**How to use a Machine Learning Model to Make Predictions on Streaming Data using PySpark**

Overview

* Streaming data เป็นหลักการที่น่าตื่นเต้นของ machine learning
* เรียนรู้วิธีการใช้ machine learning (logistic regression) เพื่อทำนาย Streaming data โดยใช้ PySpark
* ครอบคลุมพื้นฐานของ Streaming data และ Spark Streaming

Introduction

ทุกวินาทีมีการส่ง Tweets มากกว่า 8,500 Tweets รูปภาพ มากกว่า 900 รูปภาพถูกอัพโหลดบน Instagram มีการโทร Skype มากกว่า 4,200 ครั้ง มีการโทรค้นหาโดย google มากกว่า 78,000 ครั้งและมีการส่ง email มากกว่า 2 ล้านฉบับ

เรากำลังสร้างข้อมูลในระดับที่ไม่เคยเกิดขึ้นมาก่อน เป็นเวลาที่ดีที่จะได้ทำงานในพื้นที่วิทยาศาสตร์ข้อมูล! แต่ด้วยข้อมูลที่ยอดเยี่ยมความท้าทายที่ซับซ้อนเท่าเทียมกัน

เราจะรวบรวมข้อมูลในระดับนี้ได้อย่างไร เราจะรู้ได้อย่างไรว่า machine learning pipeline  ยังคงประมวลผลลลัพธ์ทันที่มีการสร้างและรวบรวมข้อมูล สิ่งเหล่านี้เป็นงานสำคัญในอุตสาหกกรมที่ท้าทายที่ต้องเผชิญ และ เหตุใดแนวคิด Streaming data จึงได้รับความสนใจมากขึ้นระหว่างองค์กร

ดังนั้นในบทความนี้เราจะได้เรียนรู้ว่าข้อมูลแบบสตรีมคืออะไรเข้าใจพื้นฐานของการสตรีมแบบ Spark แล้วทำงานกับชุดข้อมูลที่เกี่ยวข้องกับอุตสาหกรรมเพื่อนำข้อมูลแบบสตรีมมิ่งไปใช้โดยใช้ Spark



What is Streaming Data?

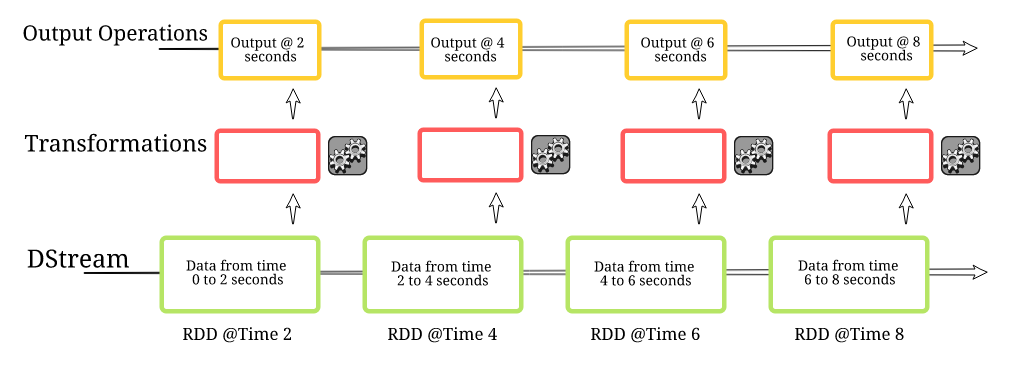
Streaming data ไม่มีจุดเริ่มต้นหรือจุดสิ้นสุดแบบต่อเนื่อง ข้อมูลนี้สร้างขึ้นทุกๆ วินาที จากข้อมูลหลายแหล่งและจำเป็นต้องประมวลผลและวิเคราะห์โดยเร็วที่สุด ข้อมูลสตรีมจำนวนมากต้องได้รับการประมวลผลแบบ real-time เช่นการค้นหาของ Google

Fundamentals of Spark Streaming

Spark Streaming เป็นส่วนเสริมของ Core Spark API ที่ช่วยให้สามารถปนะมวลผลสตรีมข้อมูลล่าสุดที่ปรับขนาดได้และไม่ผิดพลาดมาทำความเข้าใจกับองค์ประกอบต่าง ๆ ของ Spark Streaming

**1.Discretized Streams**

Discretized Streams, or DStreams, represent a continuous stream of data ขั้นตอนแรกของการสร้างแอพพลิเคชั่นสตรีมมิ่งคือการกำหนดระยะเวลาแบทช์สำหรับแหล่งข้อมูลที่เรารวบรวมข้อมูล หากระยะเวลาแบทช์คือ 2 วินาทีข้อมูลจะถูกเก็บรวบรวมทุก 2 วินาทีและเก็บไว้ใน RDD และสายโซ่ของซีรีย์ต่อเนื่องของ RDD เหล่านี้คือ DStream ซึ่งไม่เปลี่ยนรูปและสามารถใช้เป็นชุดข้อมูลแบบกระจายโดย Spark



ในระหว่างขั้นตอนการประมวลผลข้อมูลล่วงหน้าเรา จำเป็นต้อง transform variables, including converting categorical ones into numeric, creating bins ลบค่าผิดปกติ Spark เก็บข้อมูลประวัติการ transformations ที่เรากำหนดไว้ในข้อมูลใดๆ ดังนั้นเมื่อใดก็ตามที่มีความผิดพลาดเกิดขึ้นมันสามารถย้อนเส้นทางของ transformations และการสร้างผลลัพธ์ที่คำนวณใหม่อีกครั้ง

เราต้องการให้แอปพลิเคชั่น Spark ของเราทำงาน 24 x 7 และเมื่อใดก็ตามที่มีข้อผิดพลาดเกิดขึ้นเราต้องการให้มันกู้คืนโดยเร็วที่สุด แต่ในขณะที่ทำงานกับข้อมูลขนาดใหญ่ Spark จำเป็นต้องคำนวณการแปลงทั้งหมดอีกครั้งในกรณีที่มีข้อผิดพลาด อย่างที่คุณจินตนาการได้อาจมีราคาค่อนข้างสูง

**2.Caching**

นี่คือวิธีหนึ่งในการจัดการกับความท้าทายนี้ เราสามารถเก็บผลลัพธ์ที่เราคำนวณ (แคช) ชั่วคราวเพื่อรักษาผลลัพธ์ของการแปลงที่กำหนดไว้ในข้อมูล ด้วยวิธีนี้เราไม่ต้องคำนวณการแปลงเหล่านั้นซ้ำแล้วซ้ำอีกเมื่อเกิดความผิดพลาดใด ๆ

DStreams ช่วยให้เราสามารถเก็บข้อมูลสตรีมในหน่วยความจำ สิ่งนี้มีประโยชน์เมื่อเราต้องการคำนวณการดำเนินการหลายอย่างในข้อมูลเดียวกัน

**3.Checkpointing**

การแคชมีประโยชน์อย่างยิ่งเมื่อเราใช้อย่างถูกต้อง แต่ต้องการหน่วยความจำจำนวนมาก และไม่ใช่ทุกคนที่มีเครื่องหลายร้อยเครื่องที่มี RAM ขนาด 128 GB เพื่อแคชทุกอย่าง

Checkpointing คืออีกเทคนิคหนึ่งในการเก็บผลลัพธ์ของข้อมูลเฟรมที่ถูก transformation แล้วมันช่วยประหยัดสถานะของแอพพลิเคชั่นที่รันเป็นระยะ ๆ บนที่เก็บ reliable ข้อมูลคล้าย HDFS อย่างไรก็ตามมันช้าลงและมีความยืดหยุ่นน้อยกว่าแคช

เราสามารถใช้ Checkpointing เมื่อเรามีการสตรีมข้อมูล ผลลัพธ์การเปลี่ยนแปลงขึ้นอยู่กับผลลัพธ์การแปลงก่อนหน้าและจำเป็นต้องเก็บรักษาไว้เพื่อใช้งาน นอกจากนี้เรายังตรวจสอบข้อมูลเมตาดาต้าเช่นเดียวกับการกำหนดค่าที่ใช้ในการสร้างข้อมูลสตรีมมิ่งและผลลัพธ์ของชุดการดำเนินการ DStream

**4.Shared Variables in Streaming Data**

มีบางครั้งที่เราจำเป็นต้องกำหนดฟังก์ชั่นเช่นแผนที่ลดหรือกรองสำหรับแอปพลิเคชัน Spark ของเราที่จะต้องดำเนินการในหลาย ๆ กลุ่ม ตัวแปรที่ใช้ในฟังก์ชั่นนี้จะถูกคัดลอกไปยังแต่ละเครื่อง (กลุ่ม)

ที่นี่แต่ละคลัสเตอร์มีตัวดำเนินการแตกต่างกันและเราต้องการบางสิ่งที่สามารถให้ความสัมพันธ์ระหว่างตัวแปรเหล่านี้กับเรา

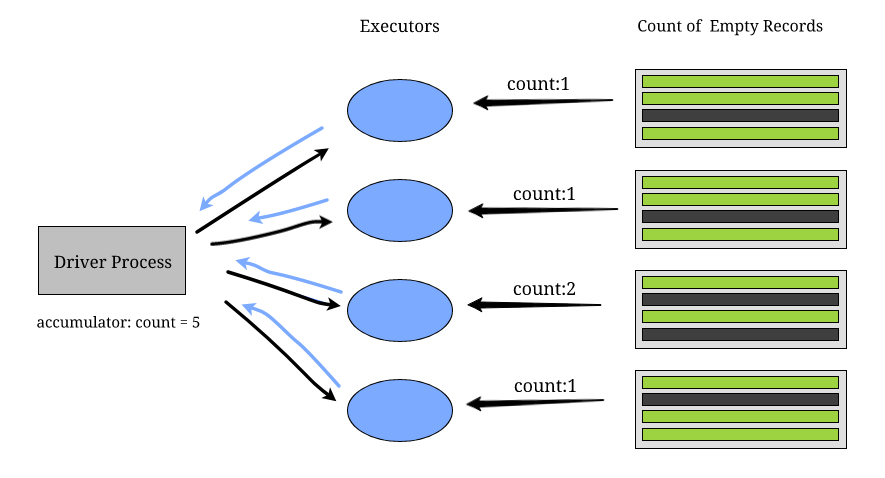
ตัวอย่างเช่นสมมติว่าแอปพลิเคชัน Spark ของเราทำงานใน 100 กลุ่มที่แตกต่างกันซึ่งจับภาพ Instagram ที่โพสต์โดยผู้คนจากประเทศต่างๆ เราต้องนับแท็กเฉพาะที่กล่าวถึงในโพสต์

ตอนนี้ผู้ดำเนินการของแต่ละคลัสเตอร์จะคำนวณผลลัพธ์ของข้อมูลที่มีอยู่ในคลัสเตอร์นั้น ๆ แต่เราต้องการบางสิ่งที่ช่วยให้กลุ่มเหล่านี้สื่อสารกันเพื่อที่เราจะได้รับผลสรุปรวม ใน Spark เรามีตัวแปรที่แชร์ซึ่งทำให้เราสามารถแก้ไขปัญหานี้ได้

**5. Accumulator Variable**

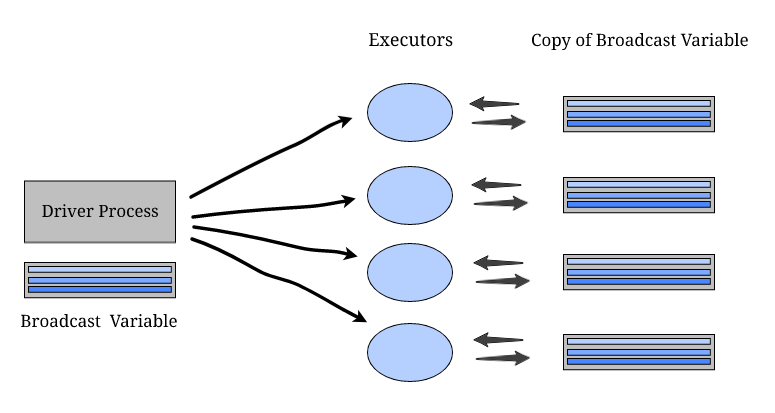
ใช้กรณีเช่นจำนวนครั้งที่เกิดข้อผิดพลาดจำนวนบันทึกว่างเปล่าจำนวนครั้งที่เราได้รับการร้องขอจากประเทศใดประเทศหนึ่ง ทั้งหมดนี้สามารถแก้ไขได้โดยใช้ตัวสะสม

