ETL2 part 1: Building Hadoop & SPARK

Table of Contents

[Overview 2](#_Toc182500079)

[Diagram 2](#_Toc182500080)

[1. Build RHEL9 VM 2](#_Toc182500081)

[2. Update systems packages & install Java on RHEL VM 7](#_Toc182500082)

[3. Install & setup Hadoop on RHEL VM 8](#_Toc182500083)

[4. Install Java on Windows VM 16](#_Toc182500084)

[6. Installing Python on Windows VM 17](#_Toc182500085)

[7. Install and configure Spark & Hadoop on Windows VM 17](#_Toc182500086)

[8. Install Jupyter Notebook 23](#_Toc182500087)

[9. Running PySpark to pull data from HDFS 24](#_Toc182500088)

# Overview

This ETL will demonstrate how to build a Hadoop system that houses and computes the data for transformation and into loading.

The data contain details of the performances of the projects within the bank. In an effort to cut down on costs in the North American region, the bank has decided to review the project within the region that have the lowest grades.

The raw data will be stored in HDFS (Hadoop File System) where the analysts will transform the data to meet business requirements and load it into a MySQL database. The visual analytics tool, Grafana, will be used to connect to the MySQL database to create the dashboard needed for the business to make decision on which projects will be terminated.

# Diagram

A computer and a cloud of information

Description automatically generated with medium confidence

# 1. Build RHEL9 VM

|  |
| --- |
| Description: Given this project is small scale, all components of Hadoop will be installed on one VM. For operational practice, it is advised to have a multinode platform. The VM design will be that of a RHEL9 machine. |
| 1. Open VirtualBox and create VM with these settings.          2. Configure RHEL settings.  - Set root password  - Connect to Red Hat: Make sure you have a subscription to Red Hat. The credentials you will use will be what you have set up for your Red Hat account.    - Installation Source: Red Hat CDN  - Software Selection: Workstation  - Installation Destination: Select Local Standard Disk    - Network & Hostname (\*Optional): HD-PROD-1    - Click Begin Installation |

# 2. Update systems packages & install Java on RHEL VM

|  |
| --- |
| Description: Hadoop requires Java to be installed so a version of Java must be installed on the RHEL machine where Hadoop will be hosted. \*Note: Recommended version is Java 8 or 11. |
| \*Special note: For first-time practice purposes, a recommendation would be perform all the steps below as root. It’s easier to execute commands to maintain project flow and avoid confusion on which profile code is being executed on. Run the following command to access root:  su –  Enter password set for root access.  1. Download Java by running this command:  sudo dnf install java-1.8.0-openjdk -y  \*Note: The below shows Java 11under a different command. This is due to a restart where Java 8 was used. However, the following would occur if installed properly.    2. Verify the installation by running this command:  java -version    3. Create an environment variable called JAVA\_HOME within the /etc/profile.d/java.sh. Add this to the file:  export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.432.b06-2.el9.x86\_64/jre  export PATH=$PATH:$JAVA\_HOME/bin    4. Apply the changes by running the command:  source /etc/profile.d/java.sh |

