# INSTALLING SPARK ON COLAB

DR. ALVIN ANG

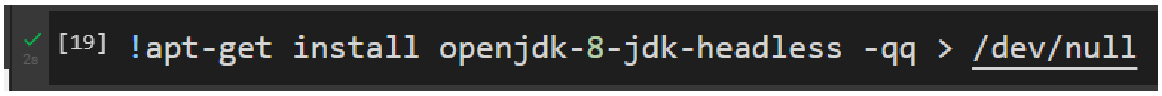
## CONTENTS

## I. STEP 1: INSTALL JAVA 8

- You may reference here for help: https://tatwan.github.io/blog/colab/python/spark/2020/01/06/Colab-Spark-Instructions.html

- IPYNB: https://www.alvinang.sg/s/How_To_Start_A_Spark_Session.ipynb

- But bear in mind that Apache Spark regularly changes its download link (as well as Spark upgrades), thus the code needs to be modified from time to time.



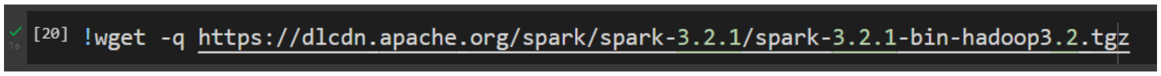
```
[19] !apt-get install openjdk-8-jdk-headless -qq > /dev/null
```

### A. CODE

```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
```

- Run and make sure this line of code works first.

## II. STEP 2: INSTALL APACHE SPARK

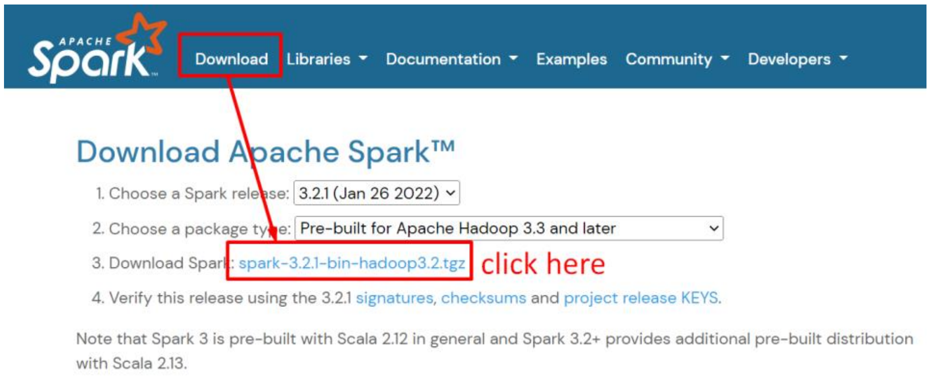


```
[20] !wget -q https://dlcdn.apache.org/spark/spark-3.2.1/spark-3.2.1-bin-hadoop3.2.tgz
```

### A. CODE

!wget -q https://dlcdn.apache.org/spark/spark-3.2.1/spark-3.2.1-bin-hadoop3.2.tgz