ตัวจัดการบนแต่ละคลัสเตอร์จะส่งข้อมูลกลับไปยังกระบวนการของไดรเวอร์เพื่ออัพเดทค่าของตัวแปรตัวสะสม การสะสมใช้เฉพาะกับการดำเนินการที่เชื่อมโยงและสลับกัน ตัวอย่างเช่นผลรวมและสูงสุดจะทำงานในขณะที่ค่าเฉลี่ยจะไม่



**6.Broadcast Variable**

Broadcast Variable อนุญาตให้โปรแกรมเมอร์เก็บตัวแปรแคชแบบอ่านอย่างเดียวในแต่ละเครื่อง โดยปกติ Spark จะกระจาย Broadcast Variable โดยอัตโนมัติโดยใช้อัลกอริธึมการ Broadcast ที่มีประสิทธิภาพ แต่เรายังสามารถกำหนดได้หากเรามีงานที่ต้องใช้ข้อมูลเดียวกันหลายขั้นตอน



**Understanding the Problem Statement**

งานคือการจัดหมวดหมู่ Tweets แบ่งแยกเชื้อชาติหรือกีดกันทางเพศจากทวีตอื่น ๆ เราจะใช้ข้อมูลตัวอย่างการฝึกอบรมของ Tweets และป้ายกำกับโดยที่ป้ายกำกับ '1' แสดงว่าทวีตนั้นเป็นชนชั้นเหยียดผิว / เพศหญิงและป้ายกำกับ '0'

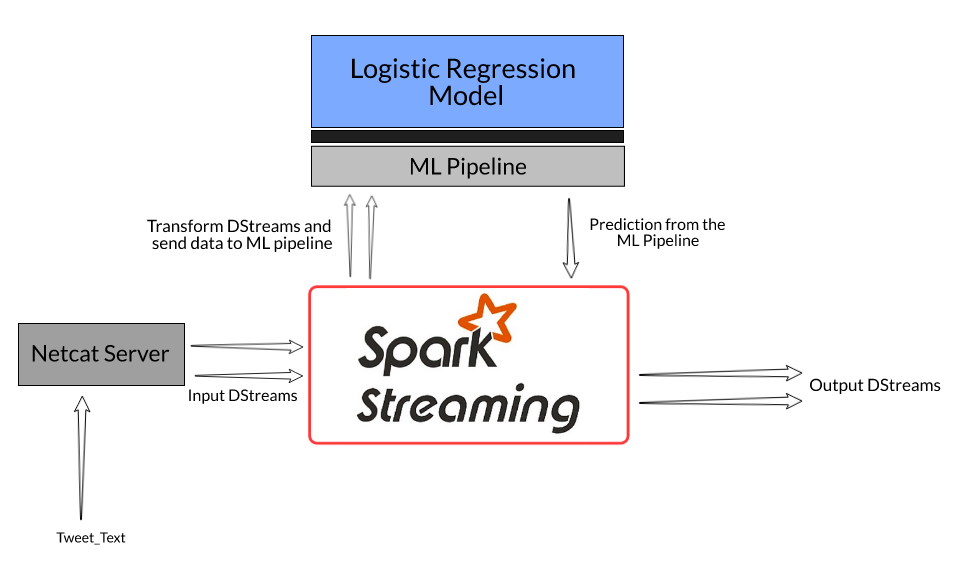
**Setting up the Project Workflow**

1.Model Building:  เราจะสร้าง Logistic Regression Model pipeline เพื่อจำแนกว่าทวีตมีคำพูดแสดงความเกลียดชังหรือไม่ ที่นี่เรามุ่งเน้นที่จะไม่สร้างรูปแบบการจัดประเภทที่แม่นยำมาก แต่เพื่อดูวิธีการใช้แบบจำลองใด ๆ และส่งคืนผลข้อมูลการสตรีม

2.Initialize Spark Streaming Context: เมื่อสร้างแบบจำลองแล้วเราจำเป็นต้องกำหนดชื่อโฮสต์และหมายเลขพอร์ตจากตำแหน่งที่เรารับข้อมูลสตรีม

3.Stream Data: ต่อไปเราจะเพิ่มทวีตจากเซิร์ฟเวอร์ netcat จากพอร์ตที่กำหนดและ Spark Streaming API จะได้รับข้อมูลหลังจากระยะเวลาที่กำหนด

4.Predict and Return Results: เมื่อเราได้รับข้อความ Tweets เราจะส่งข้อมูลไปยังขั้นตอนการเรียนรู้ของเครื่องที่เราสร้างขึ้นและส่งคืนความรู้สึกที่คาดการณ์จากแบบจำลอง

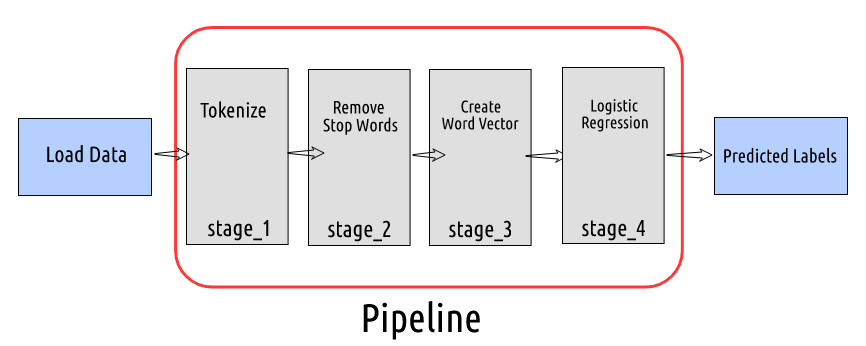


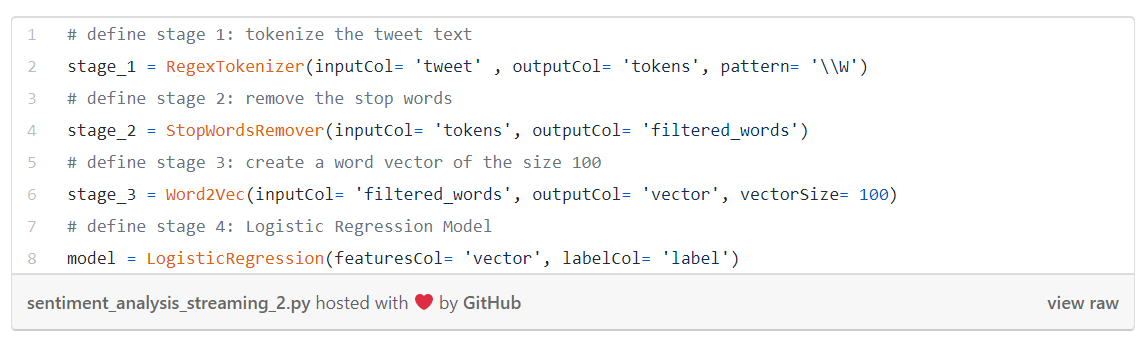
**Defining the Stages of our Machine Learning Pipeline**

ตอนนี้เรามีข้อมูลใน Spark dataframe แล้วเราต้องกำหนด stage ต่าง ๆ ที่เราต้องการแปลงข้อมูลจากนั้นใช้มันเพื่อรับ label ที่ถูกทำนายจากโมเดลของเรา

ใน stage แรกเราจะใช้ RegexTokenizer เพื่อแปลงข้อความ Tweets เป็นรายการของคำ จากนั้นเราจะลบคำหยุดออกจาก word list และ [word vectors](https://www.analyticsvidhya.com/blog/2019/07/how-to-build-recommendation-system-word2vec-python/?utm_source=blog&utm_medium=streaming-data-pyspark-machine-learning-model) ในขั้นตอนสุดท้ายเราจะใช้คำว่าเวกเตอร์เหล่านี้เพื่อสร้างแบบจำลองการถดถอยโลจิสติกส์(logistic regression model)และรับความรู้สึกที่ทำนายไว้

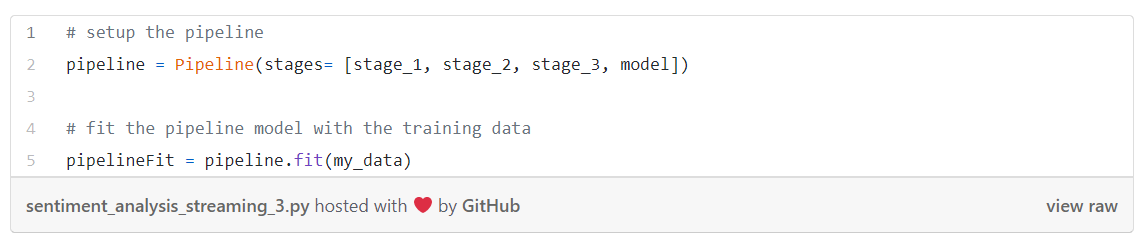
จำไว้ว่า - เราไม่ได้มุ่งเน้นที่การสร้างรูปแบบการจำแนกที่แม่นยำมาก แต่เพื่อดูว่าเราจะใช้ตัวแบบการทำนายเพื่อรับผลลัพธ์จากการส่งข้อมูล





**Setup our Machine Learning Pipeline**

เพิ่มขั้นตอนในวัตถุ Pipeline  แล้วเราจะทำการแปลงตามลำดับ ติดตั้ง Pipeline  กับชุดข้อมูลการฝึกอบรมและตอนนี้เมื่อใดก็ตามที่เรามี Tweets ใหม่เราเพียงแค่ต้องส่งผ่านวัตถุ Tweets และแปลงข้อมูลเพื่อรับการคาดการณ์



**Stream Data and Return Results**

สมมติว่าเราได้รับความคิดเห็นนับร้อยต่อวินาทีและเราปิดกั้นผู้ใช้ที่โพสต์ความคิดเห็นที่มีคำพูดแสดงความเกลียดชัง ดังนั้นเมื่อใดก็ตามที่เราได้รับข้อความใหม่เราจะส่งสิ่งนั้นเข้าไปใน Pipeline  และรับความรู้สึกที่คาดการณ์ไว้ เราจะกำหนดฟังก์ชั่น get\_prediction ซึ่งจะลบประโยคว่างและสร้างดาต้าเฟรมที่แต่ละแถวมีทวีต ดังนั้นเริ่มต้นบริบท Spark Streaming และกำหนดระยะเวลาแบทช์ 3 วินาที ซึ่งหมายความว่าเราจะทำการคาดการณ์ข้อมูลที่เราได้รับทุก 3 วินาที:



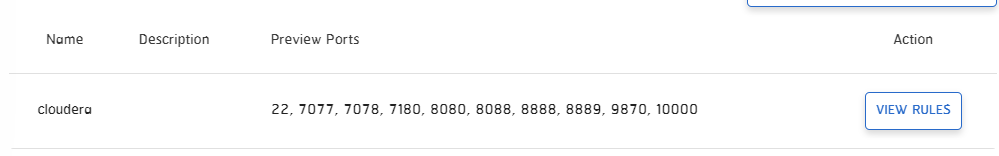
รันโปรแกรมในเทอร์มินัลเดียวและใช้ Netcat (เครื่องมือยูทิลิตี้ที่สามารถใช้ในการส่งข้อมูลไปยังชื่อโฮสต์และหมายเลขพอร์ตที่กำหนด) คุณสามารถเริ่มการเชื่อมต่อ TCP โดยใช้คำสั่งนี้:

nc -lk port\_number

**ขอบคุณข้อมูลจาก website**

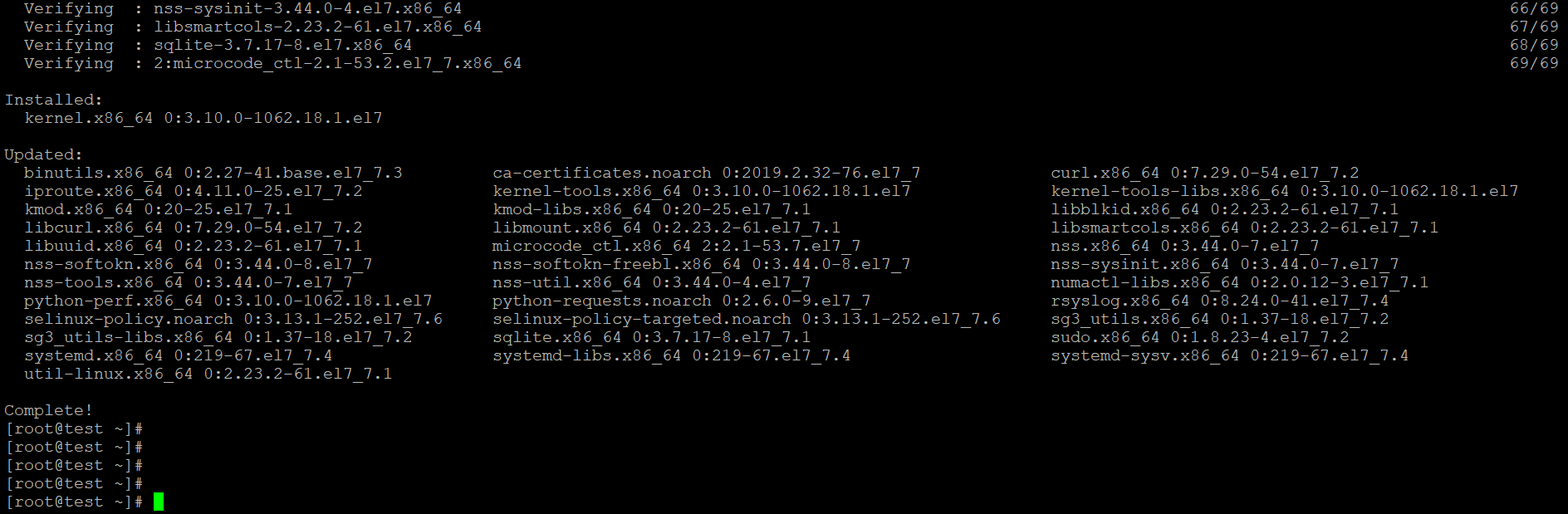
<https://www.analyticsvidhya.com/blog/2019/12/streaming-data-pyspark-machine-learning-model/>

Remark: after you create instance don’t forget Manage Security Group in openlandscape

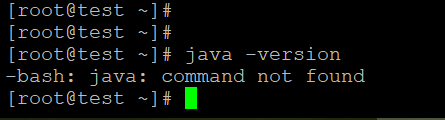


**Installing Java 8 on CentOS 7**

yum -y update

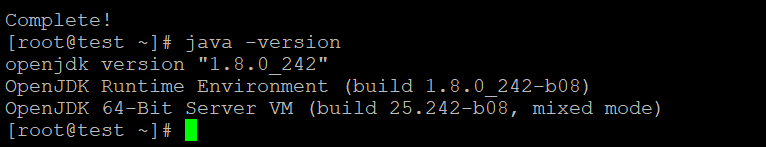


java -version



yum install java-1.8.0-openjdk

java -version



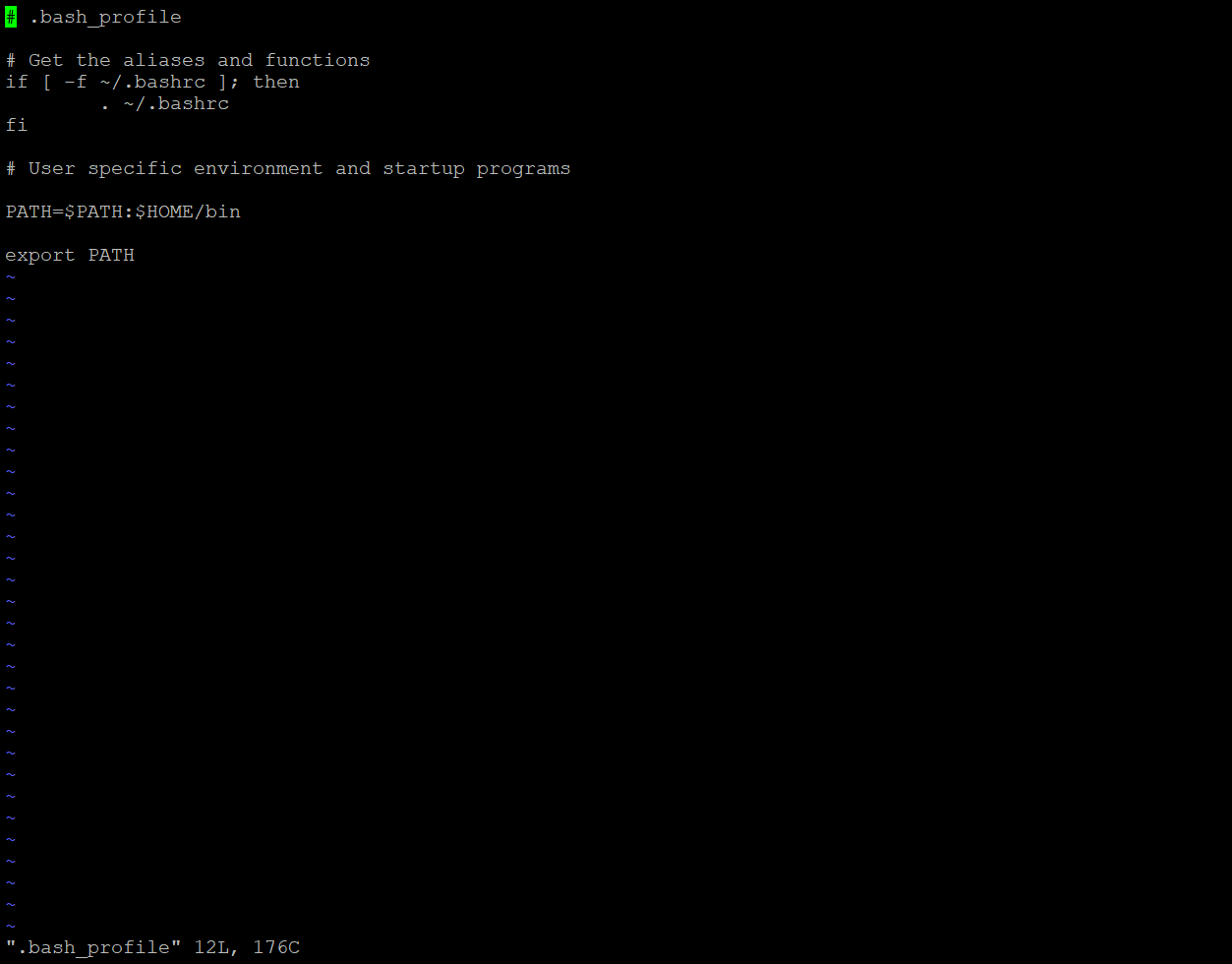
**Set Java’s Home Environment**

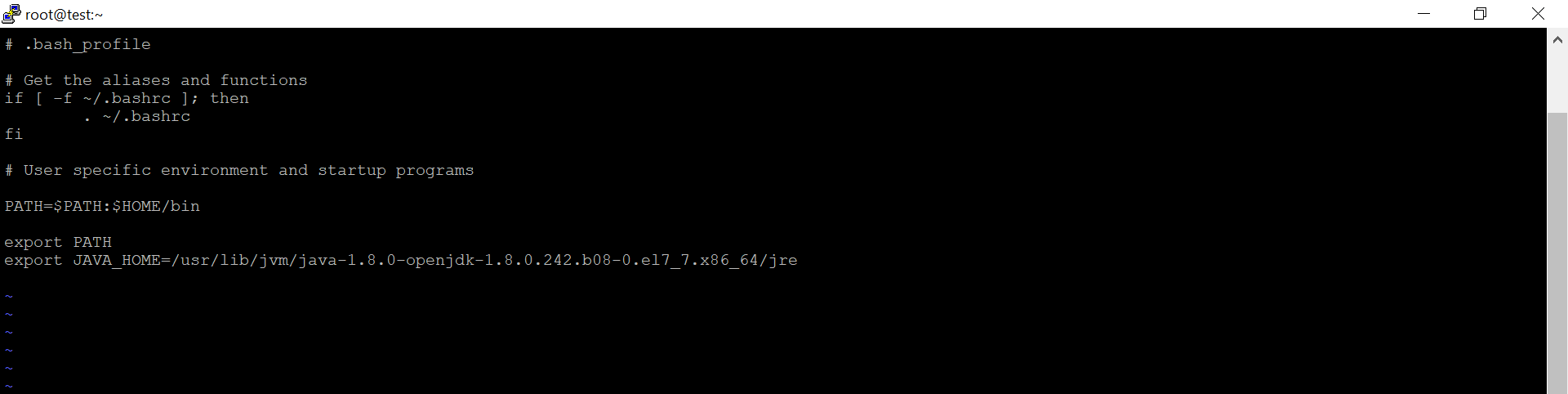
update-alternatives --config java

## 

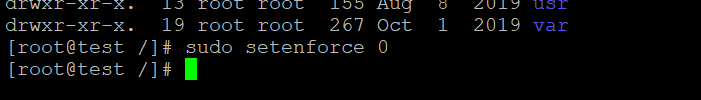
export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.242.b08-0.el7\_7.x86\_64/jre