# 3. Install & setup Hadoop on RHEL VM

|  |
| --- |
| Description: With Java installed and the environment variable set, Hadoop must be downloaded and configured. |
| 1. Download Hadoop by running the following command:  sudo wget <https://downloads.apache.org/hadoop/common/hadoop-x.y.z/hadoop-x.y.z.tar.gz>  \*Note: The ‘x.y.z’ portion must be looked up to retrieve the recent and correct version found on the Hadoop site. In this case, version Hadoop-3.3.5 was installed. Also, due to project restart the below shows a different version. This is to show what a successful installation looks like.    2. Extract the Hadoop download by running the following command:  tar -xzvf hadoop-3.3.5.tar.gz  3. Move the Hadoop download to /opt/hadoop by running this command:  mv ~/hadoop-3.3.5.tar.gz /opt/hadoop  4. Create Hadoop environment variables by adding the following exports to the /etc/profile.d/Hadoop.sh file:  export HADOOP\_HOME=/opt/hadoop export HADOOP\_INSTALL=$HADOOP\_HOME export HADOOP\_MAPRED\_HOME=$HADOOP\_HOME export HADOOP\_COMMON\_HOME=$HADOOP\_HOME export HADOOP\_HDFS\_HOME=$HADOOP\_HOME export HADOOP\_YARN\_HOME=$HADOOP\_HOME export HADOOP\_CONF\_DIR=$HADOOP\_HOME/etc/hadoop export PATH=$PATH:$HADOOP\_HOME/bin:$HADOOP\_HOME/sbin    5. Apply changes by running the following command:  source /etc/profile.d/hadoop.sh  6. Edit the core-site.xml by running the following command:  nano /opt/hadoop/etc/hadoop/core-site.xml  Then add the coding between the configuration tags:  <property>  <name>fs.defaultFS</name>  <value>hdfs://<\*Place VM IP here\*>:9000</value>  </property>    7. Edit the hdfs-site.xml file with the following:  <configuration>  <property> <name>dfs.replication</name>  <value>1</value>  </property>  <property>  <name>dfs.namenode.name.dir</name> <value>/opt/hadoop/dfs/name</value>  </property>  <property>  <name>dfs.datanode.data.dir</name> <value>/opt/hadoop/dfs/data</value>  </property>  </configuration>    8. Edit the yarn-site.xml with the following:  <property>  <name>yarn.resourcemanager.webapp.address</name>  <value>0.0.0.0:8088</value>  </property>    9. Add SSH keys by running the following commands:  ssh-keygen -t rsa -P "" -f ~/.ssh/id\_rsa  cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys  chmod 600 ~/.ssh/authorized\_keys  Test and verify SSH by running the following commands:  ssh localhost  sudo systemctl status sshd    10. Update /etc/hosts file with the following:  <Your Linux VM IP> <Your Linux VM IP> <name of host>    11. Edit the hadoop-env.sh file and at the following:  Locate “# Set Hadoop-specific environment variables here.” And underneath:  export HDFS\_NAMENODE\_USER=”root”  export HDFS\_DATANODE\_USER=”root”  export HDFS\_SECONDARYNAMENODE\_USER=”root”    Locate “# variable is REQUIRED on ALL platforms except OS X!” and add the following underneath:  Export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.432.b06-2.el9.x86\_64/jre    12. Define the Hadoop user roles in the yarn-env.sh by adding the following:  export YARN\_RESOURCEMANAGER\_USER="<name of user>"  export YARN\_NODEMANAGER\_USER="<name of user>"    13. Format the namenode by running the following command:  hdfs namenode -format  14. Start HDFS by running the following command:  start-all.sh    15.Verify the name node UI by searching on the following in a web browser:  http://<Linux VM IP>:9870    16. Verify the resource manager UI by searching on the following in a web browser:  http://<Your Linux VM IP>:8088    17. Open up firewall rules to prepare connectivity to Windows VM that will be running Pyspark. Run the following commands:  firewall-cmd --add-port=9000/tcp --permanent # NameNode  firewall-cmd --add-port=9864/tcp --permanent # DataNode  firewall-cmd --add-port=9870/tcp --permanent # Web UI  firewall-cmd --add-port=8088/tcp --permanent # ResourceManager UI (if using YARN)  firewall-cmd –add-port=22/tcp –permanent #SSH  firewall-cmd –add-port=9866/tcp –permanent #DataNode  firewall-cmd –reload  firewall-cmd –list-all |

# 4. Install Java on Windows VM

|  |
| --- |
| Description: Java must now be installed on the Windows VM to be compatible with Spark and Hadoop operations that will occuring with the Linux VM. Java 8 will be installed to match the Java version installed on the Linux VM |
| 1. Open up a web browser and search for Java 8 downloads. The Oracle site would be the best place to download.    2. Once you download the executable and click on it, guidance prompts will show up for an easy installation.  3. Verify the Java version once installed by running the following command:  java -version |

