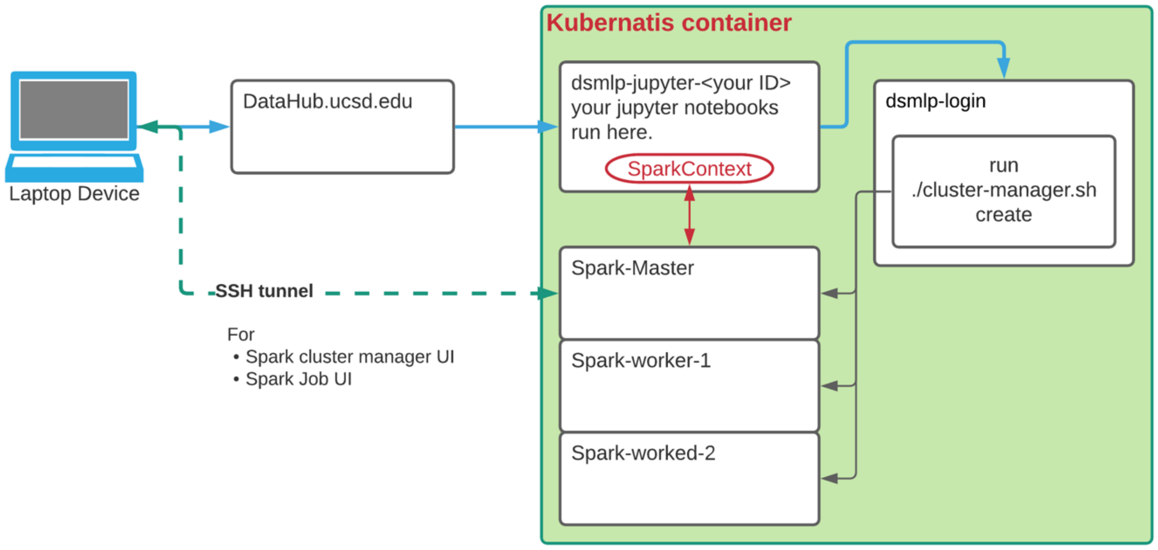
Spark cluster for DSC291

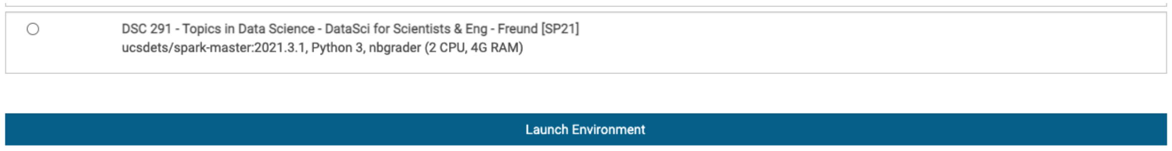
Overview

The spark instances for this class are hosted on a machine called DSMLP. Each student is allocated a kubernetes container. Each student logs into datahub.ucsd.edu selects the DSE291 package and is forwarded to an account called dsmlp-jupyter-<user ID>. This is the account where the student will spend most of her time. The interface exposed to the user is a jupyter notebook front page, which can be used to open new notebooks or to start a terminal on the machine. In order to create a cluster the user uses the terminal to connect, through ssh, to a computer called dsmlp-login, in that account it uses a script called ./cluster-manager.sh to create the cluster (details below). The cluster consists of two workers and one master node.
Once the cluster is up, the user can connect with it from a notebook on dsmp-jupyter through the pyspark object called "SparkContext". In addition, the user can open an ssh tunnel between their laptop and the spark master. Using this tunnel the user can open user interfaces for monitoring the cluster.

