoop configuration

- Find the Hadoop configuration files
  - cd $HADOOP_HOME/etc/hadoop
  - vim hadoop-env.sh (Add the location of java to this file, namely the following line)
  - JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64



```
###

#
# When building Hadoop, one can add the class paths to the commands
# via this special env var:
# export HADOOP_ENABLE_BUILD_PATHS="true"

#
# To prevent accidents, shell commands be (superficially) locked
# to only allow certain users to execute certain subcommands.
# It uses the format of (command)_(subcommand)_USER.
#
# For example, to limit who can execute the namenode command,
# export HDFS_NAMENODE_USER=hdfs


###
# Registry DNS specific parameters
###
# For privileged registry DNS, user to run as after dropping privileges
# This will replace the hadoop.id.str Java property in secure mode.
# export HADOOP_REGISTRYDNS_SECURE_USER=yarn

# Supplemental options for privileged registry DNS
# By default, Hadoop uses jsvc which needs to know to launch a
# server jvm.
# export HADOOP_REGISTRYDNS_SECURE_EXTRA_OPTS="-jvm server"
JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
"hadoop-env.sh" 431L, 16698C
```

- Some files that you need to edit to configure Hadoop
    - Open the core-site.xml and add the following properties in between <configuration>, </configuration> tags.

```
<configuration>
   <property>
      <name>fs.default.name</name>
      <value>hdfs://localhost:9000</value>
   </property>
</configuration>
```

- Some files that you need to edit to configure Hadoop

  - Open the hdfs-site.xml and add the following properties in between <configuration>, </configuration> tags.

```
<configuration>
   <property>
      <name>dfs.replication</name>
      <value>1</value>
   </property>

   <property>
      <name>dfs.name.dir</name>
      <value>file:///home/hadoop/hadoopinfra/hdfs/namenode </value>
   </property>

   <property>
      <name>dfs.data.dir</name>
      <value>file:///home/hadoop/hadoopinfra/hdfs/datanode </value>
   </property>
</configuration>
```

- Some files that you need to edit to configure Hadoop

  - Open the yarn-site.xml and add the following properties in between <configuration>, </configuration> tags.

```
<configuration>
   <property>
      <name>yarn.nodemanager.aux-services</name>
      <value>mapreduce_shuffle</value>
   </property>
</configuration>
```

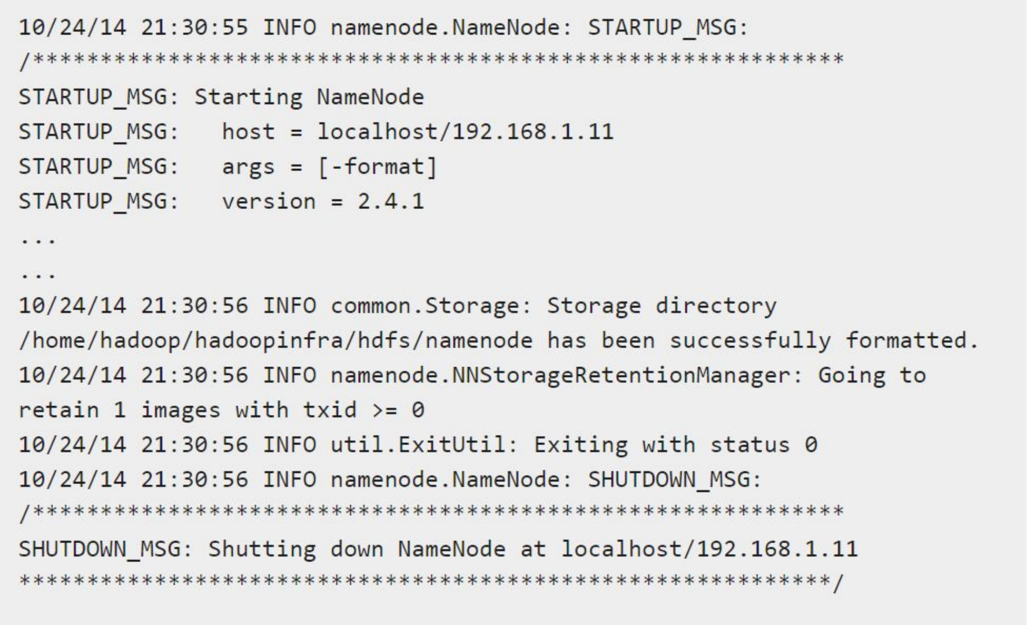
- Some files that you need to edit to configure Hadoop

  - Open the mapred-site.xml and add the following properties in between <configuration>, </configuration> tags.

```
<configuration>
   <property>
      <name>mapreduce.framework.name</name>
      <value>yarn</value>
   </property>
</configuration>
```

