Running Apache Spark in a Docker Container

The following instructions can be used to install Spark in a SLES 12 SP1/Docker instance running on IBM LinuxONE Community Cloud. One example provides an interactive Spark shell to store and read values from an array. A second example starts Spark master and worker nodes to estimate the value of Pi.

Prerequisites

Sign up for a LinuxONE Community Cloud trial account

If you have not done so already, register at http://www.ibm.com/linuxone/try for a 120-day trial account. You will receive an email containing credentials to access the LinuxONE Community Cloud self-service portal. This is where you can start exploring all our available services.

Deploy a virtual server instance

If you have not deployed a virtual server already, please follow these instructions http://developer.ibm.com/linuxone/wp-content/uploads/sites/57/virtual-servers-quick-start.pdf to create one before proceeding. Make sure you select a flavor (resource definition) with 4 GB of memory.

This quick-start guide has been tested with the following Linux distributions:

- SUSE Linux Enterprise Server (SLES) 12

Step 1: Create the Docker container

1. Switch to root user for this exercise.
> sudo –i

```
linux1@dockerspark2:~> sudo -i
dockerspark2:~ #
```

2. Start the Docker service on your Linux system.
systemctl start docker

```
dockerspark2:~ # systemctl start docker
dockerspark2:~ #
```

3. Download the Spark Docker file from the repository.
docker pull brunswickheads/spark-1.5.2-s390x

```
dockerspark2:/ # docker pull brunswickheads/spark-1.5.2-s390x
Using default tag: latest
latest: Pulling from brunswickheads/spark-1.5.2-s390x
b5257eca2dd8: Pull complete
78745e6c93f9: Pull complete
b6a646867400: Pull complete
b89ba56b8f32: Pull complete
ca77adadcce6: Pull complete
a8e5f3181b12: Pull complete
Digest: sha256:c55f9185e571fbbccd6a2a2c126aed51700a441c67c8368da24b060612c0be7a
Status: Downloaded newer image for brunswickheads/spark-1.5.2-s390x:latest
```

4. Launch a Spark Docker container and open the ports required for Spark. Start a bash shell within the container.

docker run -p 8080:8080 -p 8081:8081 -p 7077:7077 -p 6066:6066 -i -t brunswickheads/spark-1.5.2-s390x /bin/bash

```
dockerspark2:~ # docker run -p 8080:8080 -p 8081:8081 -p 7077:7077 -p 6066:6066 -i -t brunswickheads/spark-1.5.2-s390x /bin/bash
bash-4.2#
```

5. Exit the bash shell.
   # exit

```
bash-4.2# exit
exit
dockerspark2:/ #
```

6. The docker container is stopped when you exit the bash shell.

docker ps

```
dockerspark2:~ # docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
```

7. Switch to the Docker containers folder.
   # cd /data/docker/containers/

```
dockerspark2:/ # cd /data/docker/containers/
dockerspark2:/data/docker/containers #
```

8. List the contents of the folder to view your Docker containers.

ls

```
dockerspark2:/data/docker/containers # ls
422caf42253098ad225e2cf38ba91c335f04be6bb60f8733941b0ead4db35710
dockerspark2:/data/docker/containers #
```

Step 2: Start the Apache Spark shell

1. Restart your Docker container using the first three characters of your Docker container directory.

docker start 422

```
dockerspark2:/data/docker/containers # docker start 422
422
dockerspark2:/data/docker/containers #
```

2. Check the status of your running Docker container.

docker ps

Note your Docker container id.

```
dockerspark2:/data/docker/containers # docker ps
CONTAINER ID        IMAGE                                COMMAND             CREATED             STATUS              PORTS
                                                 NAMES
422caf422530        brunswickheads/spark-1.5.2-s390x     "/bin/bash"         45 minutes ago      Up 6 minutes        0.0.0.0:6066->6066/tcp, 0.0.0.0:7077->707
7/tcp, 0.0.0.0:8080-8081->8080-8081/tcp   sharp_mccarthy
dockerspark2:/data/docker/containers #
```

3. Attach to your running Docker container to restart the bash shell.
   a) # docker attach 422
   b) Hit ‘enter’ or ‘return’ again

```
dockerspark2:/data/docker/containers # docker attach 422

bash-4.2#
```