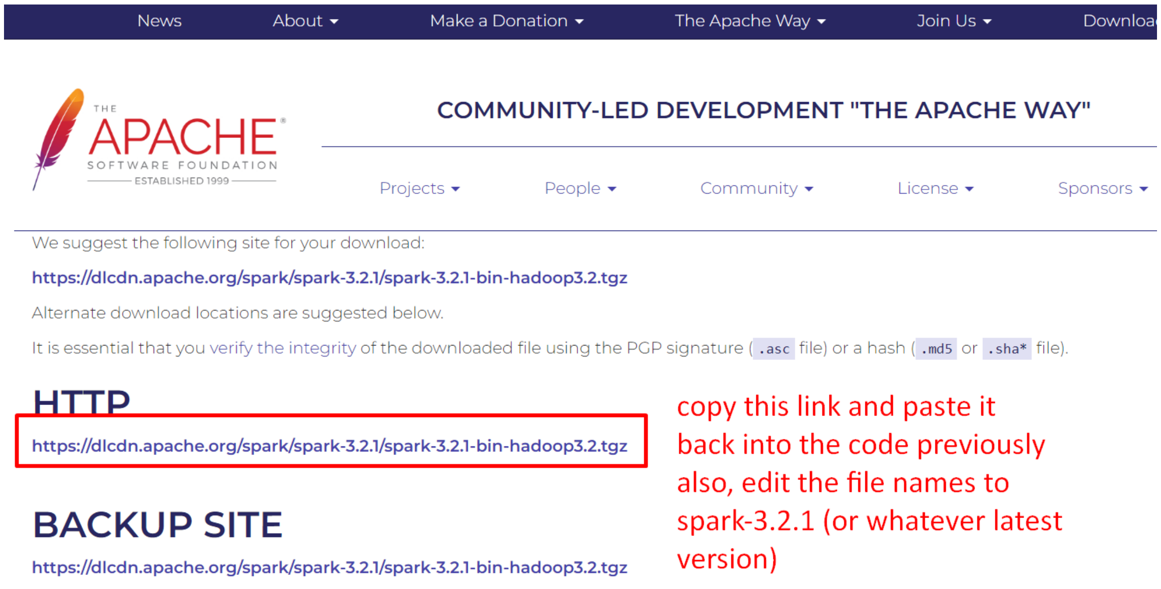
- If this line of code doesn't work, most probably Apache has changed its link.
- Go here: https://spark.apache.org/downloads.html



Download Apache Spark™

1. Choose a Spark release: 3.2.1 (Jan 26 2022)
2. Choose a package type: Pre-built for Apache Hadoop 3.3 and later
3. Download Spark: spark-3.2.1-bin-hadoop3.2.tgz
4. Verify this release using the 3.2.1 signatures, checksums and project release KEYS.

Note that Spark 3 is pre-built with Scala 2.12 in general and Spark 3.2+ provides additional pre-built distribution with Scala 2.13.



We suggest the following site for your download:

https://dlcdn.apache.org/spark/spark-3.2.1/spark-3.2.1-bin-hadoop3.2.tgz

Alternate download locations are suggested below.

It is essential that you verify the integrity of the downloaded file using the PGP signature (`.asc` file) or a hash (`.md5` or `.sha*` file).

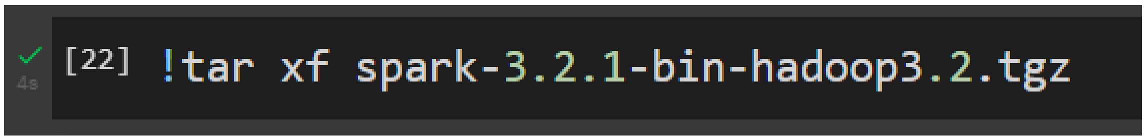
HTTP

https://dlcdn.apache.org/spark/spark-3.2.1/spark-3.2.1-bin-hadoop3.2.tgz

BACKUP SITE

https://dlcdn.apache.org/spark/spark-3.2.1/spark-3.2.1-bin-hadoop3.2.tgz

copy this link and paste it back into the code previously also, edit the file names to spark-3.2.1 (or whatever latest version)



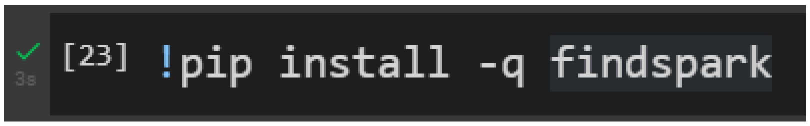
```
[22] !tar xf spark-3.2.1-bin-hadoop3.2.tgz
```

### A. CODE

**!tar xf spark-3.2.1-bin-hadoop3.2.tgz**

- Remember: If Apache upgrades its veresion to 3.X.X or whatever new versions, do edit the name accordingly.