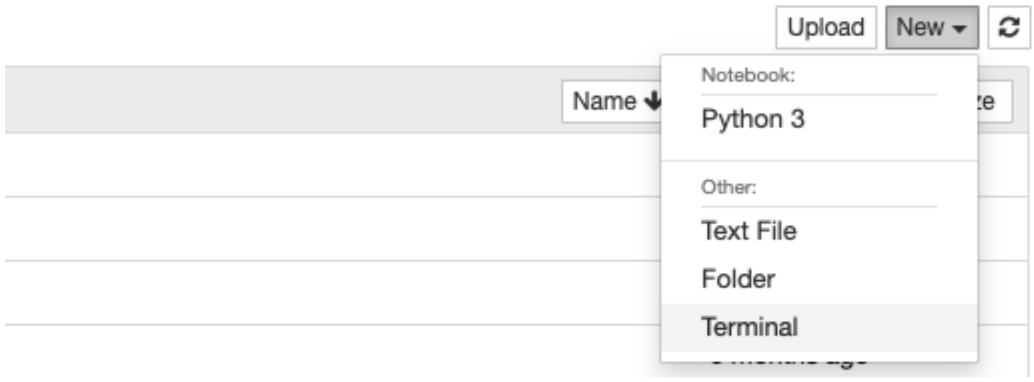


Starting a spark cluster

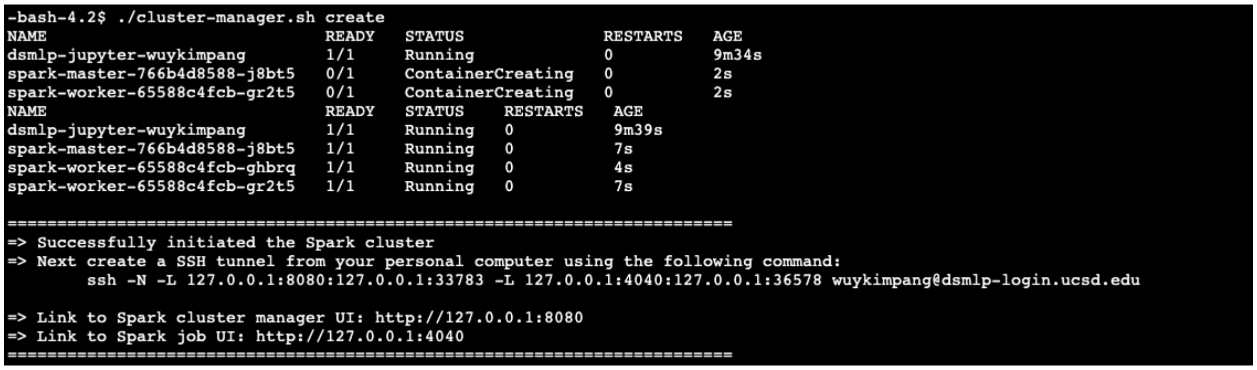
1. Login to datahub.ucsd.edu
2. Select the "DSC 291" environment and click the "Launch Environment" button


DSC 291 - Topics in Data Science - DataSci for Scientists & Eng - Freund [SP21]
ucsdets/spark-master:2021.3.1, Python 3, nbgrader (2 CPU, 4G RAM)

3. Once your notebook server is running, select “New > Terminal”



4. Switch to the terminal tab in your web browser
5. Once at the terminal, run the command “**ssh <username>@dsmlp-login**”. Replace <username> with your username.
6. Verify that you are working on “**dsmlp-login”** by using the command “**hostname”**
7. Download the scripts for your course by running the command **git clone https://github.com/ucsd-ets/dsc291-spark-cluster.git**
8. You should now see a directory called **dsc291-spark-cluster** within your terminal (run the command “**ls**” to list the contents of the directory).
9. Change into the directory by running the command “**cd dsc291-spark-cluster**”
10. To start your spark cluster, run the command “**./cluster-manager.sh create**”. You should see output on your screen similar to the example below:

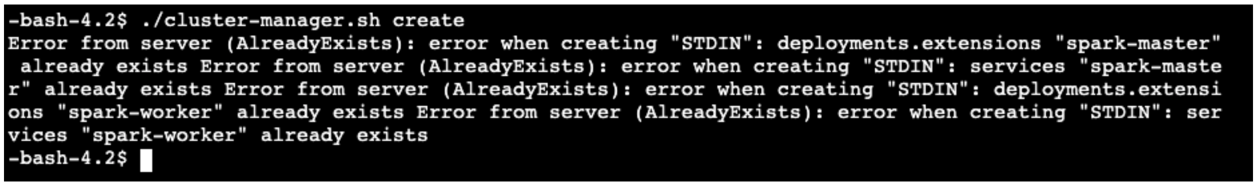


```
-bash-4.2$ ./cluster-manager.sh create
NAME                            READY   STATUS              RESTARTS   AGE
dsmlp-jupyter-wuykimpang        1/1     Running             0          9m34s
spark-master-766b4d8588-j8bt5   0/1     ContainerCreating   0          2s
spark-worker-65588c4fcb-gr2t5   0/1     ContainerCreating   0          2s
NAME                            READY   STATUS    RESTARTS   AGE
dsmlp-jupyter-wuykimpang        1/1     Running   0          9m39s
spark-master-766b4d8588-j8bt5   1/1     Running   0          7s
spark-worker-65588c4fcb-ghbrq   1/1     Running   0          4s
spark-worker-65588c4fcb-gr2t5   1/1     Running   0          7s

=========================================================================
=> Successfully initiated the Spark cluster
=> Next create a SSH tunnel from your personal computer using the following command:
        ssh -N -L 127.0.0.1:8080:127.0.0.1:33783 -L 127.0.0.1:4040:127.0.0.1:36578 wuykimpang@dsmlp-login.ucsd.edu

=> Link to Spark cluster manager UI: http://127.0.0.1:8080
=> Link to Spark job UI: http://127.0.0.1:4040
=========================================================================
```

