Answers

1. Switch to Linux ec2 user and navigate to home directory

```
Session ID: od[redacted]
sh-4.2$ sudo su ec2-user
[ec2-user@ip-10-10-20-228 bin]$ cd ~
[ec2-user@ip-10-10-20-228 ~]$
```

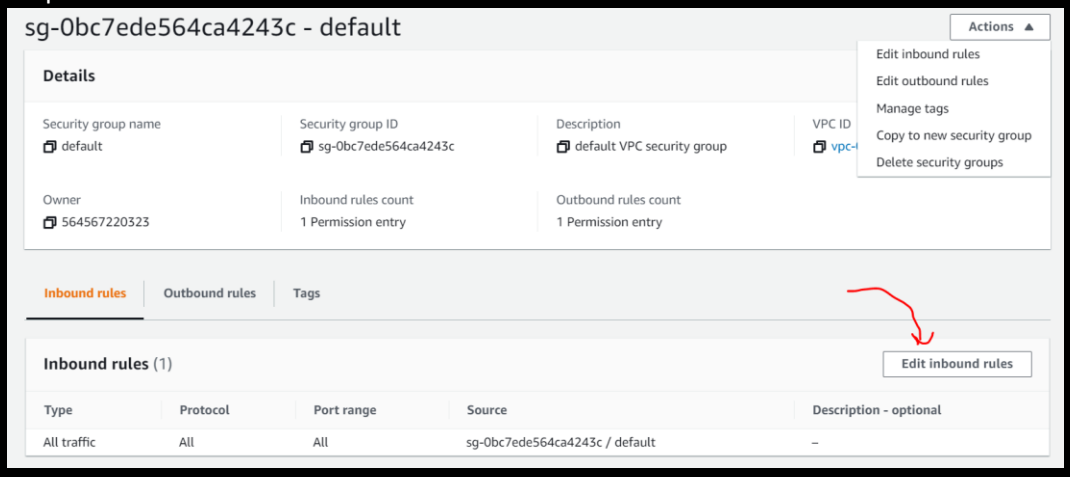
2. Run *<ifconfig>* from master instance to return machines private IP address, run same command on EC2 instance for private IP address of EC2 instance

```
[hadoop@ip-10-10-10-62 bin]$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9001
        inet 10.10.10.62  netmask 255.255.255.0  broadcast 10.10.10.255
        inet6 fe80::10d7:c5ff:fe4d:dc9  prefixlen 64  scopeid 0x20<link>
        ether 12:d7:c5:4d:0d:c9  txqueuelen 1000  (Ethernet)
        RX packets 2950059  bytes 4056470333 (3.7 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 701229  bytes 945721242 (901.9 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 574558  bytes 78131019 (74.5 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 574558  bytes 78131019 (74.5 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

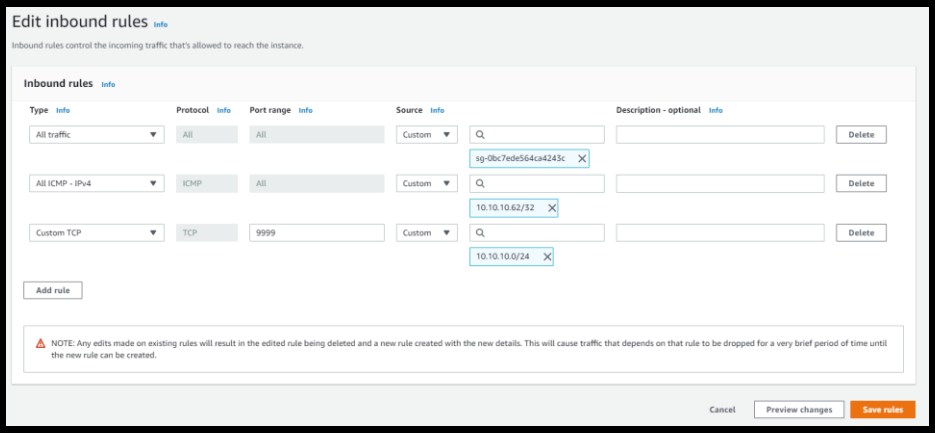
3. Let's test basic connectivity between cluster and EC2 instance using a ping command

```
[hadoop@ip-10-10-10-62 bin]$ ping 10.10.20.228 -c 5
PING 10.10.20.228 (10.10.20.228) 56(84) bytes of data.

