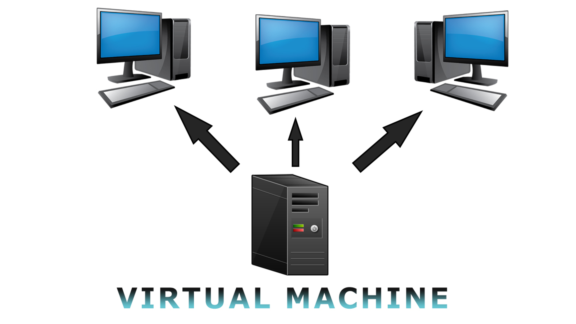


**POC ON DEVICE FAILURE PREDICTION FOR LENOVO**

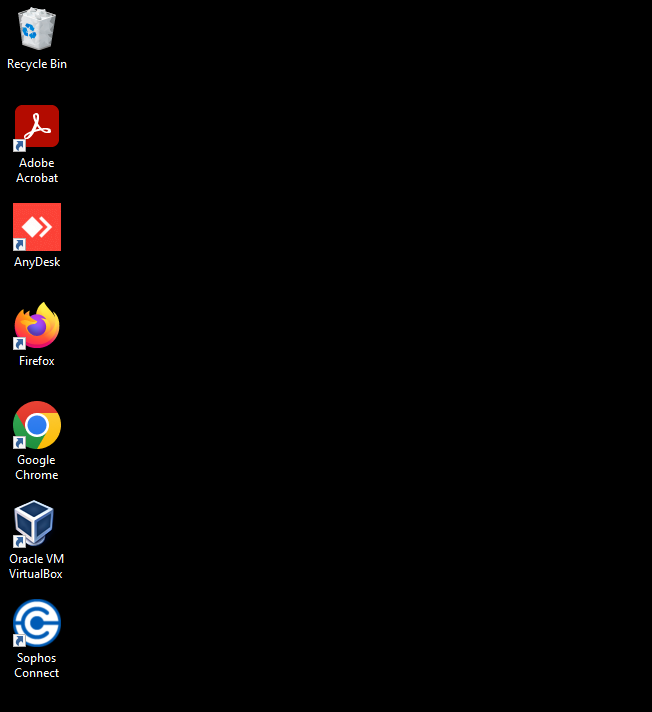
**AWS**

# Step 1: Set up Virtual Machine

* To perform all operations we need one VM
* Following are the configuration which is required to operate this Lenovo POC:
* RAM:64GB
* Hardisk:500 GB
* Operating System: Ubantu
* Some other details:
* Sophos connect and Putty
* VM Display Name: Manas\_Presales\_34685
* IP ADDR: 172.30.44.147



# Step 2: Need to open VPN (Sophos connect)

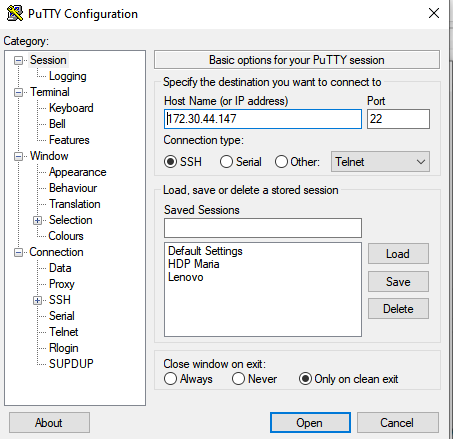


# Step 3: Give all credentials

* Username : Enter our user name
* Password: Give password + googleautheticator no
* Open Google authenticator from mobile and take 6 digit no from there and add in password

