Step 1: Install docker on your platform by visiting:

https://docs.docker.com/get-docker/

or

https://www.docker.com/get-started/

You need to follow the instructions from these site on how to install docker on your own Operating System.

Step 2: Install git on your platform by visiting:

Windows:
https://gitforwindows.org

MAC:
https://git-scm.com/download/mac

Ubuntu:
sudo apt update
sudo apt install git-all

Step3: Create a directory on your file system where you want to store the Spark and Hadoop docker containers: For example
I create a new folder named foo. Inside foo, clone the docker containers for Spark and Hadoop

git clone https://github.com/gettyimages/docker-spark.git

A new directory called docker-spark will be generated. Docker container info will be stored under that directory.

Step 4: Set the JAVA_HOME environment variable:

Under docker-spark directory open docker-compose.yml file and add the following line:

JAVA_HOME: /usr/lib/jvm/java-8-openjdk-amd64/

Under the line SPARK_PUBLIC_DNS: localhost (it exists in two places so do it in both places)

Step 4.5:

Ubuntu: apt install docker-compose (sudo needed if not running as sudo su)

Step 5: Ready to fire up the docker containers:

Under the docker-spark directory execute:
Sudo docker-compose up -d (Make sure to have the -d, if you run it without -d first, please run docker-compose up -d - - remove-orphans to remove the orphaned workers.

Then two docker containers will come up: docker-spark_worker_1
docker-spark_master_1

You can check this by typing in the regular command prompt:
~$ docker ps
If the docker containers come up correct you would see the following:

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
f6738487db21 gettyimages/spark "bin/spark-class org…" 3 minutes ago Up 3 minutes 7012-7015/tcp, 8881/tcp, 0.0.0.0:8081->8081/tcp, :::8081->8081/tcp
docker-spark_worker_1 c3ee3c304a19 gettyimages/spark "bin/spark-class org…" 3 minutes ago Up 3 minutes 0.0.0.0:4040->4040/tcp, :::4040->4040/tcp, 0.0.0.0:6066->6066/tcp, :::6066->6066/tcp,
0.0.0.0:7077->7077/tcp, :::7077->7077/tcp, 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 70017005/tcp docker-spark_master_1

To access either of those containers and work in those you have to execute the following two commands in two separate prompt terminals, in order to obtain two separate containers, one for the master and one for the worker: ~$ docker exec -it docker-spark_worker_1 bash and ~$ docker exec -it docker-spark_master_1 bash

****PLEASE NOTE:

It may be the case that in the compose-up instruction docker container has obtained a different name, if you check carefully by typing in the regular command prompt: ~$ docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
f6738487db21 gettyimages/spark "bin/spark-class org…" 3 minutes ago Up 3 minutes

7012-7015/tcp, 8881/tcp, 0.0.0.0:8081->8081/tcp, :::8081->8081/tcp
dockerspark_worker_1
c3ee3c304a19 gettyimages/spark "bin/spark-class org…" 3 minutes ago Up 3 minutes
0.0.0.0:4040->4040/tcp, :::4040->4040/tcp, 0.0.0.0:6066->6066/tcp, :::6066->6066/tcp,
0.0.0.0:7077->7077/tcp, :::7077->7077/tcp, 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp,
70017005/tcp dockerspark_master_1

To access either of those containers and work in those you have to execute the following two commands, in two separate prompt terminals, in order to obtain two separate containers, one for the master and one for the worker: ~$ docker exec -it dockerspark_worker_1 bash and ~$ docker exec -it dockerspark_master_1 bash

This will bring up a prompt of the form root@master and/or root@worker indicating that you are in the corresponding docker container.
In order to push files from your machine to docker containers then we have mounted the data directory under **docker-spark/data in the /tmp/data** directory on both containers. Any file you have stored in the docker-spark/data on the host machine will appear in **/tmp/data** of the docker containers (docker-spark_worker_1, docker-spark_master_1),
Or containers with the names (dockerspark_worker_1, dockerspark_master_1).

YOU MUST STORE YOUR FILES in BOTH MASTER AND WORKER CONTAINERS.

Step 6: Bringing up the pyspark shell

In the docker-spark_master_1 docker (docker exec -it docker-spark_master_1 bash) you can type pyspark and you wil get the pyspark shell.
Alternatively, do this in the dockerspark_master_1 docker (if this is the name of the master docker). BUT YOU HAVE TO BRING UP PYSPARK ON THE MASTER.

Welcome to

```
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.4.1
      /_/
```

Using Python version 3.5.3 (default, Sep 27 2018 17:25:39) SparkSession
available as 'spark'.
>>>

To access some of your files from your pyspark shell, once you are in the Master Container you must do the following:
Let's assume you have a file stored in /tmp/data named file1.txt

Do:
>>> from pyspark import SparkContext
>>> from operator import add
>>> sc = SparkContext.getOrCreate()
>>> lines = sc.textFile("/tmp/data/file1.txt")

Step 7: Tearing down the docker containers

If you are done with docker containers, then you can tear them down (stop them) by going under the docker-spark directory and typing docker-compose down. If you want to restart them look at Step 5.

INSTALLING TOOKS IN THE DOCKER
Make sure you run those commands in the master, not the worker. If you have issues connecting to http://deb.debian.org/debian, reboot your machine and try again.

For example to use nano and vim

```
apt-get update apt-get
install vim nano
```

CUSTOMIZE DOCKER FOR HADOOP

You also need to run the following commands if you are working with Hadoop and if you are using command make:

```
apt-get update
```

```
apt-get install openjdk-8-jre apt-get
install openjdk-8-jdk-headless apt-get
install make
```

However, the repository in the debian server is updated and changed. So, we want to go and pull the update from the archives version of the debian.
Workaround --> Before going for the update, the user has to execute this two lines:

```
echo "deb http://archive.debian.org/debian/ stretch main contrib non-
free" > /etc/apt/sources.list
echo "deb-src http://archive.debian.org/debian/ stretch main contrib non-
free" >> /etc/apt/sources.list
```

And then try

```
apt-get update
apt-get install vim nano
```

and rest....

This code helps the docker container to take the update from the archived debian.