--- 10.10.20.228 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4091ms
```

4. Let's fix the reachability by adding rules to the EC2 instances security group (firewall). First we will add rule for ICMP and a second rule to add Custom TCP port number 9999 to cluster range (10.10.10.0/24) or 0.0.0.0/0
    a. Step 1:



b. Step 2:



c. Testing reachability again and checking latency between them

```
[hadoop@ip-10-10-10-62 bin]$ ping 10.10.20.228 -c 5
PING 10.10.20.228 (10.10.20.228) 56(84) bytes of data.
64 bytes from 10.10.20.228: icmp_seq=1 ttl=255 time=0.178 ms
64 bytes from 10.10.20.228: icmp_seq=2 ttl=255 time=0.168 ms
64 bytes from 10.10.20.228: icmp_seq=3 ttl=255 time=0.171 ms
64 bytes from 10.10.20.228: icmp_seq=4 ttl=255 time=0.165 ms
64 bytes from 10.10.20.228: icmp_seq=5 ttl=255 time=0.158 ms

--- 10.10.20.228 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4076ms
rtt min/avg/max/mdev = 0.158/0.168/0.178/0.006 ms
[hadoop@ip-10-10-10-62 bin]$
```

5. Create python file *<nano WordCount.py>* and fill it with pyspark code using traditional copy and past (clipboard).

a. Step 1: creating WordCount.py script

```
[hadoop@ip-10-10-10-62 bin]$ sudo nano WordCount.py
```

b. Step 2: paste and save

```
Session ID: odwy[redacted]                                    Ins
 GNU nano 2.9.8

from pyspark.sql import SparkSession
from pyspark.sql.functions import explode
from pyspark.sql.functions import split

spark = SparkSession \
    .builder \
    .appName("StructuredNetworkWordCount") \
    .getOrCreate()

# Create DataFrame representing the stream of input lines from connection to localhost:9999
lines = spark \
    .readStream \
    .format("socket") \
    .option("host", "localhost") \
    .option("port", 9999) \
    .load()

# Split the lines into words
words = lines.select(
   explode(
       split(lines.value, " ")
   ).alias("word")
)

# Generate running word count
wordCounts = words.groupBy("word").count()

query = wordCounts \
    .writeStream \
    .outputMode("complete") \
    .format("console") \
   .start()

query.awaitTermination()
```

c. Step 3: Open new terminal (terminalB) run command *< nc -lk 9999>*

```
 E:::::EEEEEEEEEE    M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
 E:::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::::RR
 E:::::EEEEEEEEEE    M:::::M   M:::::M   M:::::M   R:::RRRRRR:::::R
 E:::::E             M:::::M    M:::M    M:::::M   R:::R      R:::::R
 E:::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R:::::R
E::::::EEEEEEEE:::::E M:::::M             M:::::M   R:::R      R:::::R
E::::::::::::::::::::E M:::::M             M:::::M RR:::::R     R:::::R
EEEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRRR

[hadoop@ip-10-10-10-62 bin]$ cd ~
[hadoop@ip-10-10-10-62 ~]$ nc -lk 9999
```

d. Step 4: Start our pyspark script in spark-submit on terminalA using *<spark-submit WordCount.py localhost 9999>*.

```
[hadoop@ip-10-10-10-62 bin]$
[hadoop@ip-10-10-10-62 bin]$ spark-submit WordCount.py localhost 9999
21/06/17 20:14:03 INFO SparkContext: Running Spark version 2.4.7-amzn-1
21/06/17 20:14:03 INFO SparkContext: Submitted application: StructuredNetworkWordCount
21/06/17 20:14:03 INFO SecurityManager: Changing view acls to: hadoop
21/06/17 20:14:03 INFO SecurityManager: Changing modify acls to: hadoop
21/06/17 20:14:03 INFO SecurityManager: Changing view acls groups to:
21/06/17 20:14:03 INFO SecurityManager: Changing modify acls groups to:
```

e. From terminalB start typing words

```
[hadoop@ip-10-10-10-62 ~]$ nc -lk 9999
I love Amazon
This is my first VLS event
This is a lab test
This is a lab test
```

f. Count results seen from terminalA