- # Verify Hadoop installation

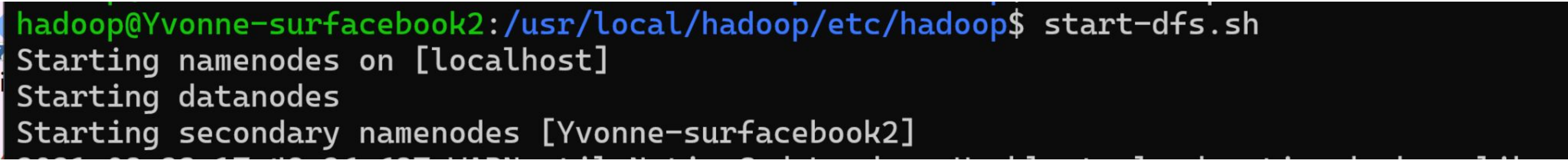
- Set up the namenode
  - cd ~
  - hdfs namenode -format



```
10/24/14 21:30:55 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = localhost/192.168.1.11
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.4.1
...
...
10/24/14 21:30:56 INFO common.Storage: Storage directory
/home/hadoop/hadoopinfra/hdfs/namenode has been successfully formatted.
10/24/14 21:30:56 INFO namenode.NNStorageRetentionManager: Going to
retain 1 images with txid >= 0
10/24/14 21:30:56 INFO util.ExitUtil: Exiting with status 0
10/24/14 21:30:56 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at localhost/192.168.1.11
************************************************************/
```

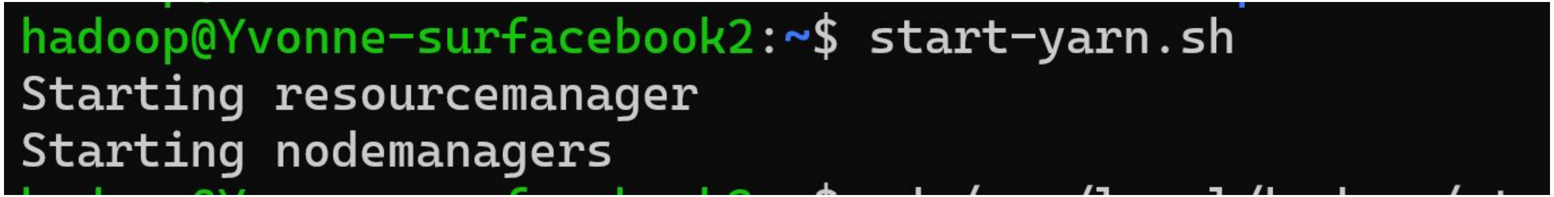
- # Verify Hadoop installation

- Verify Hadoop dfs
  - Start-dfs.sh



```
hadoop@Yvonne-surfacebook2:/usr/local/hadoop/etc/hadoop$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [Yvonne-surfacebook2]
```

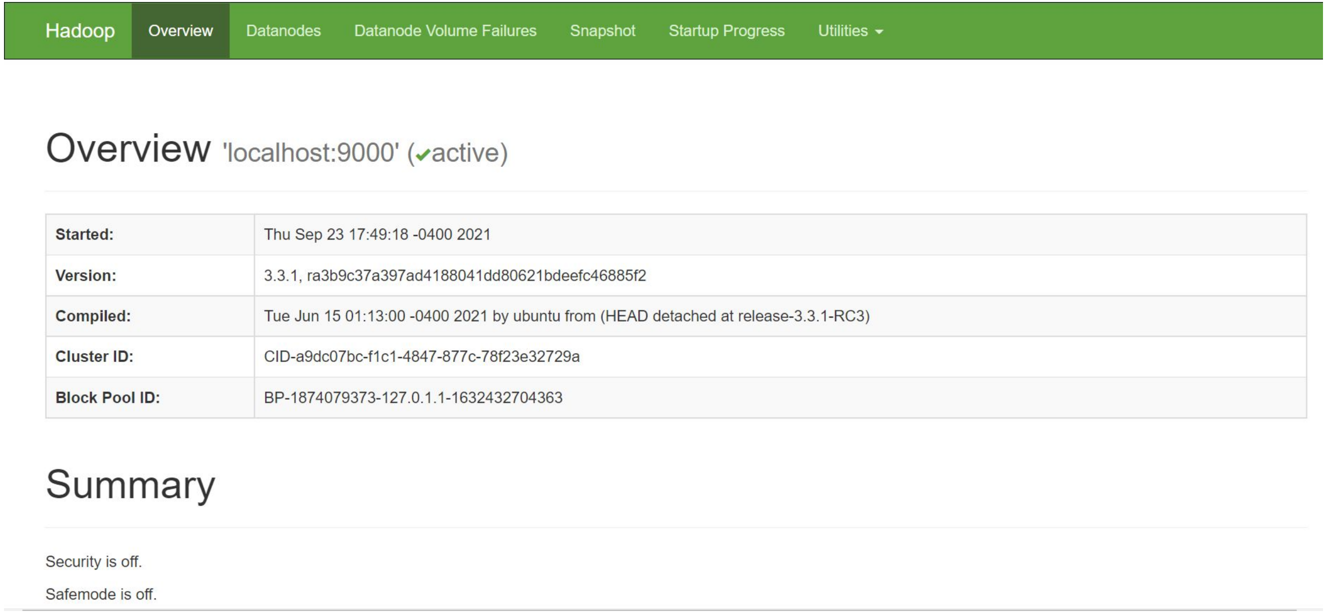
- Verify yarn script
  - start-yarn.sh



```
hadoop@Yvonne-surfacebook2:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
```

