Overview

Spark instances are created on the Data Science and Machine Learning Platform at UCSD.

Start a Spark Cluster

1. Navigate to datahub.ucsd.edu
2. Select your course and click “Launch Environment”

Launching the course will automatically spin up 1 spark master and 2 spark workers. Give the server some time to spin up the spark cluster.

3. Confirm that the cluster has started by opening a jupyter terminal and running the command “kubectl get pods”

```
77818@dsmlp-jupyter-wuykimpang:~$ kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
dsmlp-jupyter-wuykimpang    1/1     Running   0          2m37s
spark-master-0              1/1     Running   0          107s
spark-worker-0              1/1     Running   0          107s
spark-worker-1              1/1     Running   0          45s
77818@dsmlp-jupyter-wuykimpang:~$
```

Pods will start off with a status of 0/1 under column “READY”. Please wait until you see status “1/1

4. Confirm that the master has been elected by running “kubectl logs spark-master-0”

```
77818@dsmlp-jupyter-wuykimpang:~$ kubectl logs spark-master-0
 21:34:14.62
 21:34:14.62 Welcome to the Bitnami spark container
 21:34:14.63 Subscribe to project updates by watching https://github.com/bitnami/containers
 21:34:14.63 Submit issues and feature requests at https://github.com/bitnami/containers/issues
 21:34:14.63
 21:34:14.65 INFO  ==> ** Starting Spark setup **
realpath: /bitnami/spark/conf: No such file or directory
 21:34:14.70 INFO  ==> Generating Spark configuration file...
find: '/docker-entrypoint-initdb.d/': No such file or directory
 21:34:14.72 INFO  ==> No custom scripts in /docker-entrypoint-initdb.d
 21:34:14.73 INFO  ==> ** Spark setup finished! **

 21:34:14.77 INFO  ==> ** Starting Spark in master mode **
starting org.apache.spark.deploy.master.Master, logging to /opt/bitnami/spark/logs/spark--org.apache.spark.deploy.master.Master
-1-spark-master-0.out
Spark Command: /opt/bitnami/java/bin/java -cp /opt/bitnami/spark/conf/:/opt/bitnami/spark/jars/* -Xmx1g org.apache.spark.deploy
.master.Master --host spark-master-0.spark-headless.wuykimpang.svc.cluster.local --port 7077 --webui-port 8080
========================================
Using Spark's default log4j profile: org/apache/spark/log4j2-defaults.properties
22/11/01 21:34:18 INFO Master: Started daemon with process name: 44@spark-master-0
22/11/01 21:34:18 INFO SignalUtils: Registering signal handler for TERM
22/11/01 21:34:18 INFO SignalUtils: Registering signal handler for HUP
22/11/01 21:34:18 INFO SignalUtils: Registering signal handler for INT
22/11/01 21:34:19 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes w
here applicable
22/11/01 21:34:20 INFO SecurityManager: Changing view acls to: spark
22/11/01 21:34:20 INFO SecurityManager: Changing modify acls to: spark
22/11/01 21:34:20 INFO SecurityManager: Changing view acls groups to:
22/11/01 21:34:20 INFO SecurityManager: Changing modify acls groups to:
22/11/01 21:34:20 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permission
s: Set(spark); groups with view permissions: Set(); users  with modify permissions: Set(spark); groups with modify permissions:
 Set()
22/11/01 21:34:21 INFO Utils: Successfully started service 'sparkMaster' on port 7077.
22/11/01 21:34:21 INFO Master: Starting Spark master at spark://spark-master-0.spark-headless.wuykimpang.svc.cluster.local:7077
22/11/01 21:34:21 INFO Master: Running Spark version 3.3.1
22/11/01 21:34:21 INFO Utils: Successfully started service 'MasterUI' on port 8080.
22/11/01 21:34:21 INFO MasterWebUI: Bound MasterWebUI to 0.0.0.0, and started at http://spark-master-0.spark-headless.wuykimpan
g.svc.cluster.local:8080
22/11/01 21:34:22 INFO Master: I have been elected leader! New state: ALIVE
22/11/01 21:34:52 INFO Master: Registering worker 10.37.32.61:36637 with 2 cores, 2.7 GiB RAM
77818@dsmlp-jupyter-wuykimpang:~$
```

See the last logs “I have been elected leader!”

5. Once the cluster has been created, navigate back to the notebook tree tab (https://datahub.ucsd.edu/hub/<username>/tree?” and select New > spark-driver

Alive Workers: 2
Cores in use: 4 Total, 0 Used
Memory in use: 5.4 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (3)

Worker Id	Address	State	Cores	Memory	Resources
worker-20221103171255-10.37.32.25-43412	10.37.32.25:43412	DEAD	2 (0 Used)	2.7 GiB (0.0 B Used)	
worker-20221103171327-10.38.0.33-43495	10.38.0.33:43495	ALIVE	2 (0 Used)	2.7 GiB (0.0 B Used)	
worker-20221103171610-10.37.32.25-33064	10.37.32.25:33064	ALIVE	2 (0 Used)	2.7 GiB (0.0 B Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

6. There's a test jupyter notebook that confirms that the spark cluster works located at /opt/sanity_check.ipynb. You can confirm that you can use pyspark with this cluster by running that notebook. See optional instruction steps below for more details
7. (OPTIONAL) Go back to the jupyter terminal and run the command "cp /opt/sanity_check.ipynb ~"
8. (OPTIONAL) Open the jupyter notebook and execute it. If all goes well, you'll see the output like below

In [1]:
```
import os
import pyspark
from pyspark import SparkContext

def get_local_ip():
    import socket
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s.connect(("8.8.8.8", 80))
    ip = s.getsockname()[0]
    s.close()
    return ip

os.environ['SPARK_LOCAL_IP']="" #driver_host
driver_host = get_local_ip()
conf = pyspark.SparkConf()
conf.setAppName("spark test")
conf.setMaster('spark://spark-master-svc:7077')
conf.set("spark.blockmanager.port", "50002")
conf.set("spark.driver.bindAddress", driver_host)
conf.set("spark.driver.host", driver_host)
conf.set("spark.driver.port", "50500")
conf.set("spark.cores.max", "2")
conf.set("spark.executor.memory", "512m")
conf.set('spark.authenticate', False)

sc = SparkContext(conf=conf)
rdd = sc.parallelize([i for i in range(100)])
rdd.count()
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).

22/11/01 21:43:06 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-ja
va classes where applicable
```

Out[1]:
```
100
```

Troubleshooting

Why don’t I see any or just a single spark worker at the spark dashboard?

When the workers try to join the cluster, they look for the master node. If the master node is still being setup, they’ll ping the master and try to reconnect up to 7 times. If they’ve exceeded their connection retry times, the pod will give up and fail to connect.

To fix this, and do the following:

1. check your spark dashboard and get the IP of the connected worker pod (if any)

   Workers (1)

Worker Id
worker-20221103171327-10.38.0.33-43495

2. Open a jupyter terminal and run `kubectl get pods -o wide`
3. Find the **worker pod** whose Cluster IP is not on the list. Copy the pod name to your clipboard
4. Run kubectl delete pod <worker_pod_name>

The above will restart the worker pods and will have a better chance of successfully joining the master pod if it’s been elected.

You can confirm that it joined the master pod by looking at the master pods logs with the following command: “kubectl logs <master_pod_name>”. See if there's an output towards the bottom that says “Registering worker…”. This may take some time as well since the worker pod must start up. Alternatively, you may also simply check the spark dashboard