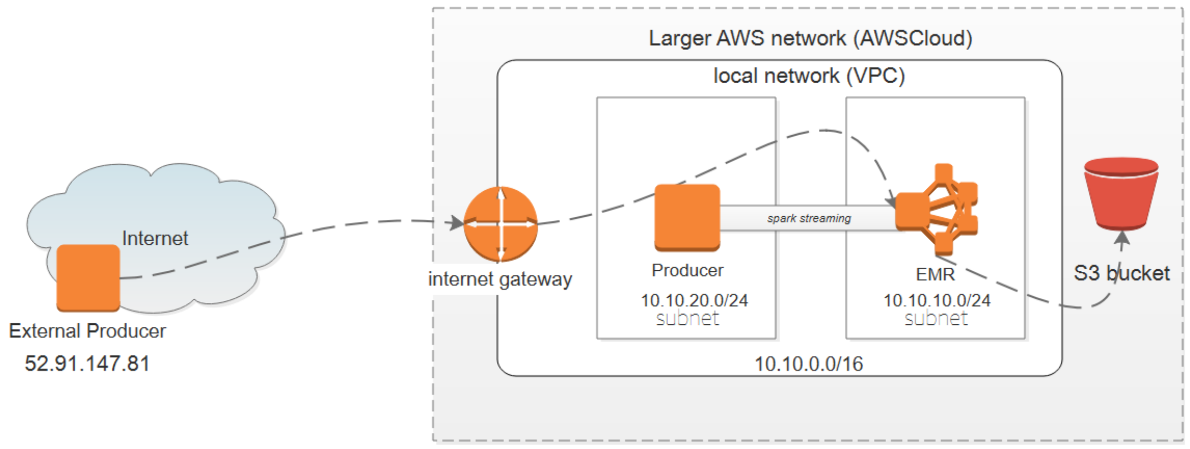
Consolidation Lab:

In this Lab we will be setting up our EC2 instance to produce streaming traffic to our Hadoop cluster which will have a spark streaming script for consuming the data. You will see how something as simple as a word can be leveraged over a TCP socket streaming to allow multiple spark executors to process the data and produce output into HDFS. We will be able to monitor and control this job using Spark, YARN and HDFS web applications.



1. First you will need to connect to you EC2 instance, make sure that the instance and Hadoop cluster have network reachability.
    a. What is the ip address of the master instance?
    b. Which port will the spark streaming producer push traffic in? (would you stream on port 443?)
2. Now we see that there is a security group rule (firewall rule) that we need to add for the spark streaming to go over the network
    a. What is the subnet range we need to allow on the security group rule?
3. Let's create python file which we will use as the pyspark application, we will later submit this using spark-submit. As a first run test we will use localhost and observe if we can stream over a TCP socket. Open another terminal (let's call this terminalB) on your master instance, run this netcat command *< nc -lk 9999>*
4. Now on terminalA lets use spark-submit to start the streaming application and read text coming from terminalB. To do this, we use this spark-submit command on terminalA *<spark-submit WordCount.py localhost 9999>*. From terminalB start typing words
    a. What is happening on terminal while you are typing in terminalB?
    b. How can we improve this by streaming from remote producer, what would you change in the code base?
5. Let's modify our WordCount.py code to connect to local producer, in this case our EC2 instance. Instead of typing text inside the terminal, we will be running a python script to produce the streaming traffic.
    a. Create a new python file wordcount-producer.py and copy producer code into it. What are we streaming and what network socket are we using for to push to spark?

   b. Try running the python script on the EC2 instance using *<python3 wordcount-producer.py>*, what do you see?
   c. Let's try running the modified version of WordCount.py, we will rename it to WordCountRemote.py. How will our spark-submit command look like?
   d. How would we improve this wordcount application? Maybe we can store the output from our stream on to some files to use for later analysis.
6. Let's use the already prepared pyspark code from the github repository and copy it into a new python file called WordCountExternalProducer.py.
   a. What is the IP address of the external producer?
   b. How to we verify if the producer is reachable on the needed port for streaming?
   c. What HDFS path will we find the checkpoint files?
   d. Where else are we pushing the output too?
7. Now that we have our WordCountExternalProducer.py lets submit it to yarn using spark-submit you can use the following command *<spark-submit WordCountExternalProducer.py 52.91.147.81 9999 --deploy-mode cluster --master yarn>*
   a. While the job is still running open a tab on your chrome/firefox browser and access the spark history console <*http://<DNSname-of-masterNode:8088/*>
   b. Inside the Web UI navigate through main console and job tab to try and find your latest application run in the Spark Web UI