## IV. STEP 4: INSTALL FIND SPARK



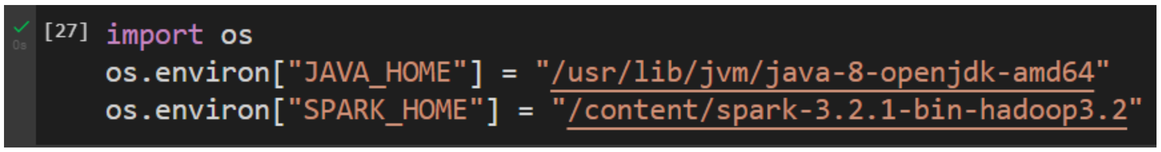
```
[23] !pip install -q findspark
```

### A. CODE

**!pip install -q findspark**

- This is to initialize PySpark.
- Findspark ensures that the environment variables will be properly set.
- PySpark will be imported upon importing findspark

## V. STEP 5: SET ENVIRONMENT HOME



```
[27] import os
     os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
     os.environ["SPARK_HOME"] = "/content/spark-3.2.1-bin-hadoop3.2"
```

### A. CODE

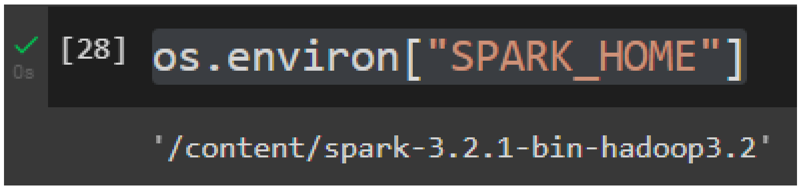
**import os**

**os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"**

**os.environ["SPARK_HOME"] = "/content/spark-3.2.1-bin-hadoop3.2"**

- Once again, remember to change the folder names to the latest updated Apache Spark version.
- Make sure every line runs properly before going to the next line of code.

## VI. STEP 6: CHECK THE FILE LOCATION



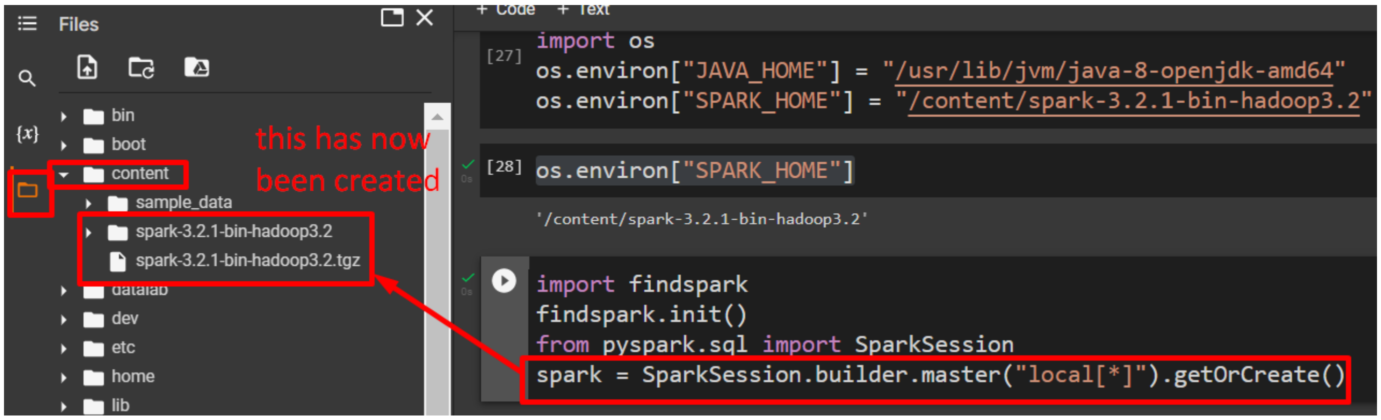
```
[28] os.environ["SPARK_HOME"]

'/content/spark-3.2.1-bin-hadoop3.2'
```