vi .bash\_profile

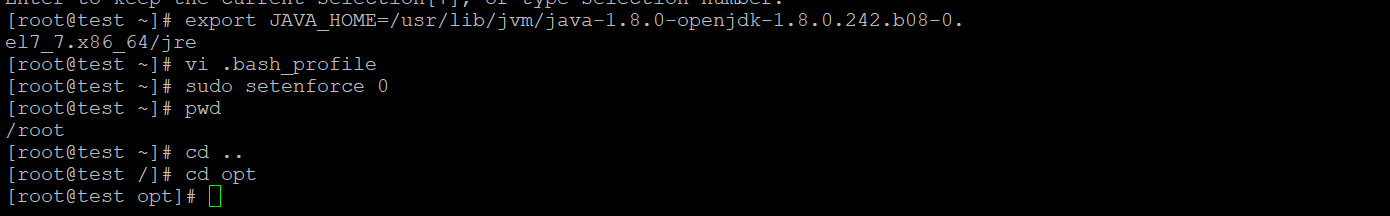




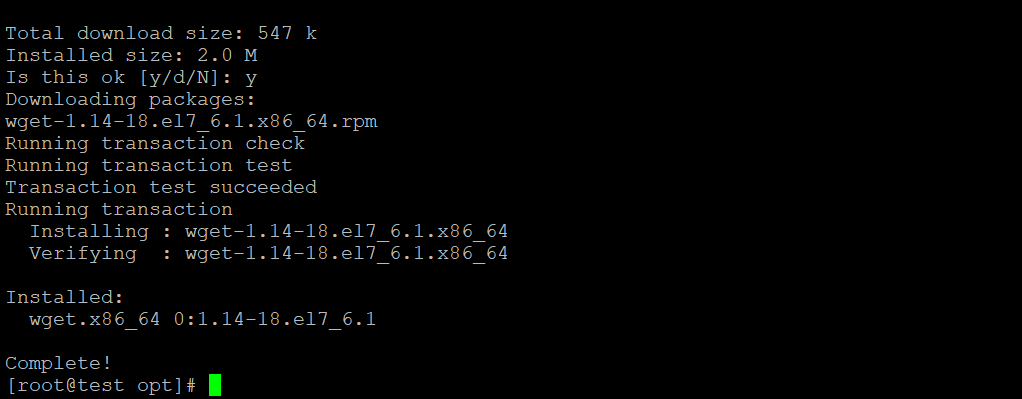
sudo setenforce 0 (**Disable SELinux)**



cd opt

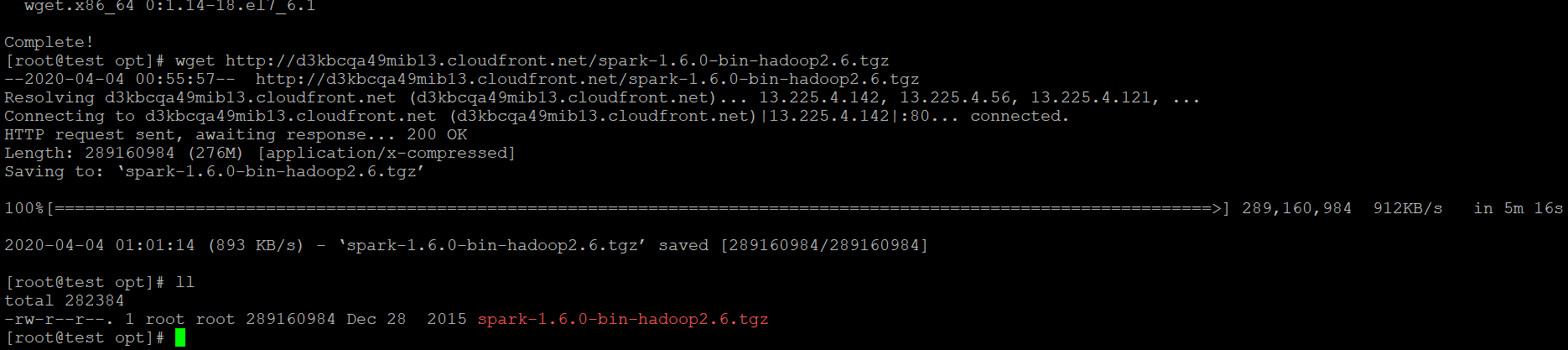


yum install wget

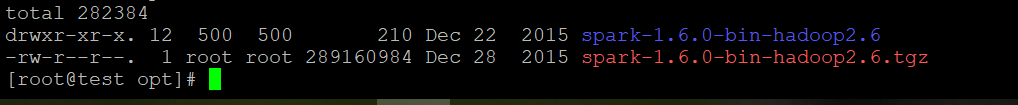


wget http://d3kbcqa49mib13.cloudfront.net/spark-1.6.0-bin-hadoop2.6.tgz

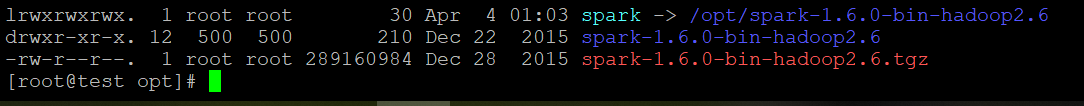




tar -xzf spark-1.6.0-bin-hadoop2.6.tgz



ln -s /opt/spark-1.6.0-bin-hadoop2.6 /opt/spark

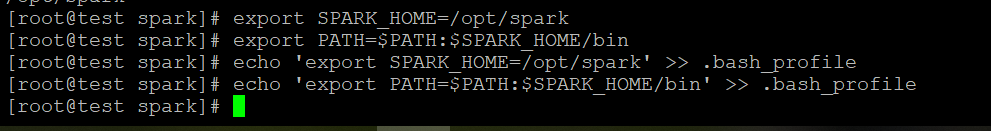


export SPARK\_HOME=/opt/spark

export PATH=$PATH:$SPARK\_HOME/bin

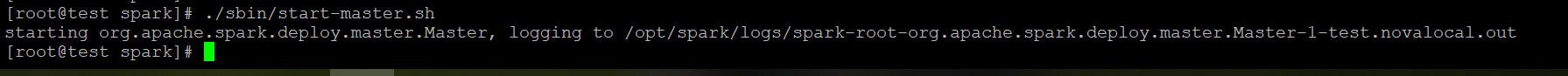
echo 'export SPARK\_HOME=/opt/spark' >> .bash\_profile

echo 'export PATH=$PATH:$SPARK\_HOME/bin' >> .bash\_profile



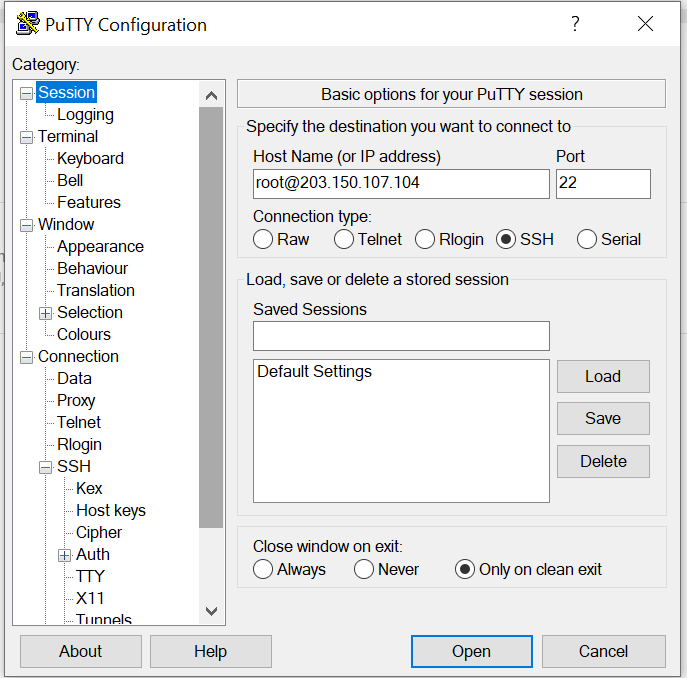
cd spark

./sbin/start-master.sh

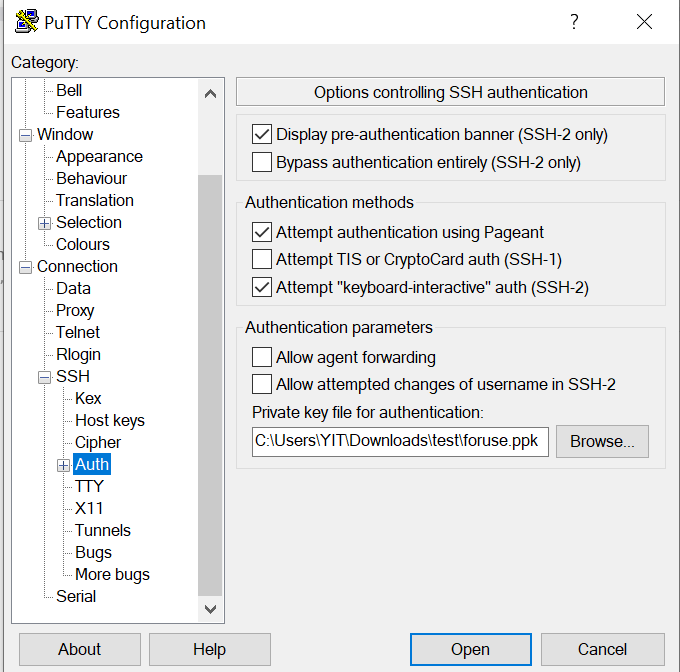


**putty tunnels**

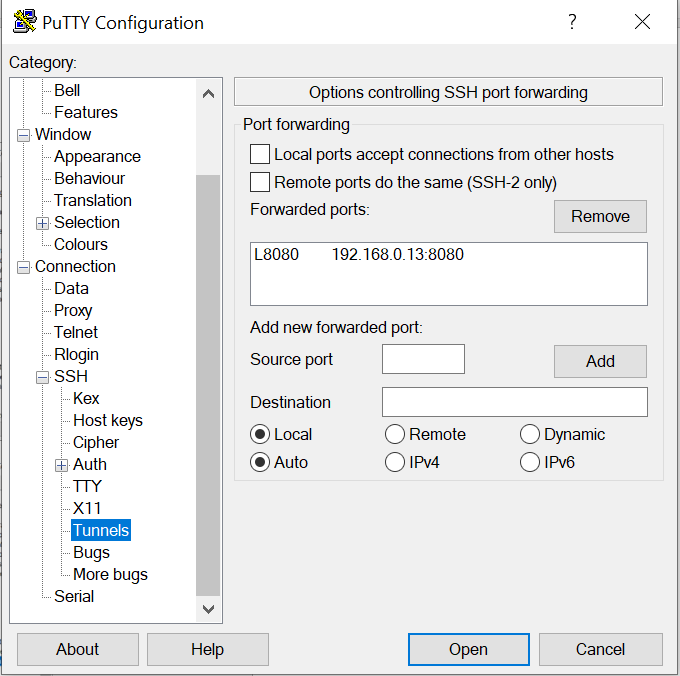
1.Set Public Network ip:



2.Put Public key

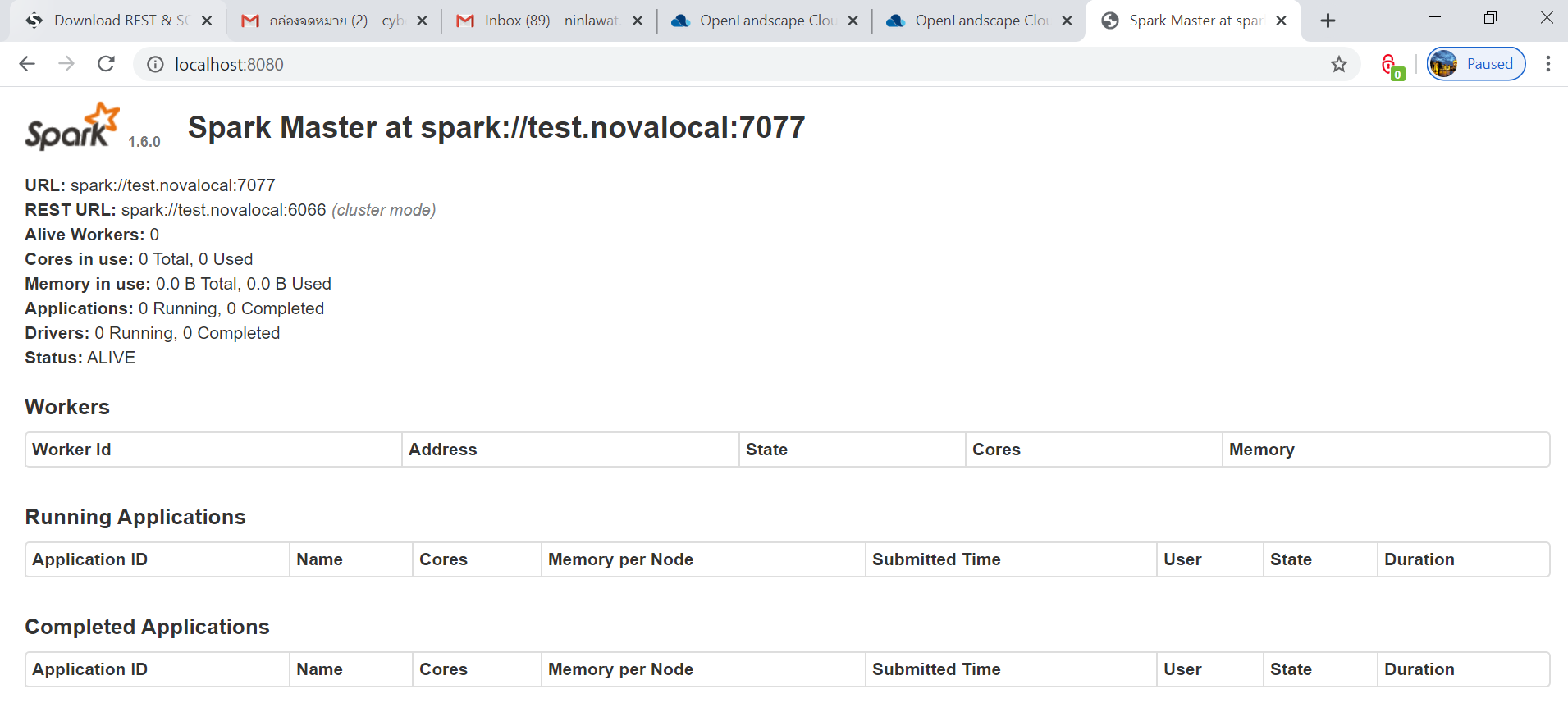


3.Destination : 192.168.0.13:8080,Source port:8080 and then press button Add



Addition explain <https://chamibuddhika.wordpress.com/2012/03/21/ssh-tunnelling-explained/>