A copy of this output is stored in the file **cluster-manager.log**

11. A common problem:

If you see output like the following:



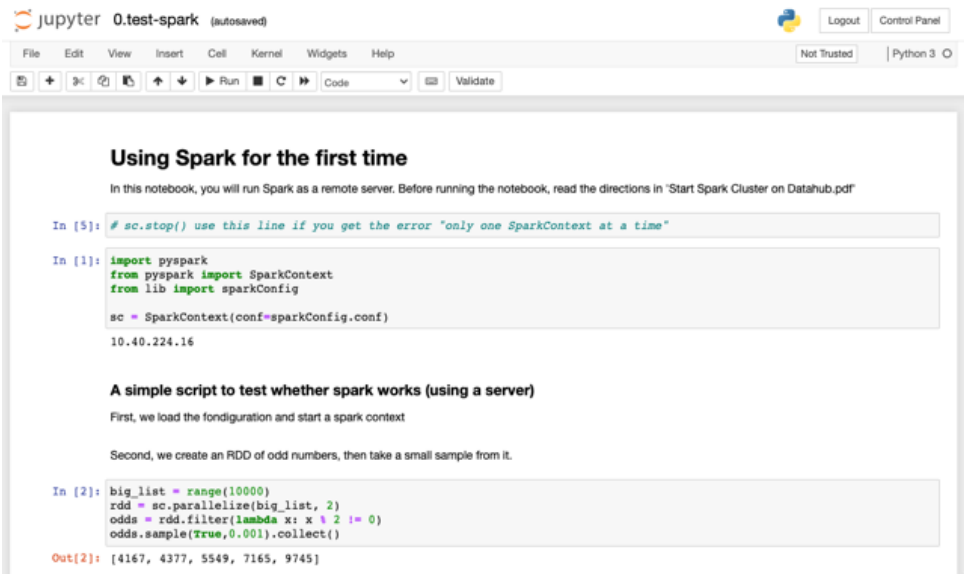
```
-bash-4.2$ ./cluster-manager.sh create
Error from server (AlreadyExists): error when creating "STDIN": deployments.extensions "spark-master"
 already exists Error from server (AlreadyExists): error when creating "STDIN": services "spark-maste
r" already exists Error from server (AlreadyExists): error when creating "STDIN": deployments.extensi
ons "spark-worker" already exists Error from server (AlreadyExists): error when creating "STDIN": ser
vices "spark-worker" already exists
-bash-4.2$ █
```

That means that your spark cluster is already running and you may start using it. You can also recreate the cluster by running the command “**./cluster-manager.sh delete**” and then the command “**./cluster-manager.sh create**”.

12. You are now ready to run your first spark notebook.

Starting your first pyspark notebook

1. To access pyspark in jupyter, open the datahub.ucsd.edu in your browser. You should automatically be connected to a jupyter notebook through which you can connect to your spark cluster. Select the directory **dsc291-spark-cluster**, then select the notebook **0.test-spark.ipynb**
2. Execute the second and 3rd cells (the 1st is there to help in case you already have a sparkContext) If both generate output similar to the one seen blow, and don't report errors then, **congratulations, you managed to create a spark cluster and run a notebook using it!**


jupyter 0.test-spark (autosaved)
Logout
Control Panel
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
Validate

Using Spark for the first time

In this notebook, you will run Spark as a remote server. Before running the notebook, read the directions in 'Start Spark Cluster on Datahub.pdf'

```
In [5]: # sc.stop() use this line if you get the error "only one SparkContext at a time"
```

```
In [1]: import pyspark
        from pyspark import SparkContext
        from lib import sparkConfig

        sc = SparkContext(conf=sparkConfig.conf)

        10.40.224.16
```

A simple script to test whether spark works (using a server)

First, we load the fondiguration and start a spark context

Second, we create an RDD of odd numbers, then take a small sample from it.

```
In [2]: big_list = range(10000)
        rdd = sc.parallelize(big_list, 2)
        odds = rdd.filter(lambda x: x % 2 != 0)
        odds.sample(True,0.001).collect()
Out[2]: [4167, 4377, 5549, 7165, 9745]
```