```
-------------------------------------------
Batch: 4
-------------------------------------------
+------+-----+
|  word|count|
+------+-----+
|   VLS|    1|
|  love|    1|
|   lab|    2|
|    is|    3|
|    my|    1|
|     I|    1|
|  This|    3|
|     a|    2|
| event|    1|
|Amazon|    1|
| first|    1|
|  test|    2|
+------+-----+
```

6. Use nano to create wordcount-producer.py file and copy code into it

a. Step 1: create file

```
[ec2-user@ip-10-10-20-228 bin]$ cd ~
[ec2-user@ip-10-10-20-228 ~]$ sudo nano wordcount-producer.py
```

b. Step 2: Coping code

```
# Producer
import socket
import time
import json
import random
import datetime


def getReferrer():
    data = {}
    now = datetime.datetime.now()
    str_now = now.isoformat()
    f1=random.choice(['Madrid', 'CapeTown', 'Denver', 'Texas', 'Yaounde','Paris', 'Mumbai', 'Sydney'])
    temp = random.random() * 100
    f2=int(round(temp, 2))
    f3=random.choice(['OK', 'FAIL', 'WARN'])
    myrecord= str(f1)+","+str(f2)+","+str(f3)
    return myrecord


#HOST = 'localhost'
#HOST = '192.168.56.108'
HOST = '172.31.53.10' #ec2  server producer
#HOST='172.31.82.51'  # EMR cluster master node ip
PORT = 9999 #8888


with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    conn, addr = s.accept()
    with conn:
        print('Connected by', addr)
        while True:
            data = getReferrer()
            print(data)
            conn.send(data.encode('utf-8'))

s.close()
```

c. Step 3: running wordcount producer on EC2 instance

```
[ec2-user@ip-10-10-20-228 ~]$ sudo nano wordcount-producer.py
[ec2-user@ip-10-10-20-228 ~]$ python3 wordcount-producer.py
```

d. Step 4: starting spark streaming remotely using command *<spark-submit WordCountRemote.py <ec2-instance privateIP> 9999>*

```
[hadoop@ip-10-10-10-62 bin]$ spark-submit WordCountRemote.py 10.10.20.228 9999
21/06/17 21:41:28 INFO SparkContext: Running Spark version 2.4.7-amzn-1
21/06/17 21:41:28 INFO SparkContext: Submitted application: StructuredNetworkWordCount
21/06/17 21:41:28 INFO SecurityManager: Changing view acls to: hadoop
21/06/17 21:41:28 INFO SecurityManager: Changing modify acls to: hadoop
21/06/17 21:41:28 INFO SecurityManager: Changing view acls groups to:
21/06/17 21:41:28 INFO SecurityManager: Changing modify acls groups to:
21/06/17 21:41:28 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view
21/06/17 21:41:28 INFO Utils: Successfully started service 'sparkDriver' on port 35801.
21/06/17 21:41:28 INFO SparkEnv: Registering MapOutputTracker
21/06/17 21:41:28 INFO SparkEnv: Registering BlockManagerMaster
```

e. Step 5: check on python wordcount-producer inside ec2-user terminal

7. Coping external producer script for streaming outside network and starting the spark job with YARN as resource manager

a. Creating and copying code to new external producer

```
[hadoop@ip-10-10-10-62 bin]$ l^C
[hadoop@ip-10-10-10-62 bin]$ sudo nano WordCountExternalProducer.py
```

view IP address

1. ifconfig

#testing reachability

2. ping <ip-address> -c 5

creating WordCount App

3. sudo nano wordcount.py

#creating listener on secibd terminal

4. nc -lk 9999

#checking if application is avaliable on port

5. lsof -i :9999

#start spark wordcount app locally

6. spark-submit wordcount.py localhost 9999

6.2 yarn logs –applicationId <application id>

#creating wordcount producer script

7. sudo nano wordcount-producer.py

#start wordcount producer script

8. python3 wordcount-producer.py

#start spark wordcount app to read from wordcount-producer

9. spark-submit WordCountRemote.py <ec2-instance privateIP> 9999

#creating spark app to read from external producer on internet

10. sudo nano wordcountExternalProducer.py

#check if files are created on HDFS

11. hadoop dfs -ls /user/hadoop/output/ or dhfs dfs -ls /user/hadoop/output/

#copying from local directory to HDFS

-- hadoop fs -copyFromLocal /path/in/linux /hdfs/path