4. Start an interactive Spark shell.

/opt/ibm/spark/bin/spark-shell

```
bash-4.2# /opt/ibm/spark/bin/spark-shell
log4j:WARN No appenders could be found for logger (org.apache.hadoop.metrics2.lib.MutableMetricsFactor
y).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Using Spark's repl log4j profile: org/apache/spark/log4j-defaults-repl.properties
To adjust logging level use sc.setLogLevel("INFO")
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.5.2
      /_/

Using Scala version 2.10.4 (IBM J9 VM, Java 1.8.0)
Type in expressions to have them evaluated.
Type :help for more information.
16/01/21 15:17:54 WARN MetricsSystem: Using default name DAGScheduler for source because spark.app.id
is not set.
Spark context available as sc.
16/01/21 15:17:56 WARN Connection: BoneCP specified but not present in CLASSPATH (or one of dependenci
es)
16/01/21 15:17:56 WARN Connection: BoneCP specified but not present in CLASSPATH (or one of dependenci
es)
16/01/21 15:18:00 WARN ObjectStore: Version information not found in metastore. hive.metastore.schema.
verification is not enabled so recording the schema version 1.2.0
16/01/21 15:18:00 WARN ObjectStore: Failed to get database default, returning NoSuchObjectException
16/01/21 15:18:01 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... usi
ng builtin-java classes where applicable
16/01/21 15:18:02 WARN Connection: BoneCP specified but not present in CLASSPATH (or one of dependenci
es)
16/01/21 15:18:02 WARN Connection: BoneCP specified but not present in CLASSPATH (or one of dependenci
es)
16/01/21 15:18:05 WARN ObjectStore: Version information not found in metastore. hive.metastore.schema.
verification is not enabled so recording the schema version 1.2.0
16/01/21 15:18:05 WARN ObjectStore: Failed to get database default, returning NoSuchObjectException
SQL context available as sqlContext.

scala>
```

Step 3: Run the interactive Apache Spark shell example

1. Test the Spark read-eval-print loop (REPL).
   a) Create a collection of 20 entries

scala> val data = 1 to 20

```
scala> val data = 1 to 20
data: scala.collection.immutable.Range.Inclusive = Range(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)
```

   b) Create a resilient distributed dataset (RDD)

scala> val distdata = sc.parallelize(data)

```
scala> val distdata = sc.parallelize(data)
distdata: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:23
```

   c) Select and return the even values

scala> distdata.filter(_ % 2 == 0).collect()

```
scala> distdata.filter(_ % 2 == 0).collect()
res0: Array[Int] = Array(2, 4, 6, 8, 10, 12, 14, 16, 18, 20)
```

2. Exit from the Spark REPL.
scala> :q

```
scala> :q
Stopping spark context.
bash-4.2#
```

Step 4: Run the Apache Spark master and worker nodes example

1. Switch to the Spark installation directory.
cd opt/ibm/spark/

```
bash-4.2# cd opt/ibm/spark/
bash-4.2#
```

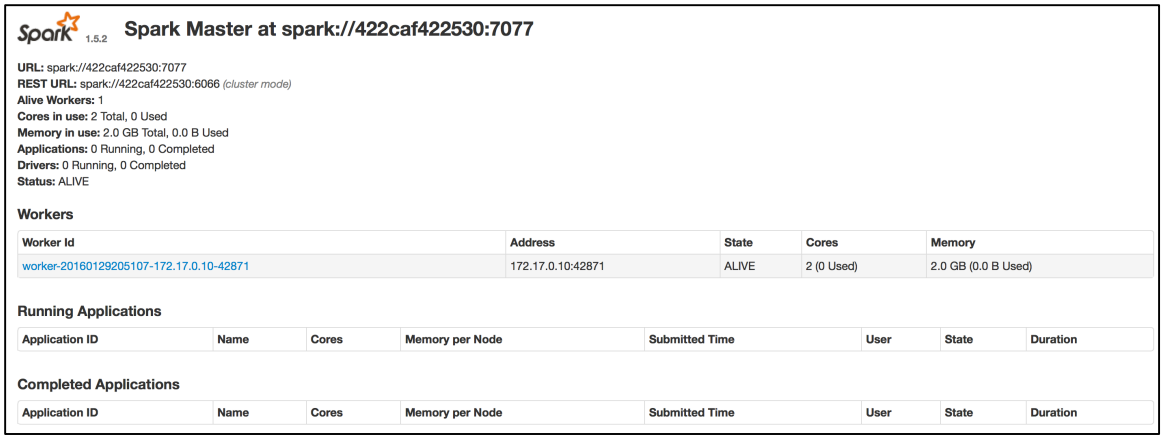
2. Start the Spark Master server.
./sbin/start-master.sh

```
bash-4.2# ./sbin/start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /opt/ibm/spark/sbin/../logs/spark--org.apac
he.spark.deploy.master.Master-1-422caf422530.out
bash-4.2#
```