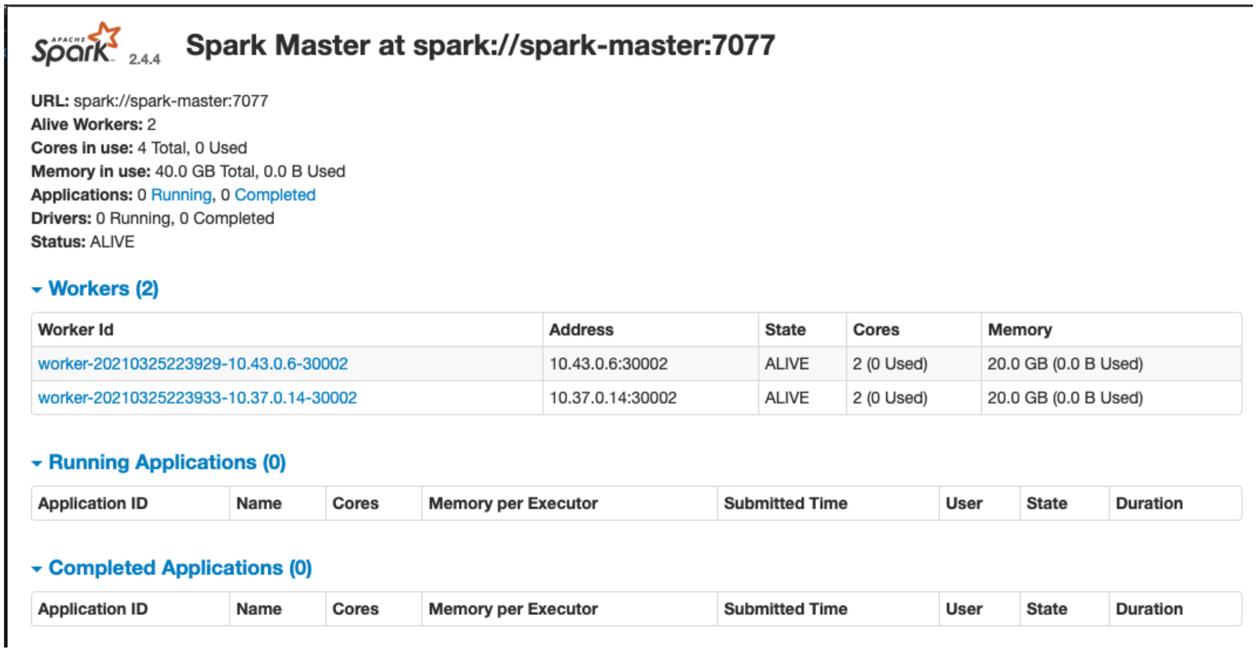
Working with Class notebooks

1. Using the terminal, create a symbolic link from your home directory to the location where the notebooks are shared:
   ln -s /datasets/ds291-sp21-A00-public/DSC291_Notebooks/ ~/Public_Notebooks
2. Create a directory in which to store your own copies of the notebooks:
   mkdir ~/My-Notebooks
3. The shared notebooks of the class are placed in subdirectories of **Public_Notebooks** . There is a separate directory for each class or group of classes and for each HW, The TA will add new directories as the quarter progresses.. To work on these notebooks copy them to **My_Notebooks**

4. We recommend that you use git to maintain the versions of notebooks in **My_Notebooks.** That would decrease the chance that you lose a version with changes that you made. Note that you do NOT want to store large data files in git.
5. You might want to create a github directory for your local git. In that case we ask you use a private repository.

The spark administration web interface

1. Open a new terminal on your local computer (**not in Datahub**) and paste the generated ssh command into it (command right below “Next create a SSH tunnel…”). This will open a tunnel between datahub servers and your local computer. Leave this terminal open to keep the tunnel open. ***Note: make sure you’re connected to UCSD’s VPN***
2. Open a new tab in your browser and navigate to http://127.0.0.1:8080. The Apache Spark dashboard will be there. See example below:



Spark Master at spark://spark-master:7077

URL: spark://spark-master:7077
Alive Workers: 2
Cores in use: 4 Total, 0 Used
Memory in use: 40.0 GB Total, 0.0 B Used
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (2)

Worker Id	Address	State	Cores	Memory
worker-20210325223929-10.43.0.6-30002	10.43.0.6:30002	ALIVE	2 (0 Used)	20.0 GB (0.0 B Used)
worker-20210325223933-10.37.0.14-30002	10.37.0.14:30002	ALIVE	2 (0 Used)	20.0 GB (0.0 B Used)

Running Applications (0)

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

Completed Applications (0)

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

3. You can now close the terminal. ***Note: your cluster will only be active for 3 hours. You’ll have to recreate it starting from step 3 in case it shuts down while you’re working with it.***

Common problems

Different output at step 9