### A. CODE

**os.environ["SPARK_HOME"]**

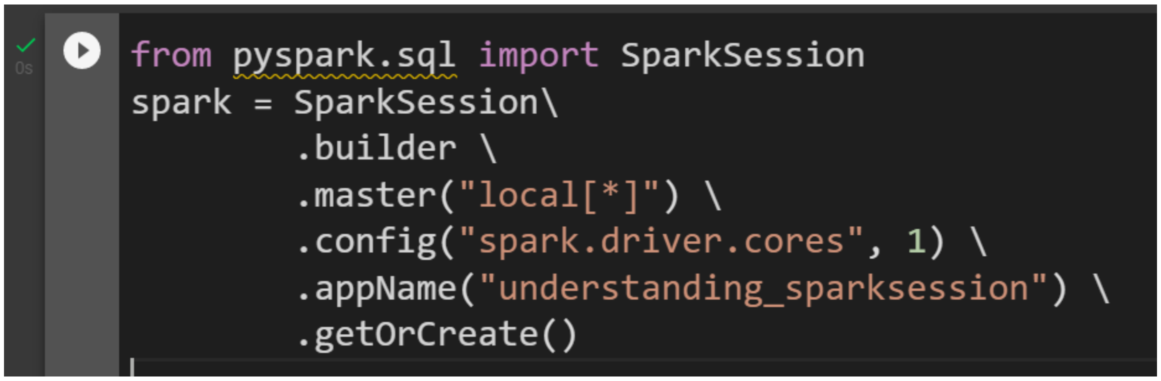
- Just to check where the folder is…

## VII. STEP 7: KICKSTART SPARK CONTEXT


Files
bin
boot
content
sample_data
spark-3.2.1-bin-hadoop3.2
spark-3.2.1-bin-hadoop3.2.tgz
datalab
dev
etc
home
lib
this has now been created
+ Code
+ Text

```
[27] import os
     os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
     os.environ["SPARK_HOME"] = "/content/spark-3.2.1-bin-hadoop3.2"

[28] os.environ["SPARK_HOME"]

'/content/spark-3.2.1-bin-hadoop3.2'

import findspark
findspark.init()
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local[*]").getOrCreate()
```

Which can also be written as….



```
from pyspark.sql import SparkSession
spark = SparkSession\
        .builder \
        .master("local[*]") \
        .config("spark.driver.cores", 1) \
        .appName("understanding_sparksession") \
        .getOrCreate()
```

**there are many ways to write the Sparksession codes.. which we will see below…

**A. CODE**

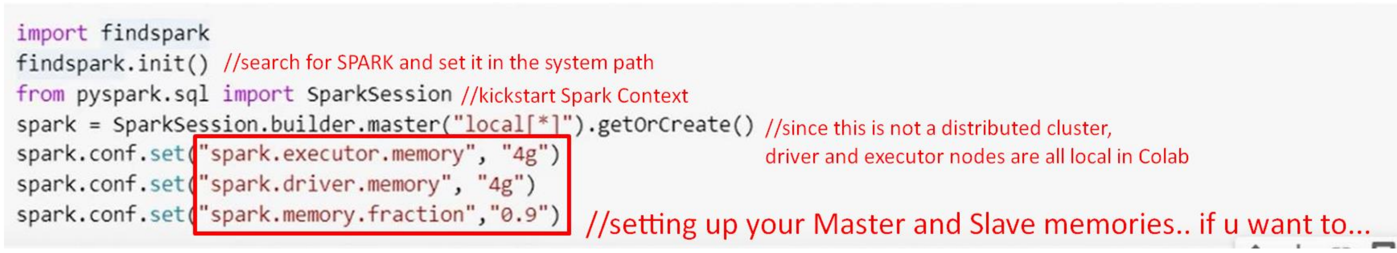
```
import findspark

findspark.init()


from pyspark.sql import SparkSession


spark = SparkSession.builder.master("local[*]").getOrCreate()
```

****note:***

- ***Config and Appname are non-essential… it can be ignored for now first…***
- ***But those above are essential…***

**B. INTERPRETATION**

Another way is to write the code below…

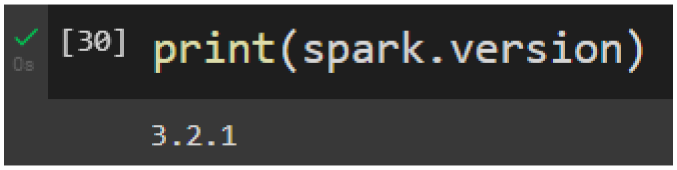


```
import findspark
findspark.init()  //search for SPARK and set it in the system path
from pyspark.sql import SparkSession //kickstart Spark Context
spark = SparkSession.builder.master("local[*]").getOrCreate() //since this is not a distributed cluster,
                                                               driver and executor nodes are all local in Colab
spark.conf.set("spark.executor.memory", "4g")
spark.conf.set("spark.driver.memory", "4g")
spark.conf.set("spark.memory.fraction","0.9")  //setting up your Master and Slave memories.. if u want to...
```