3. Start a Spark Worker in your Docker container and allocate 2GB of memory. Use your Docker container id.
./sbin/start-slave.sh -m 2G spark://<Docker container id>:7077

```
bash-4.2# ./sbin/start-slave.sh -m 2G spark://422caf422530:7077
starting org.apache.spark.deploy.worker.Worker, logging to /opt/ibm/spark/sbin/../logs/spark--org.apac
he.spark.deploy.worker.Worker-1-422caf422530.out
bash-4.2#
```

4. Using a browser, go to <your Linux ip address>:8080 to verify the Spark GUI is operational.



Spark 1.5.2 **Spark Master at spark://422caf422530:7077**

URL: spark://422caf422530:7077
REST URL: spark://422caf422530:6066 *(cluster mode)*
Alive Workers: 1
Cores in use: 2 Total, 0 Used
Memory in use: 2.0 GB Total, 0.0 B Used
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20160129205107-172.17.0.10-42871	172.17.0.10:42871	ALIVE	2 (0 Used)	2.0 GB (0.0 B Used)

Running Applications

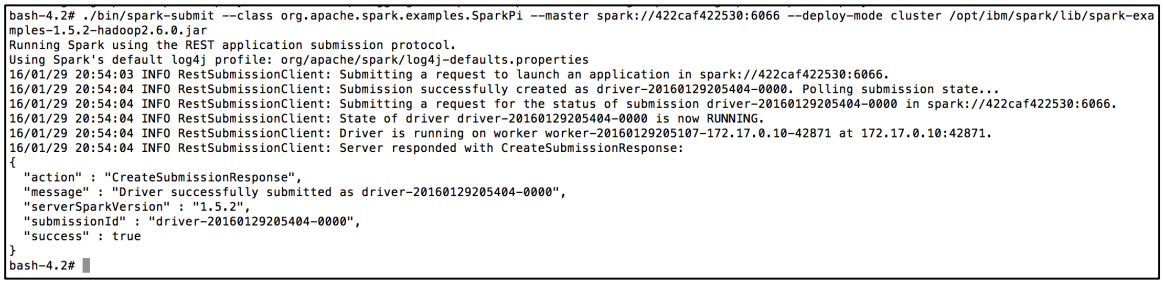
Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration

Completed Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration

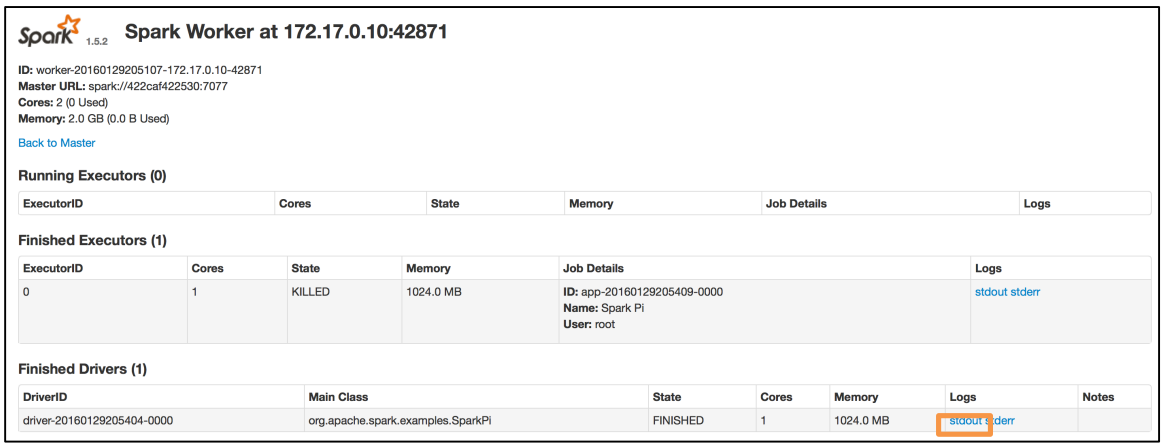
5. Run an example scala program that estimates the value of Pi. Use your Docker container id.