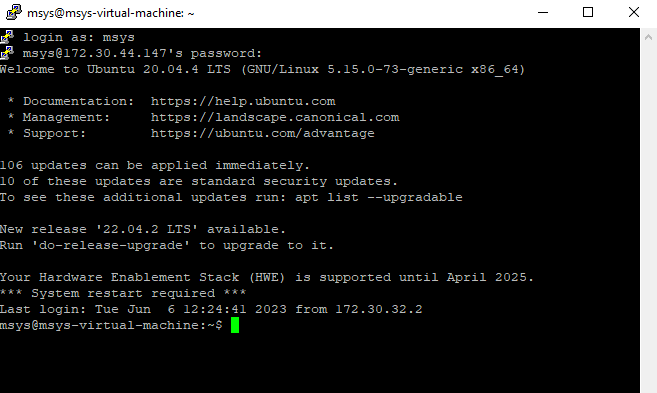
# Step 4: Open Putty

* Give IP address of our VM
* Open it
* Save it for our convenience



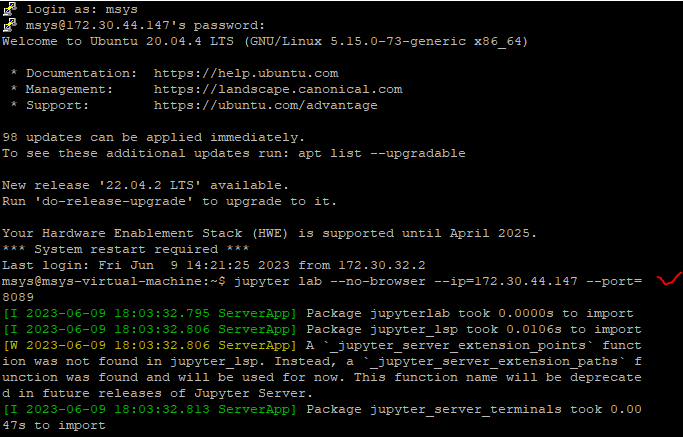
# Step 5: Give credentials of VM

* login as: msys
* pass: Give password here

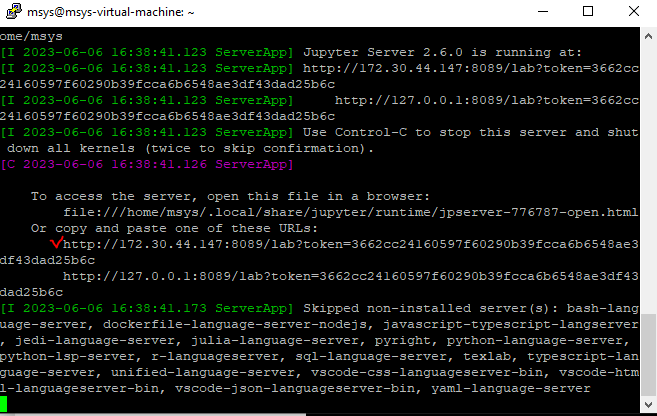


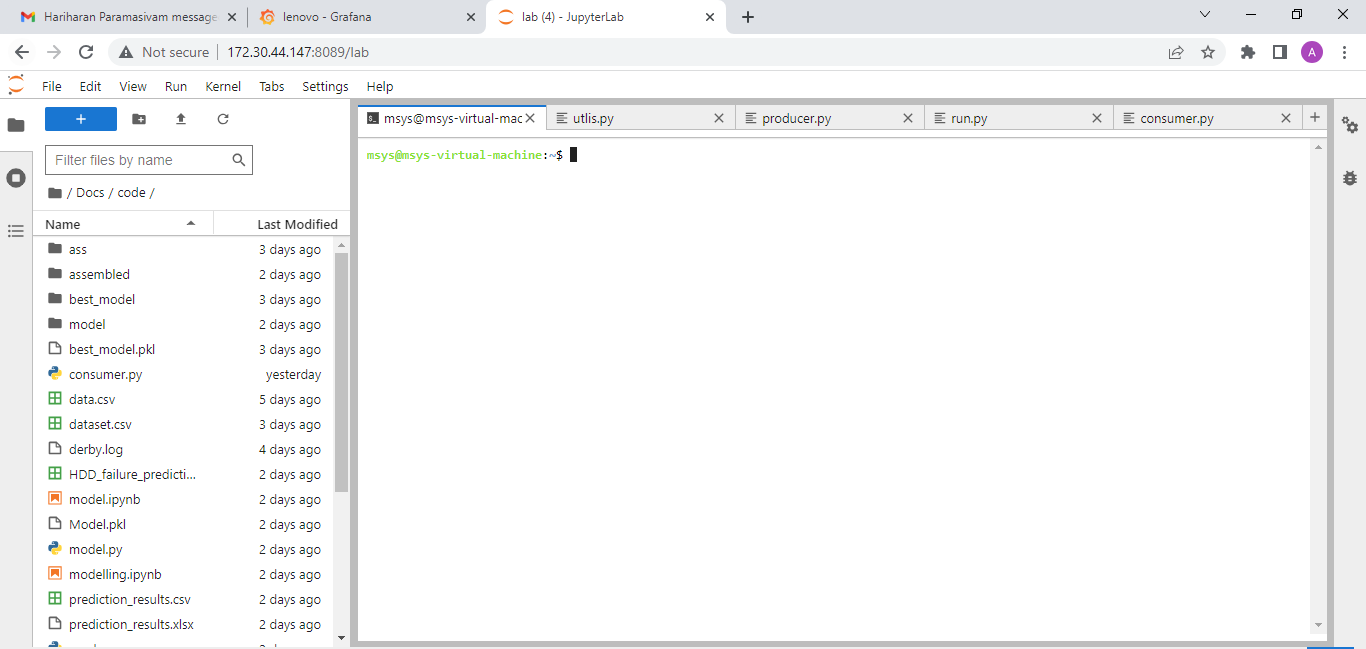
# Step 6: Open Jupyter Notebook/Lab

* Copy this and paste there
* jupyter lab --no-browser --ip=172.30.44.147 --port=8089



# Step 7: Copy url and paste in any browser





# Step 8: Installation of Kafka

"Installing" Kafka is done by downloading the code from one of the several mirrors. After finding the latest binaries from [the downloads page](https://kafka.apache.org/downloads), choose one of the mirror sites and wget it into your home directory.

~$ wget http://apache.claz.org/kafka/2.2.0/kafka\_2.12-2.2.0.tgz

After that we will need to unpack it. At this point, we also like to rename the Kafka to something a little more concise.

~$ tar -xvf kafka\_2.12-2.2.0.tgz~$ mv kafka\_2.12-2.2.0.tgz kafka

Before continuing with Kafka, we'll need to install Java.

~$ sudo apt install openjdk-8-jdk -y

Test the Java installation by checking the version.

~$ java -version

Now you can cd into the kafka/ directory and start a Zookeeper instance, create a Kafka broker, and publish/subscribe to topics. We can get a feel for this by walking thru the [Kafka Quickstart](https://kafka.apache.org/quickstart), but it will also be covered later in this example.

While we are here, we should install a Python package that will allow us to connect to our Kafka cluster. But first, we 'll need to make sure that we have Python 3 and pip installed on your system. For Lubuntu, Python 3 was already installed and accessible with python3, but we had to install pip3 with

~$ pip3 install kakfa-python

Confirm this installation with

~$ pip3 list | grep kafka

# Step 9 : Installation of Hadoop

Similar to Kafka, we first need to download the binaries from a mirror. All releases can be found [here](https://hadoop.apache.org/releases.html). We used Hadoop 2.8.5. Select a mirror, download it, then unpack it to your home directory.

* NOTE: We wrote out this step as using wget, but feel free to download the tar thru a browser (sometimes it is faster i think)
* NOTE #2: Again, We renamed my directory for convenience.