- findspark.init() = kickstart Spark Context
- SparkSession = to connect to Spark cluster
- Builder = to create a new SparkSession.
- Master =
    - Sets the Spark master URL to connect to, such as "local" to run locally, "local[4]" to run locally with 4 cores, or "spark://master:7077" to run on a Spark standalone cluster.
    - Usually, it would be either yarn or mesos depends on your cluster setup.
    - Use local[x] when running in Standalone mode (which is obvious in Colab since Colab can't run clusters..)
    - x should be an integer value and should be greater than 0.
    - Ideally, x value should be the number of CPU cores you have.
    - Putting as * is to let the computer find a value automatically for you, which is by default.
- getOrCreate = Gets an existing SparkSession or, if there is no existing one, creates a new one.

● If no configuration is done for the CPU cores / memory, it will be automatically set to MAX OUT your computer capacity…

## VIII. STEP 8: CHECK SPARK VERSION



```
[30] print(spark.version)

3.2.1
```

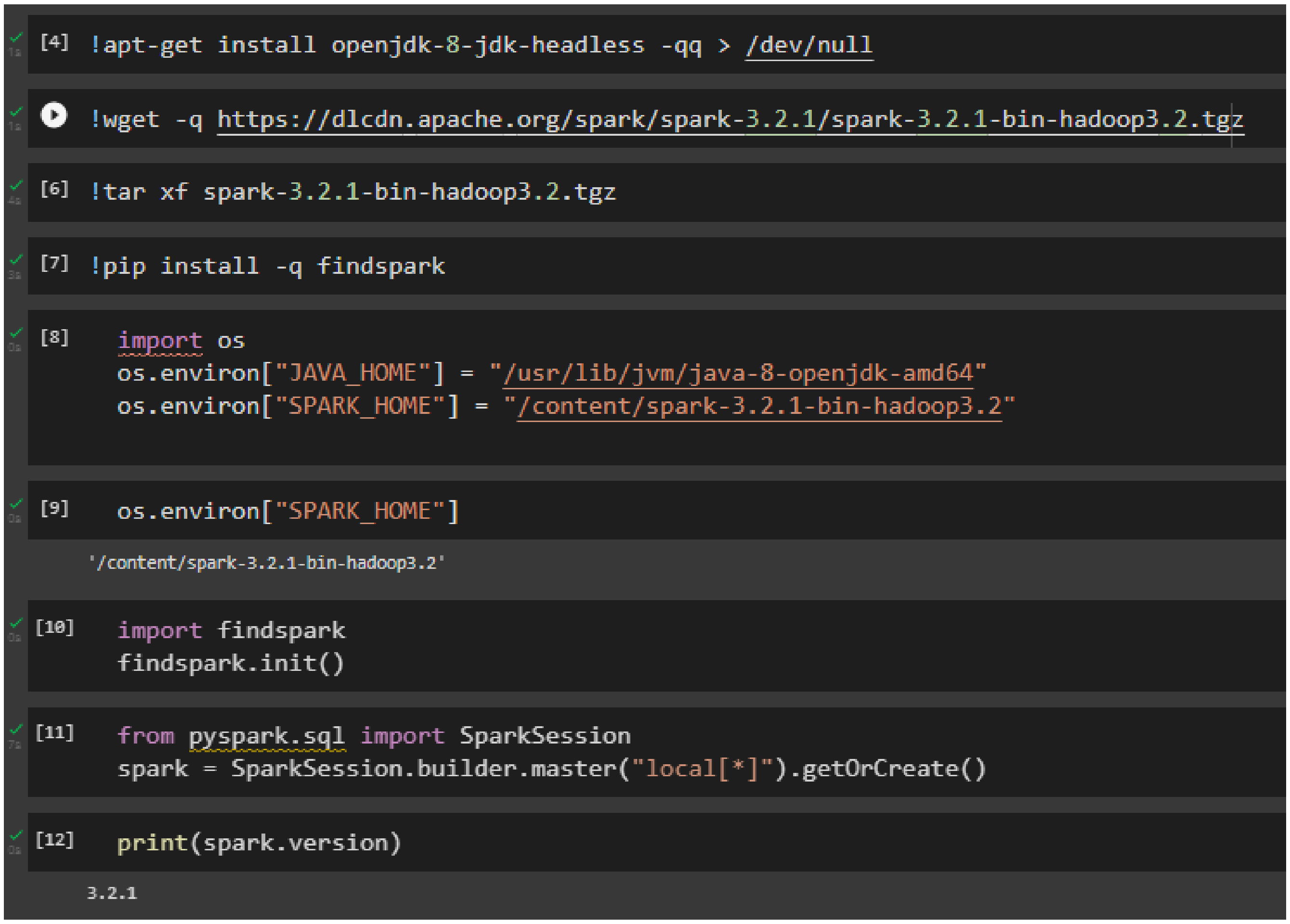
### A. CODE

**print(spark.version)**

## IX. PUTTING IT ALTOGETHER…

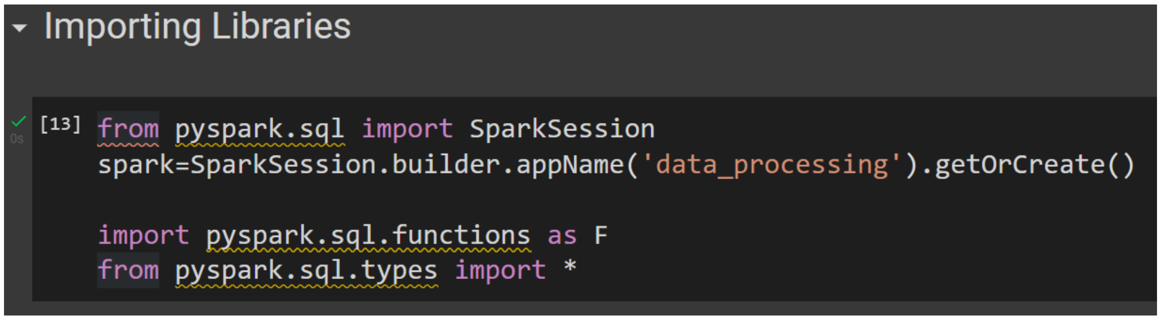
https://www.alvinang.sg/s/How_To_Start_A_Spark_Session.ipynb

### A. START A SPARK SESSION



```