<http://localhost:8080/>



Check: $JAVA\_HOME, $PATH

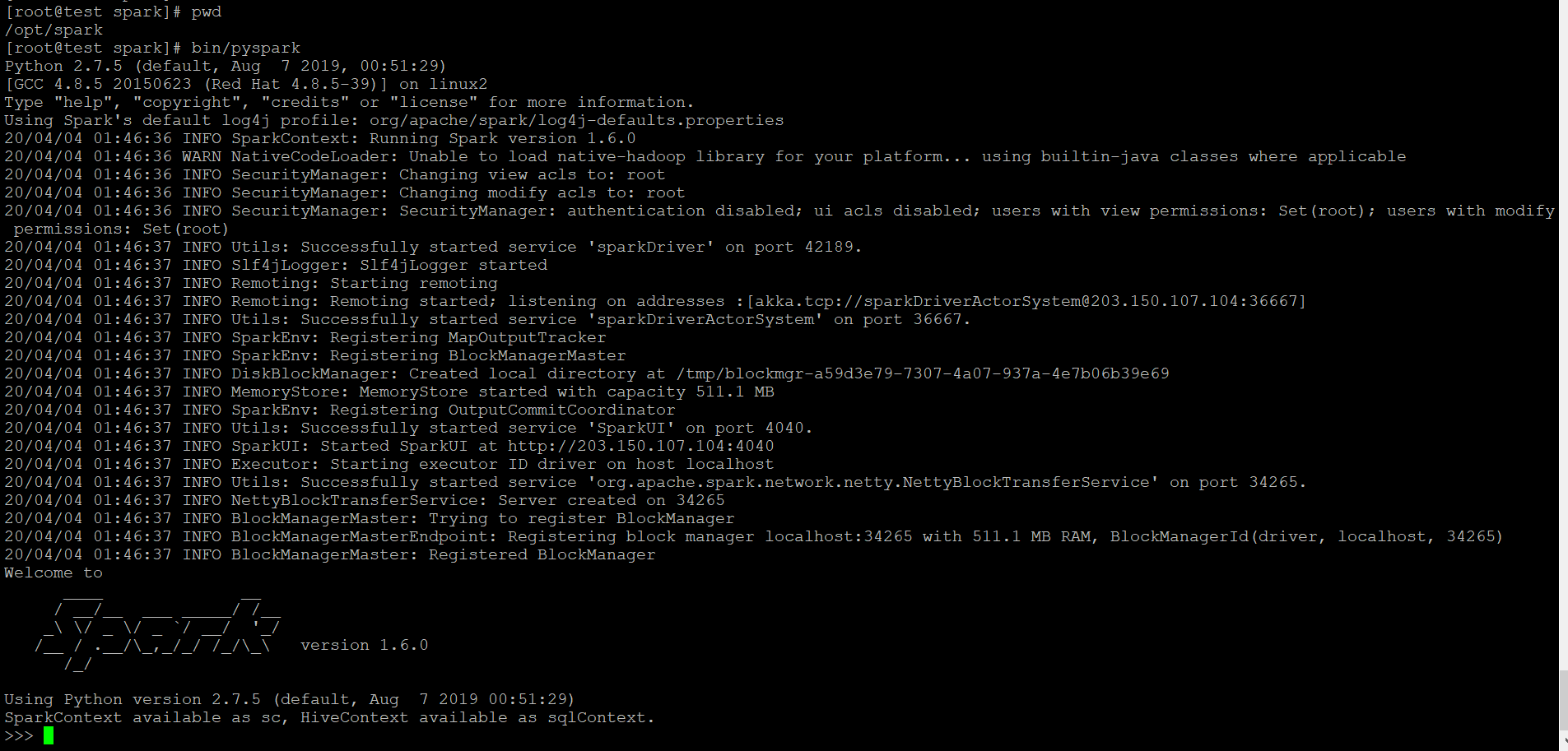
[root@test spark]# echo $JAVA\_HOME

/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.242.b08-0.el7\_7.x86\_64/jre

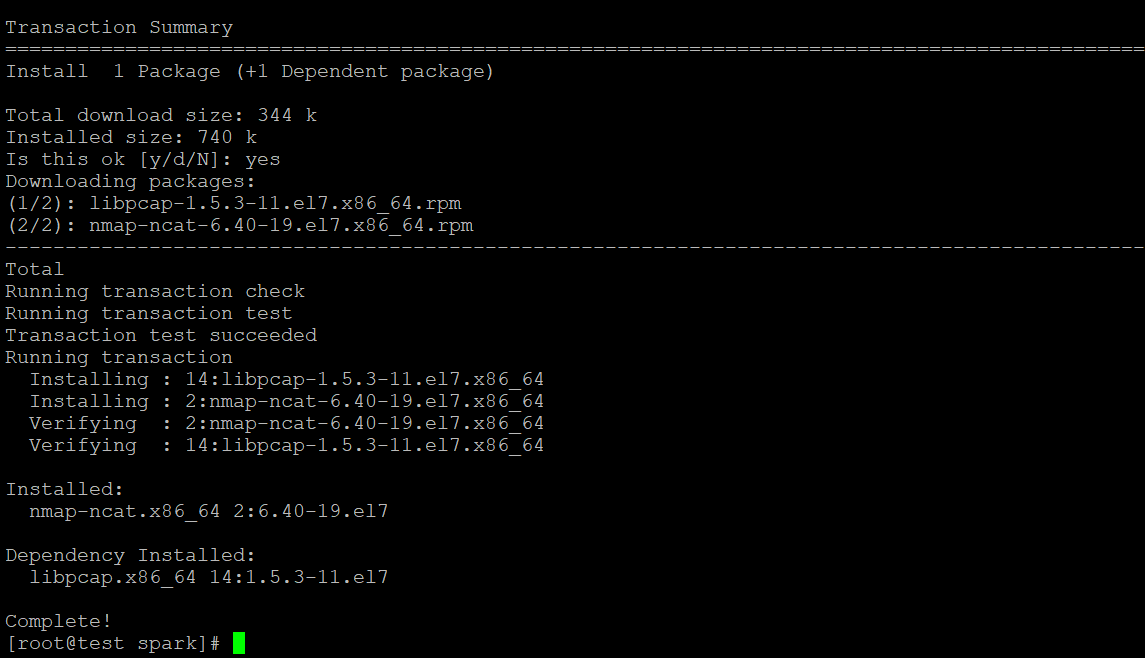
[root@test spark]# echo $PATH

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin:/bin:/bin:/bin

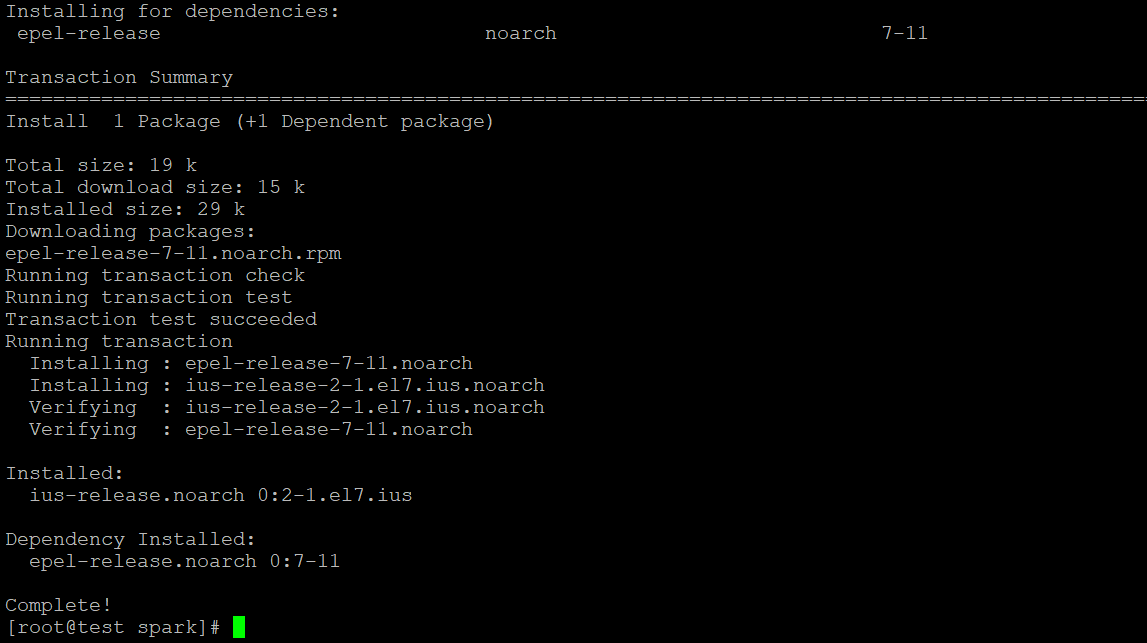
bin/pyspark



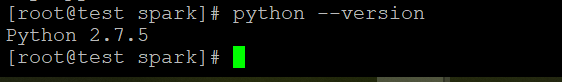
yum install nc



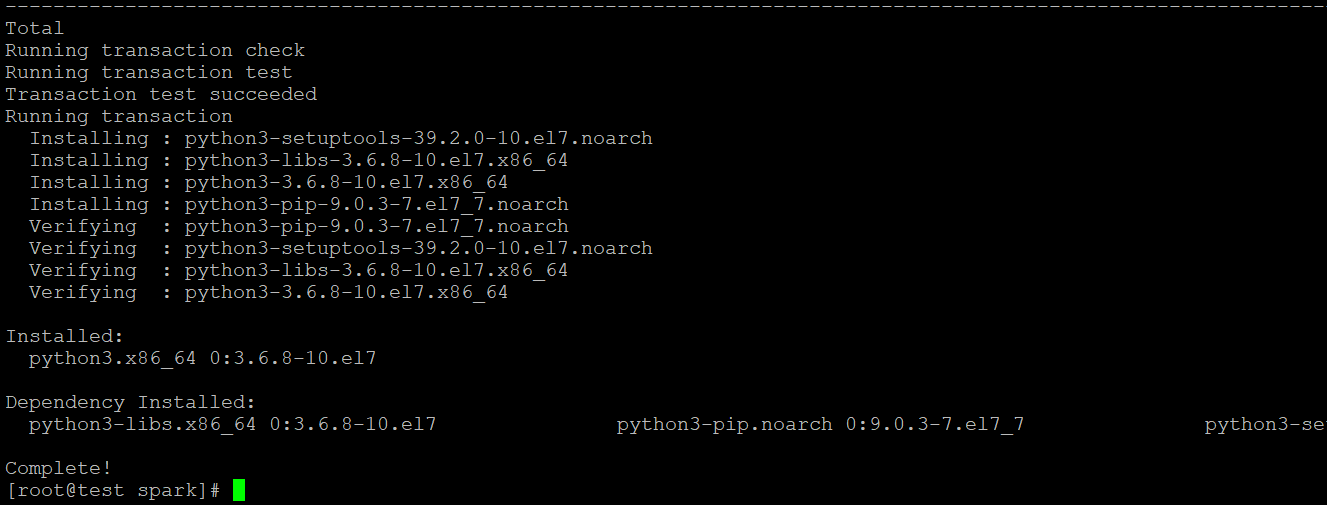
yum install -y https://centos7.iuscommunity.org/ius-release.rpm



python --version



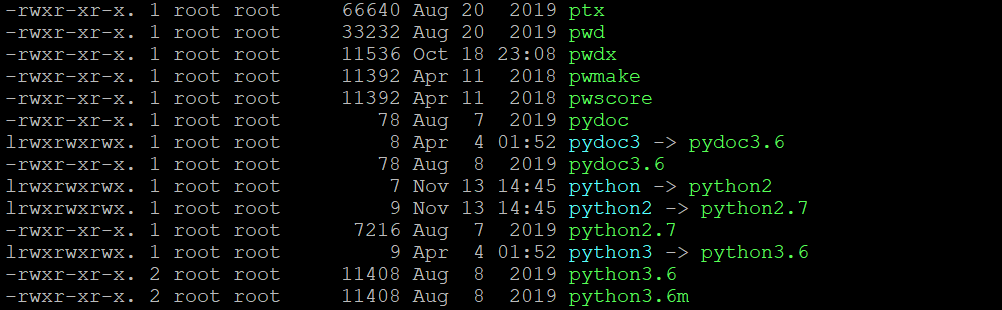
yum install -y python36u



which python



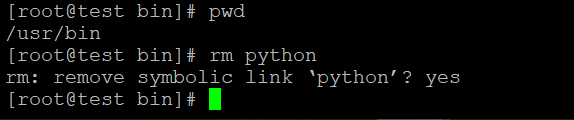
Go to this path ‘usr/bin/python’