~$ wget https://archive.apache.org/dist/hadoop/common/hadoop-2.8.5/hadoop-2.8.5.tar.gz~$ tar -xvf hadoop-2.8.5.tar.gz~$ mv hadoop-2.8.5.tar.gz hadoop~$ cd hadoop~/hadoop$ pwd

/home/<USER>/hadoop

Edit .bashrc found in your home directory, and add the following.

expotHADOOP\_HOME=/home/<USER>/hadoopexport HADOOP\_CONF\_DIR=$HADOOP\_HOME/etc/hadoopexport HADOOP\_HDFS\_HOME=$HADOOP\_HOMEexport HADOOP\_INSTALL=$HADOOP\_HOMEexport HADOOP\_MAPRED\_HOME=$HADOOP\_HOMEexport HADOOP\_COMMON\_HOME=$HADOOP\_HOMEexport HADOOP\_HDFS\_HOME=$HADOOP\_HOMEexport YARN\_HOME=$HADOOP\_HOMEexport HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_HOME/lib/nativeexport PATH=$PATH:$HADOOP\_HOME/sbin:$HADOOP\_HOME/binexport HADOOP\_OPTS="-Djava.library.path=$HADOOP\_HOME/lib/native"

export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64

Be sure to insert your username on the first line, for example, for me it is export HADOOP\_HOME=/home/davis/hadoop. Also, double check that your $JAVA\_HOME is correct. This should be the default install location, but it may go somewhere else.

Also also, remember to execute your .bashrc after editing it so that the changes take place (this will be important later on)

~$ source .bashrc

Next, we will need to edit/add some configuration files. From the Hadoop home folder (the one named hadoop that is in your home directory), cd into etc/hadoop.

Add the following to hadoop-env.sh

export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64export HADOOP\_CONF\_DIR=${HADOOP\_CONF\_DIR:-"/home/<USER>/hadoop/etc/hadoop"}

Replace the file core-site.xml with the following:

<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>fs.default.name</name>

<value>hdfs://localhost:9000</value>

</property>

</configuration>

Replcae the file hdfs-site.xml with the following:

<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.permission</name>

<value>false</value>

</property>

</configuration>

As mentioned earlier, we are just setting up a single-node Hadoop cluster. This isn'y very realistic, but it works for this example. For this to work, we need to allow our machine to SSH into itself.

First, install SSH with

~$ sudo apt install openssh-server openssh-client -y

Then, set up password-less authentication

~$ ssh-keygen -t rsa~$ cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys

Then, try SSH-ing into the machine (type exit to quit the SSH session and return)

~$ ssh localhost

With Hadoop configured and SSH setup, we can start the Hadoop cluster and test the installation.

~$ hdfs namenode -format~$ start-dfs.sh

NOTE: These commands should be available anywhere since we added them to the PATH during configuration. If we 're having troubles, hdfs and start-dfs are located in hadoop/bin and hadoop/sbin, respectively.

Finally, test that Hadoop was correctly installed by checking the Hadoop Distributed File System (HDFS):

~$ hadoop fs -ls /

If this command doesn't return an error, then we can continue!

# Step 10 : Installation of Hive

Different releases of Hive can be found [here](https://hive.apache.org/downloads.html), or from the the [Apache Archive](http://archive.apache.org/dist/hive/). We downloaded used Hive 2.3.5 for this example. Be sure to download the binaries, rather than the source.

~$ wget http://archive.apache.org/dist/hive/hive-2.3.5/apache-hive-2.3.5-bin.tar.gz~$ tar -xvf apache-hive-2.3.5-bin.tar.gz~$ mv apache-hive-2.3.5-bin.tar.gz hive

Add the following to your .bashrc and run it with source

export HIVE\_HOME=/home/<USER>/hiveexport PATH=$PATH:$HIVE\_HOME/bin

Give it a quick test with

~$ hive --version

Add the following directories and permissions to HDFS

~$ hadoop fs -mkdir -p /user/hive/warehouse~$ hadoop fs -mkdir -p /tmp~$ hadoop fs -chmod g+w /user/hive/warehouse~$ hadoop fs -chmod g+w /tmp