./bin/spark-submit --class org.apache.spark.examples.SparkPi --master spark://<Docker container id>:6066 --deploy-mode cluster /opt/ibm/spark/lib/spark-examples-1.5.2-hadoop2.6.0.jar



```
bash-4.2# ./bin/spark-submit --class org.apache.spark.examples.SparkPi --master spark://422caf422530:6066 --deploy-mode cluster /opt/ibm/spark/lib/spark-exa
mples-1.5.2-hadoop2.6.0.jar
Running Spark using the REST application submission protocol.
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
16/01/29 20:54:03 INFO RestSubmissionClient: Submitting a request to launch an application in spark://422caf422530:6066.
16/01/29 20:54:04 INFO RestSubmissionClient: Submission successfully created as driver-20160129205404-0000. Polling submission state...
16/01/29 20:54:04 INFO RestSubmissionClient: Submitting a request for the status of submission driver-20160129205404-0000 in spark://422caf422530:6066.
16/01/29 20:54:04 INFO RestSubmissionClient: State of driver driver-20160129205404-0000 is now RUNNING.
16/01/29 20:54:04 INFO RestSubmissionClient: Driver is running on worker worker-20160129205107-172.17.0.10-42871 at 172.17.0.10:42871.
16/01/29 20:54:04 INFO RestSubmissionClient: Server responded with CreateSubmissionResponse:
{
  "action" : "CreateSubmissionResponse",
  "message" : "Driver successfully submitted as driver-20160129205404-0000",
  "serverSparkVersion" : "1.5.2",
  "submissionId" : "driver-20160129205404-0000",
  "success" : true
}
bash-4.2#
```

6. Using a browser, go to <your Linux ip address>:8081 to see the output. Click on the stdout link for your DriverID.



Spark 1.5.2 **Spark Worker at 172.17.0.10:42871**

ID: worker-20160129205107-172.17.0.10-42871
Master URL: spark://422caf422530:7077
Cores: 2 (0 Used)
Memory: 2.0 GB (0.0 B Used)

Back to Master

Running Executors (0)

ExecutorID	Cores	State	Memory	Job Details	Logs

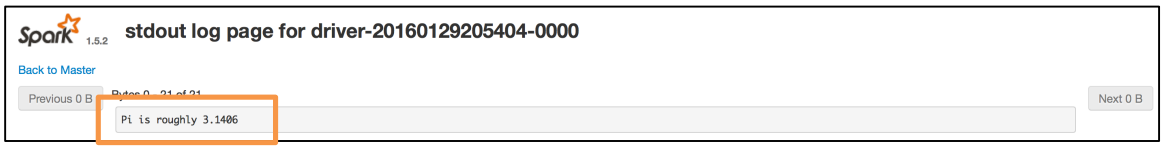
Finished Executors (1)

ExecutorID	Cores	State	Memory	Job Details	Logs
0	1	KILLED	1024.0 MB	**ID:** app-20160129205409-0000 **Name:** Spark Pi **User:** root	stdout stderr

Finished Drivers (1)

DriverID	Main Class	State	Cores	Memory	Logs	Notes
driver-20160129205404-0000	org.apache.spark.examples.SparkPi	FINISHED	1	1024.0 MB	stdout stderr	

7. Verify the result.



8. Exit the bash shell.
exit

```
bash-4.2# exit
exit
dockerspark2:/data/docker/containers #
```

9. The docker container is now stopped.
docker ps

```
dockerspark2:/data/docker/containers # docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
dockerspark2:/data/docker/containers #
```

Reference

Links

https://docs.docker.com/engine/userguide/intro/

https://hub.docker.com/r/brunswickheads/spark-1.5.2-s390x/

https://spark.apache.org/docs/latest/quick-start.html