Access Hadoop on Browser

- Use the following url to get Hadoop services on browser.
  - http://localhost:9870/



Overview 'localhost:9000' (✓active)

Started:	Thu Sep 23 17:49:18 -0400 2021
Version:	3.3.1, ra3b9c37a397ad4188041dd80621bdeefc46885f2
Compiled:	Tue Jun 15 01:13:00 -0400 2021 by ubuntu from (HEAD detached at release-3.3.1-RC3)
Cluster ID:	CID-a9dc07bc-f1c1-4847-877c-78f23e32729a
Block Pool ID:	BP-1874079373-127.0.1.1-1632432704363

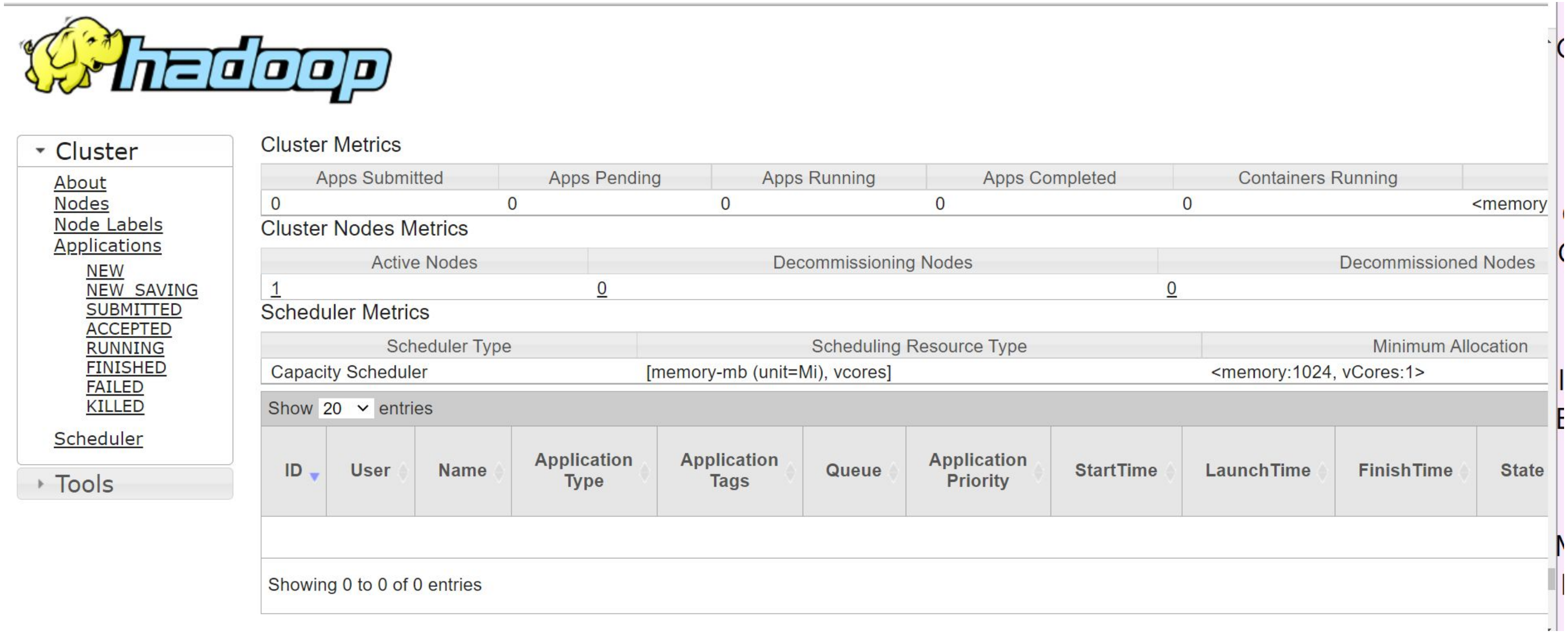
Summary

Security is off.

Safemode is off.

Access Hadoop on Browser

- Access all applications of cluster
  - http://localhost:8088/



Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	
0	0	0	0	0	<memory

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes
1	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>

Show 20 entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State

Showing 0 to 0 of 0 entries

References

- https://spark.apache.org/docs/latest/sql-getting-started.html
- https://www.analyticsvidhya.com/blog/2016/10/spark-dataframe-and-operations/
- https://spark.apache.org/docs/latest/ml-guide.html
- https://towardsdatascience.com/machine-learning-with-pyspark-and-mllib-solving-a-binary-classification-problem-96396065d2aa