You see python -> python2.7 old version need to change to python3.6

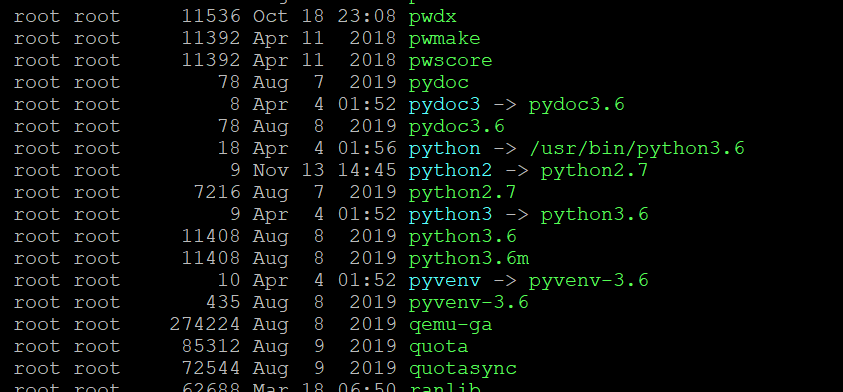
Please following below picture

rm python



ln -s /usr/bin/python3.6 /usr/bin/python

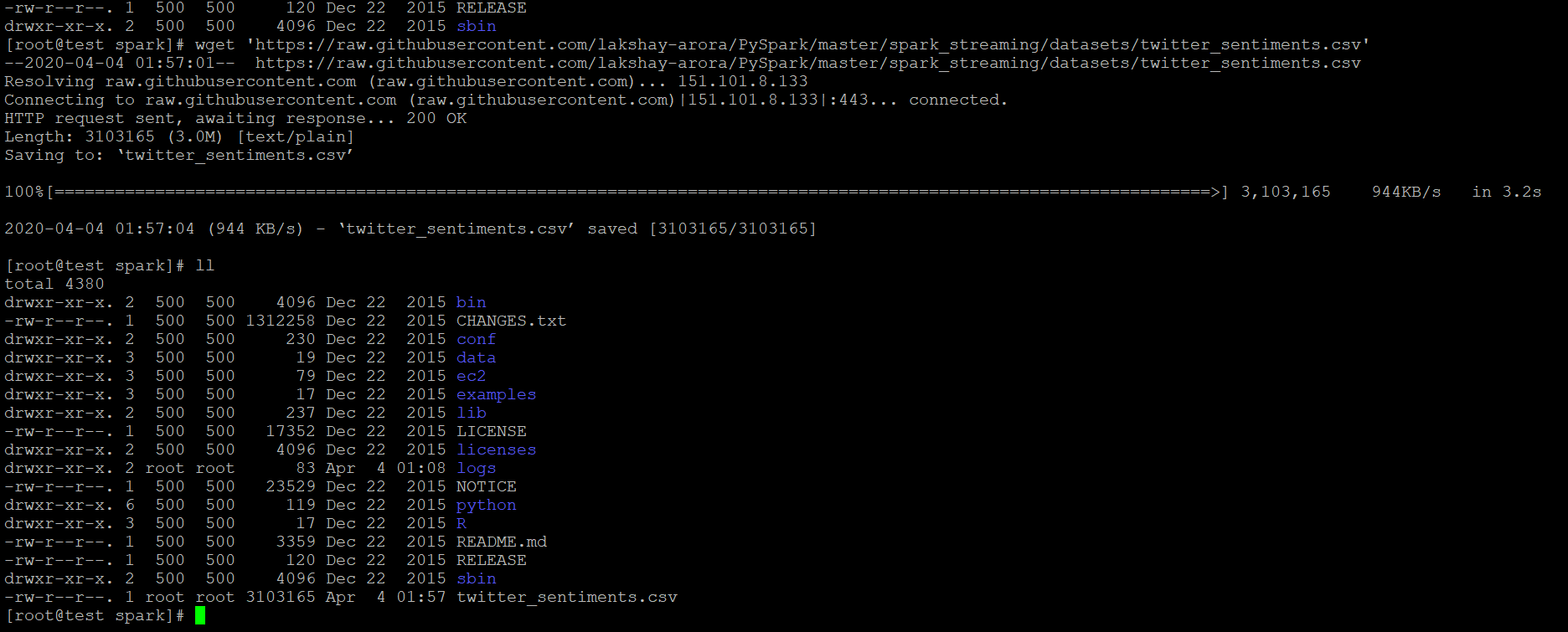
After you run command above. You can return to check to get the result as in the picture



Get dataset :

Goto path: /opt/spark

wget 'https://raw.githubusercontent.com/lakshay-arora/PySpark/master/spark\_streaming/datasets/twitter\_sentiments.csv'

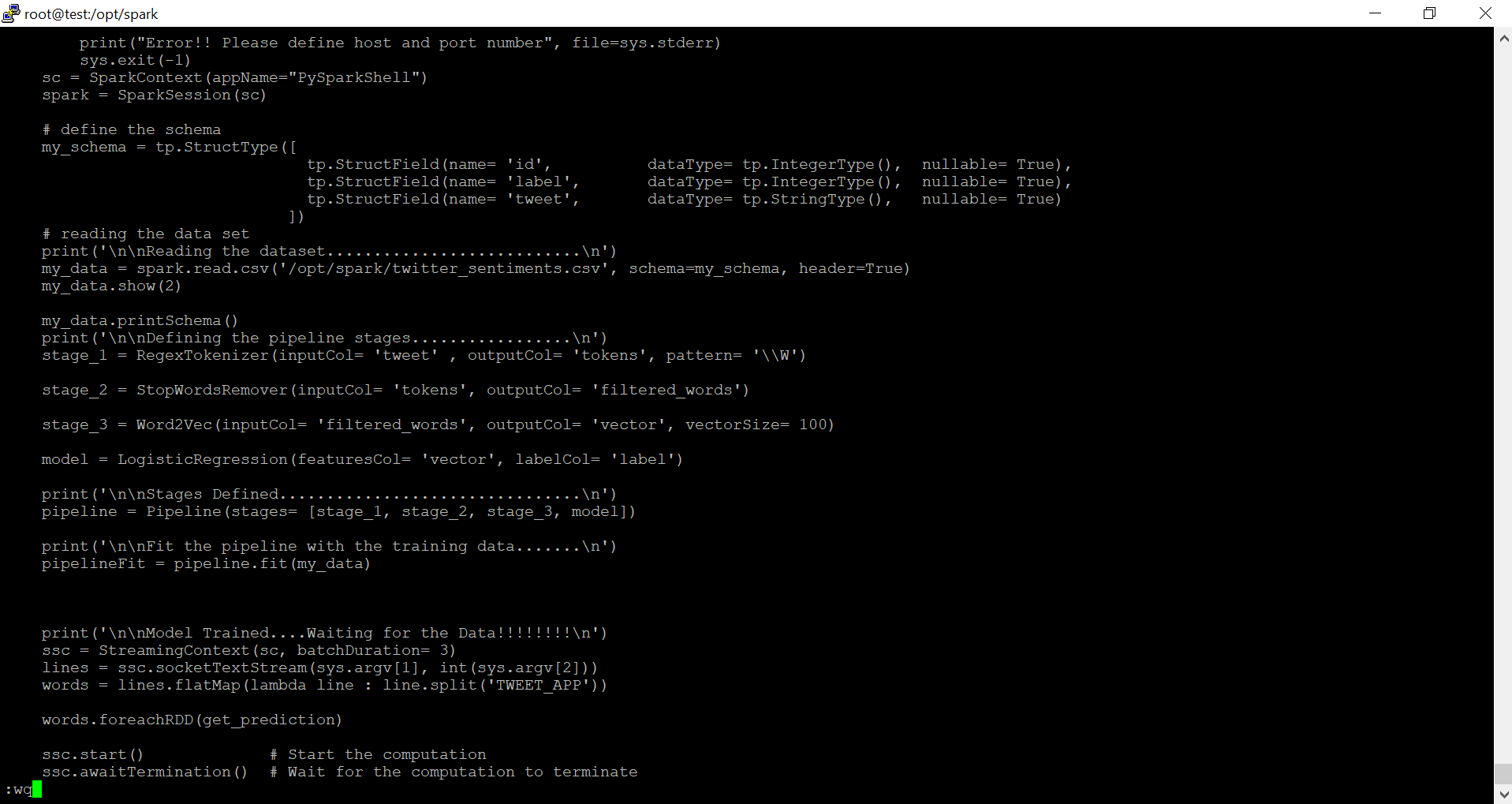


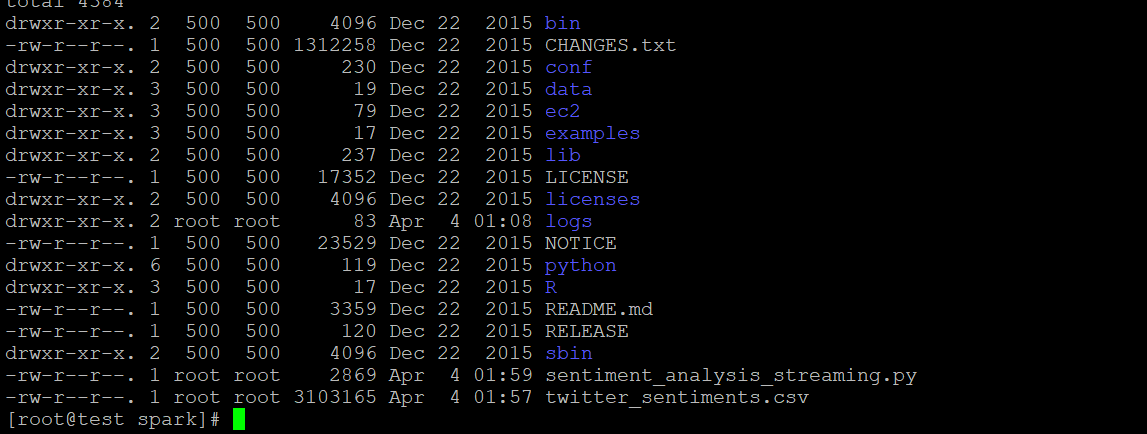
Create file python sentiment\_analysis\_streaming.py

I copy code python from

<https://github.com/lakshay-arora/PySpark/blob/master/spark_streaming/twitter_sentiment_analysis.py>

and then you use command: vi sentiment\_analysis\_streaming.py pass code python in this

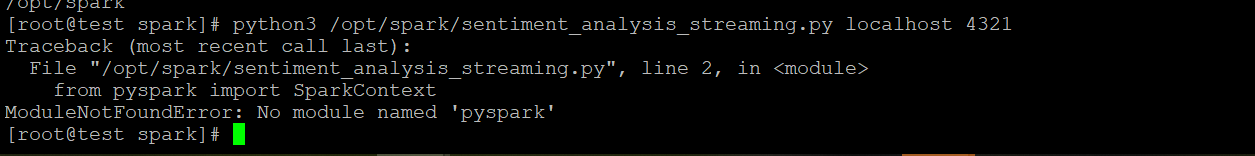




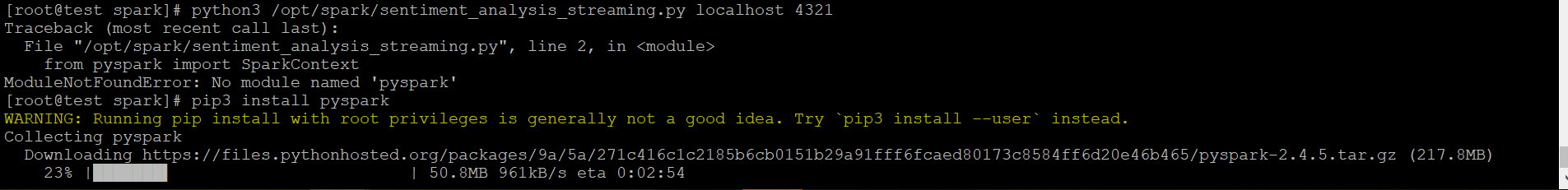
How to run command:

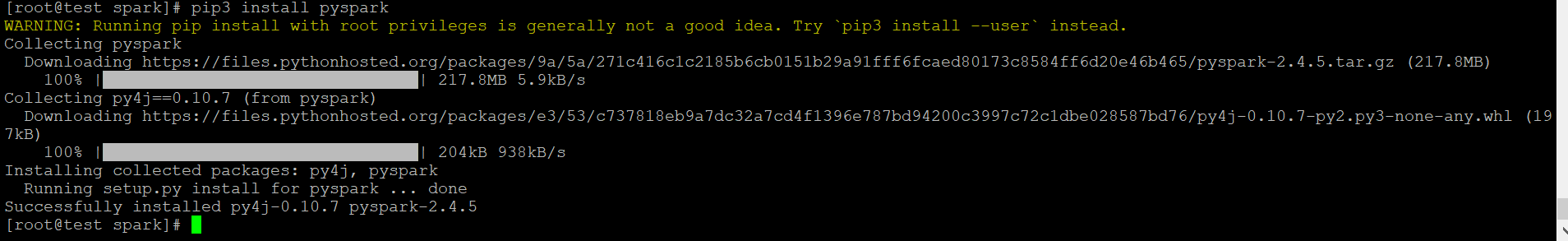
Pattern: python3 host Port

python3 /opt/spark/sentiment\_analysis\_streaming.py localhost 4321

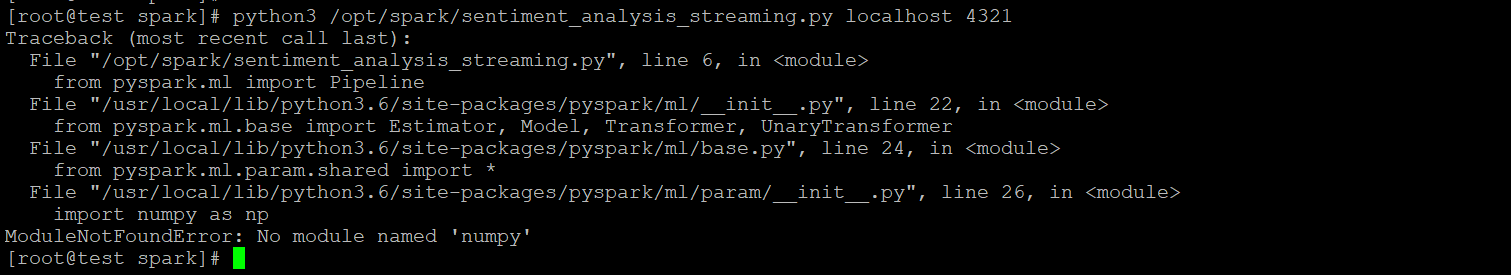


pip3 install pyspark

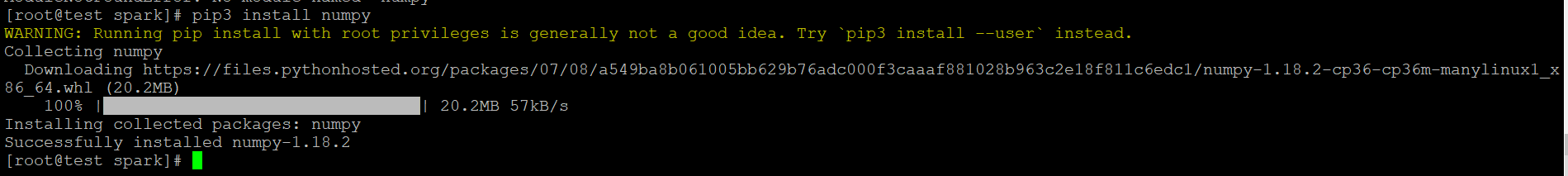


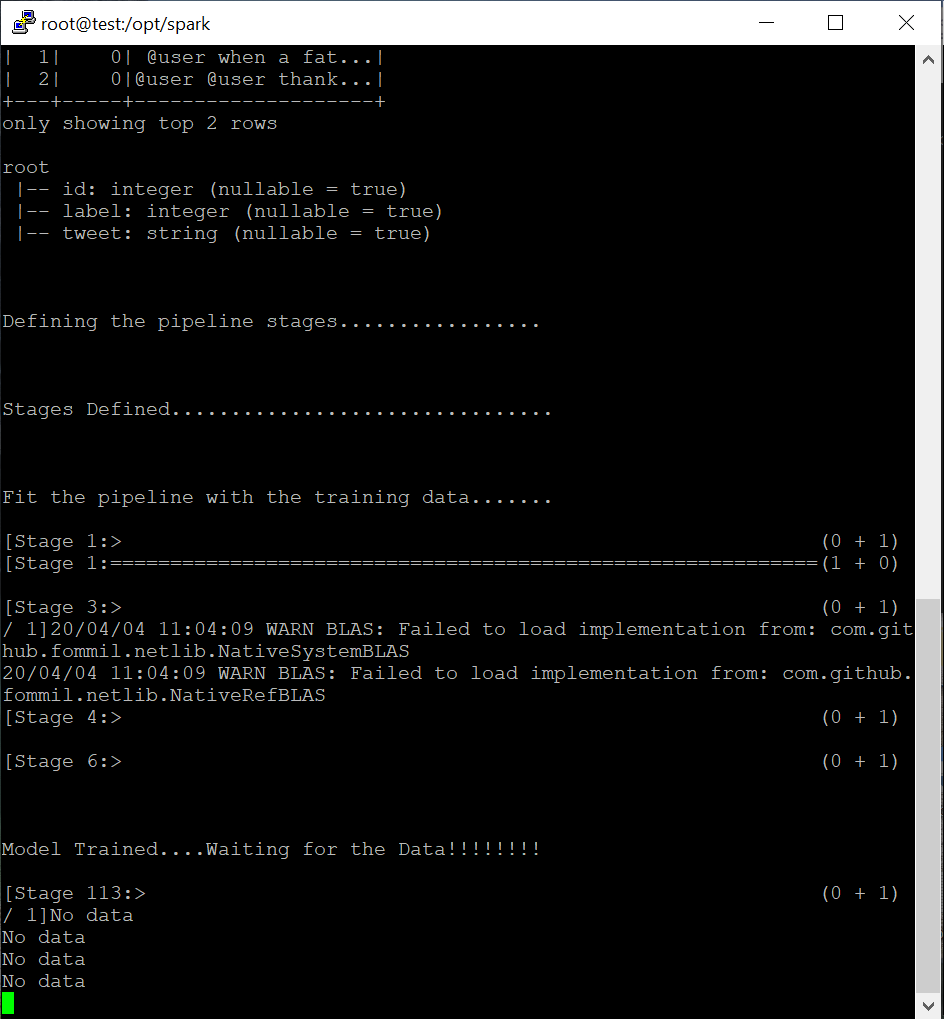


python3 /opt/spark/sentiment\_analysis\_streaming.py localhost 4321

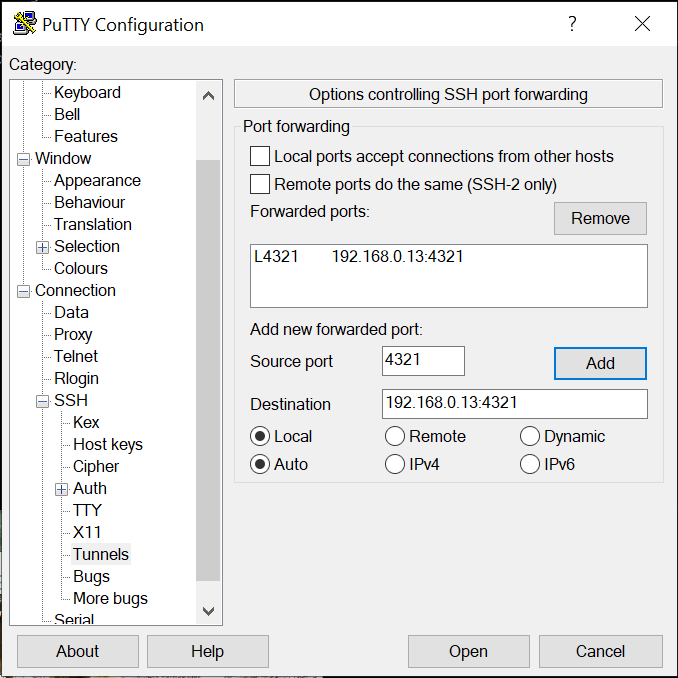


pip3 install numpy



python3 /opt/spark/sentiment\_analysis\_streaming.py 203.150.107.83 8080

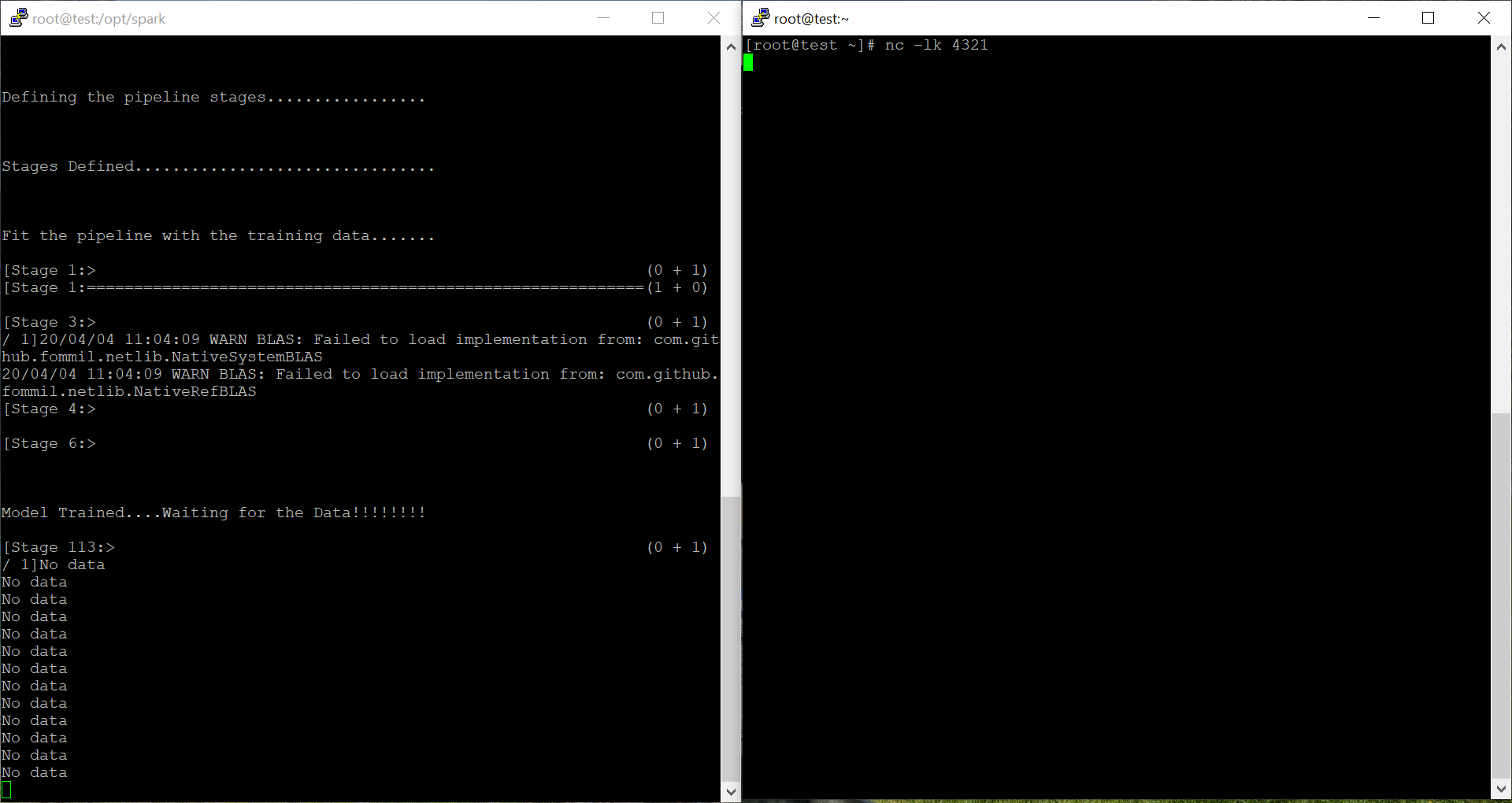
before test you need to tunnels port 4321 one session hold order to can see massage from netcat