# 6. Installing Python on Windows VM

|  |
| --- |
| Description: Python will need to be installed next as it will determine which version of Spark to download. Before downloading either, check compatibility of Python and Spark versions. For this case, Python 3.9.13 was recommended for Spark 3.5.3. |
| 1. Open your web browser and open Python 3.9. \*Note: There are plenty of sources, but an easy and secure download can be found in the Microsoft store for a free installation.    2. Once the installation steps have been completed, verify by checking the Python version by running the following command:  python –version |

# 7. Install and configure Spark & Hadoop on Windows VM

|  |
| --- |
| Description: With the foundation of Python set on the Windows VM, Spark 3.5.3 and Hadoop installations and configurations must be set as well to interact with the Linux VM. |
| 1. On a web browser, search for Apache Spark 3.5.3 downloads. Click on the tar file to download the package.    2. In the command prompt, extract the tar file for Spark to initiate download by running the following command:  tar -xvf spark-3.5.3-bin-hadoop3.tgz  3. Move the Spark installation to a different folder that’s easier to refer to by running the following command:  move “<file path of the where the Spark tar resides>” “C:\spark-3.5.3”  4. To enable Hadoop operations, winutils.exe must be downloaded. The executable can be found in a GitHub repo in the following link:  <https://github.com/cdarlint/winutils>  Select the folder for Hadoop 3.3.5    Locate winutils.exe. Click on the file and then click on the download icon to get the executable.    Move the executable into a newly created folder:  C:\hadoop\bin    If issues are encountered that reflect the message “MSVCR100.dll wasn’t found”, then open you web browser and search for vcredist 2010 download. This can be found on the Microsoft downloads site. The following is the page to find it.    Click Download and select the 64-bit executable    5. Set the environment variables for Spark and Hadoop. Click the Windows button, search “variables”, click on ‘Edit the system environment variables’, and click Environment Variables. In System variables, click New to create the following variables with their respective filepaths:  SPARK\_HOME C:\spark-3.5.3 (\*Referenced folder earlier step)  HADOOP\_HOME C:\hadoop (\*Refers to the folder where the winutils.exe is located)      6. Edit the Path system variable and the new variables with the following:  %SPARK\_HOME%\bin  %HADOOP\_HOME%\bin    7. In C:\\spark-3.5.3\conf copy spark-defaults.conf.template into a conf file called spark-defaults.conf by running the following command:  copy spark-defaults.conf.template spark-defaults.conf  8. In the spark-defaults.conf file, add the following:  spark.hadoop.fs.defaultFS hdfs://10.0.0.212:9000    9. Within the spark-3.5.3\conf directory, create the core-site.xml and hdfs-site.xml. They should look like the following: |

# 8. Install Jupyter Notebook

|  |
| --- |
| Description: Hadoop, Python, and Spark have been installed and configured. The final installation and configuration before running Pyspark on Hadoop is to install the application that will run Pyspark which is Jupyter Notebook. |
| 1. Install Jupyter Notebook by opening the command prompt and running the following command:  pip install jupyter  2. Verify installation by running this command:  jupyter notebook  3. Set environment variables so Jupyter can run Pyspark. The directory reference should be where your Python installation resides.    4. Install PySpark kernel by running the following command:  pip install findspark  5. Create a script in the spark-3.5.3\conf directory that automatically load PySpark in Jupyter by creating a file called pyspark.jupyter and adding the following into it:  import findspark  findspark.init()    6. Open Jupyter Notebook by running the following command:  jupyter notebook  7. Run the following Python code to test if Pyspark is running:  import pyspark  from pyspark.sql import SparkSession  spark = SparkSession.builder.appName("TestApp").getOrCreate() spark.range(5).show() |

# 9. Running PySpark to pull data from HDFS

|  |
| --- |
| Description: With the environments of all applications installed on the VMs, it is finally time to test PySpark by extract a data file from HDFS. |
| 1. Stop the previous Spark session by running the following code:  spark.stop()  2. Write and execute the following code to extract the data from HDFS:    Congrats! You just built Spark and Hadoop to work together! |