Inside ~/hive/conf/, create/edit hive-env.sh and add the following

export HADOOP\_HOME=/home/<USER>/hadoopexport HADOOP\_HEAPSIZE=512export HIVE\_CONF\_DIR=/home/<USER>/hive/conf

While still in ~/hive/conf, create/edit hive-site.xml and add the following

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:derby:;databaseName=/home/davis/hive/metastore\_db;create=true</value>

<description>JDBC connect string for a JDBC metastore.</description>

</property>

<property>

<name>hive.metastore.warehouse.dir</name>

<value>/user/hive/warehouse</value>

<description>location of default database for the warehouse</description>

</property>

<property>

<name>hive.metastore.uris</name>

<value>thrift://localhost:9083</value>

<description>Thrift URI for the remote metastore.</description>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>org.apache.derby.jdbc.EmbeddedDriver</value>

<description>Driver class name for a JDBC metastore</description>

</property>

<property>

<name>javax.jdo.PersistenceManagerFactoryClass</name>

<value>org.datanucleus.api.jdo.JDOPersistenceManagerFactory</value>

<description>class implementing the jdo persistence</description>

</property>

<property>

<name>hive.server2.enable.doAs</name>

<value>false</value>

</property>

</configuration>

(optional) Since Hive and Kafka are running on the same system, we 'll get a warning message about some SLF4J logging file. From your Hive home you can just rename the file

~/hive$ mv lib/log4j-slf4j-impl-2.6.2.jar lib/log4j-slf4j-impl-2.6.2.jar.bak

Now we need to create a database schema for Hive to work with using schematool

~$ schematool -initSchema -dbType derby

We are now ready to enter the Hive shell and create the database for holding tweets. First, we need to start the Hive Metastore server with the following command.

~$ hive --services metastore

This should give some output that indicates that the metastore server is running. You'll need to keep this running, so open up a new terminal tab to continue with the next steps.

Now, enter the Hive shell with the hive command

~$ hive

...

hive>

Make the database for storing our Twitter data:

hive> CREATE TABLE tweets (text STRING, words INT, length INT)

> ROW FORMAT DELIMITED FIELDS TERMINATED BY '\\|'

> STORED AS TEXTFILE;

You can use SHOW TABLES; to double check that the table was created.

# Step 11 : Installation of Spark

Download from <https://spark.apache.org/downloads.html>, make sure you choose the option for Hadoop 2.7 or later (unless you used and earlier version).

Unpack it, rename it

~$ tar -xvf Downloads/spark-2.4.3-bin-hadoop2.7.tgz~$ mv spark-2.4.3-bin-hadoop2.7.tgz spark

Although I have been able to run Spark before without installing Scala, we can avoid some issues by ensuring Scala is installed on our system.

~$ sudo apt install scala -y

Test with

~$ scala -version

Instead of writing Scala code, we will write our Spark transformer in Python, so we will need pyspark.

~$ pip3 install pyspark

Check with

~$ pip3 list | grep spark

Now we need to add the Spark /bin files to the path, so open up .bashrc and add the following

export PATH=$PATH:/home/<USER>/spark/binexport PYSPARK\_PYTHON=python3

By setting the PYSPARK\_PYTHON variable, we can use PySpark with Python3, the version of Python we have been using so far.

After running source .bashrc, try entering the PySpark shell

~$ pyspark

...

Using Python version ....

SparkSession available as 'spark'.

>>>

One last thing! We need a JAR file that wasn't included in PySpark that will allow us to connect to Kafka. Download the JAR file with the artifact ID spark-streaming-kafka-0-8-assembly\_2.11 from [search.maven.org](https://search.maven.org/search?q=a:spark-streaming-kafka-0-8-assembly_2.11).



info@msystechnologies.com