You need to open session two and session three

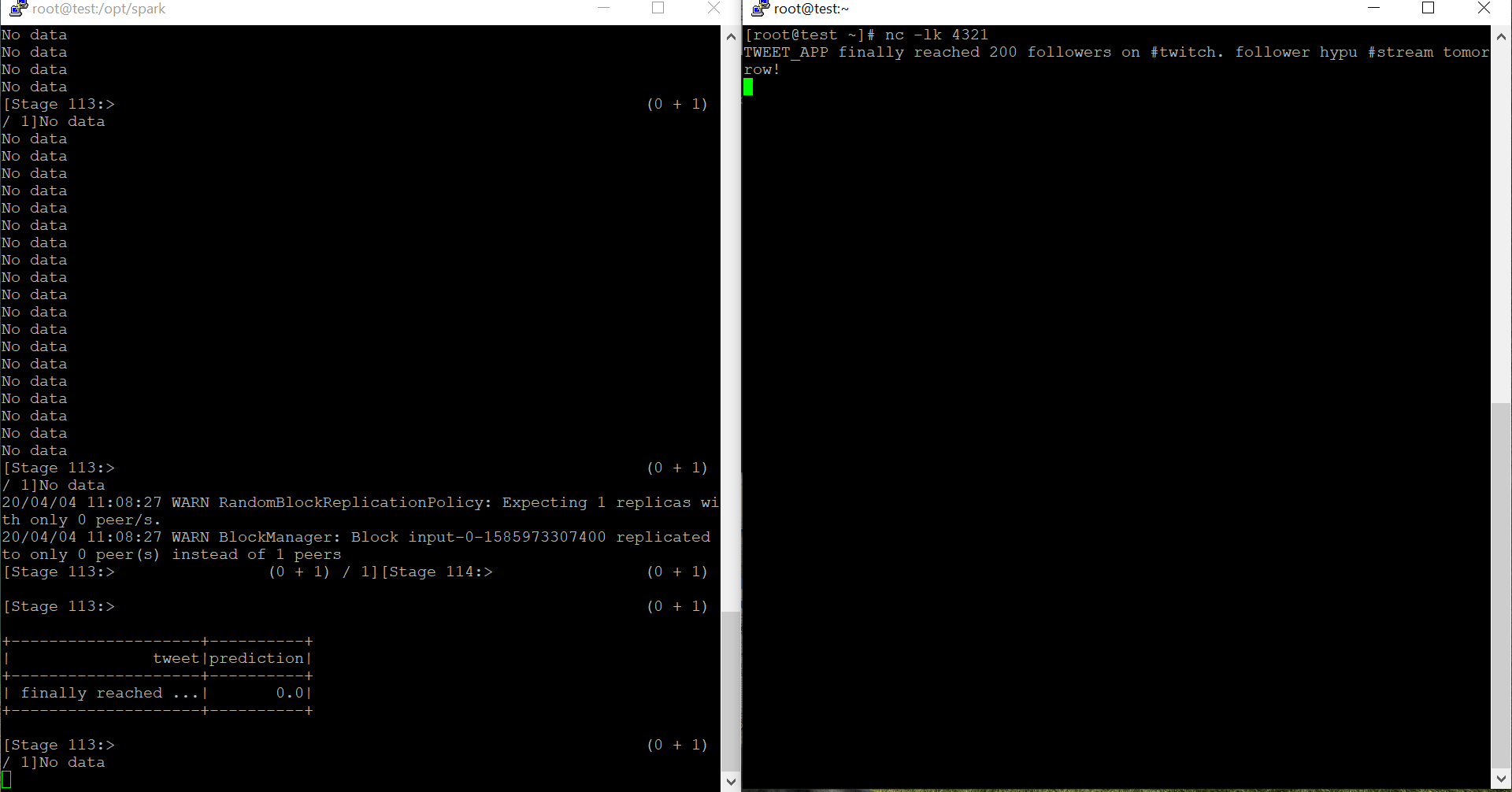
session two right side run command: nc –lk 4321

session three left side run command: python3 /opt/spark/sentiment\_analysis\_streaming.py localhost 4321



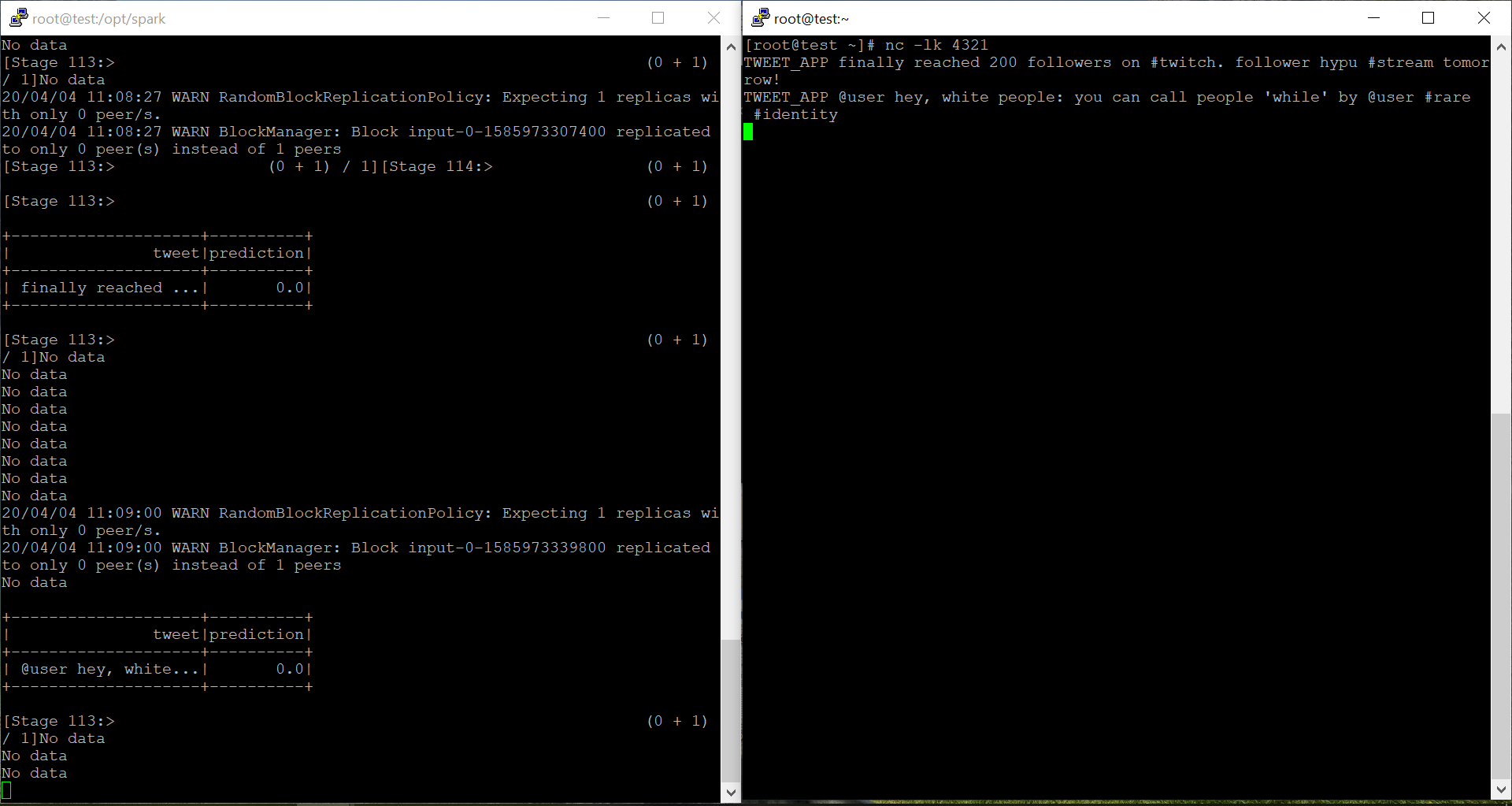
Try send massage:

TWEET\_APP finally reached 200 followers on #twitch. follower hypu #stream tomorrow!



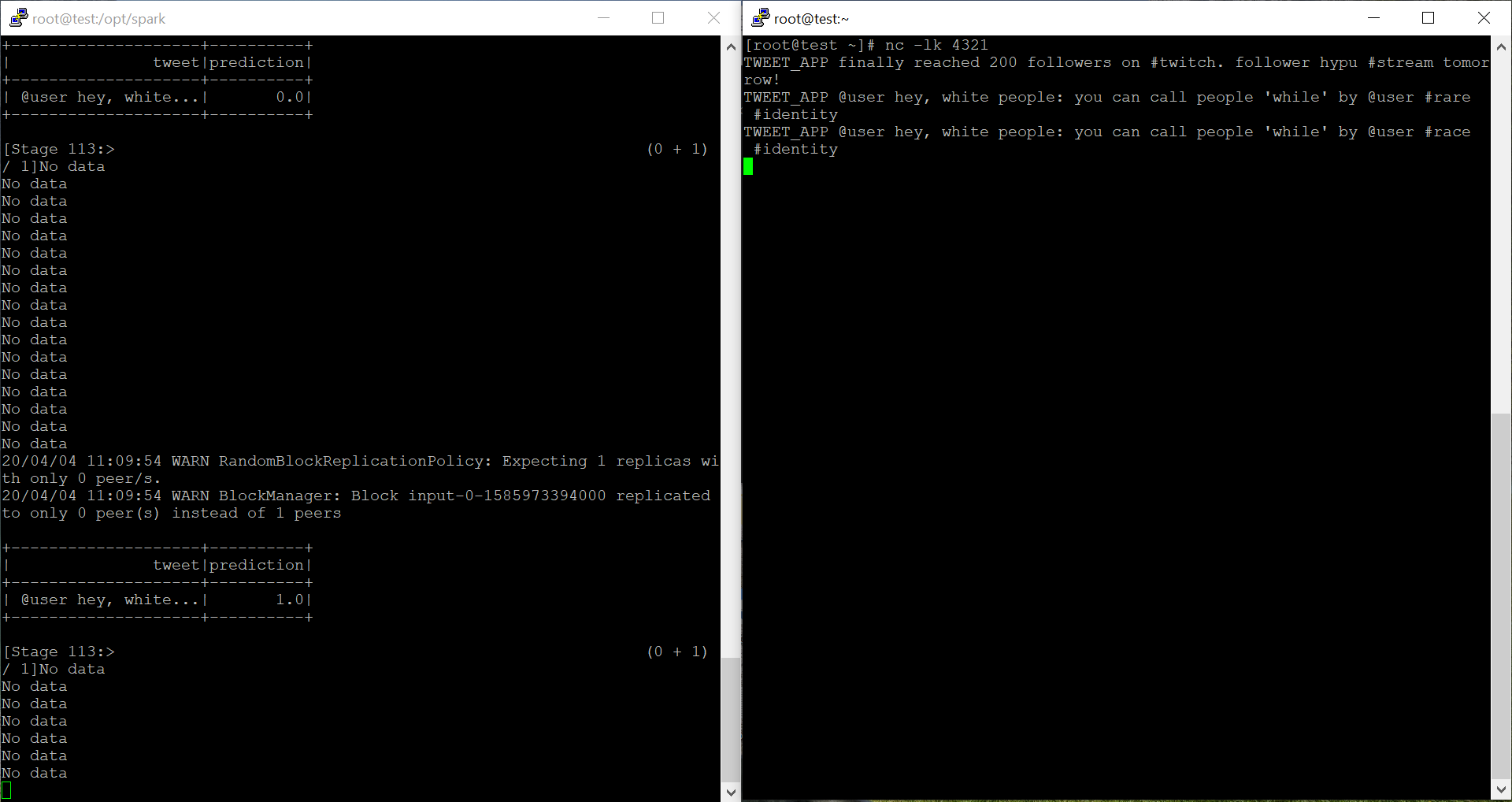
Try send massage:

TWEET\_APP @user hey, white people: you can call people 'while' by @user #rare #identity



Try send massage:

TWEET\_APP @user hey, white people: you can call people 'while' by @user #race #identity



You can get prediction in real-time in the terminal!

I appreciate all the knowledge I got from

<https://www.analyticsvidhya.com/blog/2019/12/streaming-data-pyspark-machine-learning-model/>

<https://linuxize.com/post/install-java-on-centos-7/>

<https://www.liquidweb.com/kb/install-java-8-on-centos-7/>

<http://devopspy.com/python/apache-spark-pyspark-centos-rhel/>

<https://gist.github.com/darcyliu/d47edccb923b0f03280a4cf8b66227c1>