[4] !apt-get install openjdk-8-jdk-headless -qq > /dev/null

    !wget -q https://dlcdn.apache.org/spark/spark-3.2.1/spark-3.2.1-bin-hadoop3.2.tgz

[6] !tar xf spark-3.2.1-bin-hadoop3.2.tgz

[7] !pip install -q findspark

[8]  import os
     os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
     os.environ["SPARK_HOME"] = "/content/spark-3.2.1-bin-hadoop3.2"

[9]  os.environ["SPARK_HOME"]

    '/content/spark-3.2.1-bin-hadoop3.2'

[10] import findspark
     findspark.init()

[11] from pyspark.sql import SparkSession
     spark = SparkSession.builder.master("local[*]").getOrCreate()

[12] print(spark.version)

    3.2.1
```

## B. IMPORT LIBRARIES


Importing Libraries

```
[13] from pyspark.sql import SparkSession
     spark=SparkSession.builder.appName('data_processing').getOrCreate()

     import pyspark.sql.functions as F
     from pyspark.sql.types import *
```

- appName =
    - Used to set your application name.
    - If no name is set, a randomly generated name will be used.

- Config = Sets a config option by specifying a (key, value) pair (used to setup your cores and memory I believe...)

## ABOUT DR. ALVIN ANG

Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He is a scientist, entrepreneur, as well as a personal/business advisor. More about him at [www.AlvinAng.sg](http://www.AlvinAng.sg).