BigData Analytics with Spark and Hadoop at OSC

06/06/2018
OSC workshop

Shameema Oottikkal
Data Application Engineer
Ohio Supercomputer Center
email:soottikkal@osc.edu

What is BigData

Bigdata is an evolving term that describes any voluminous amount of structured and unstructured data that has the potential to be mined for information.

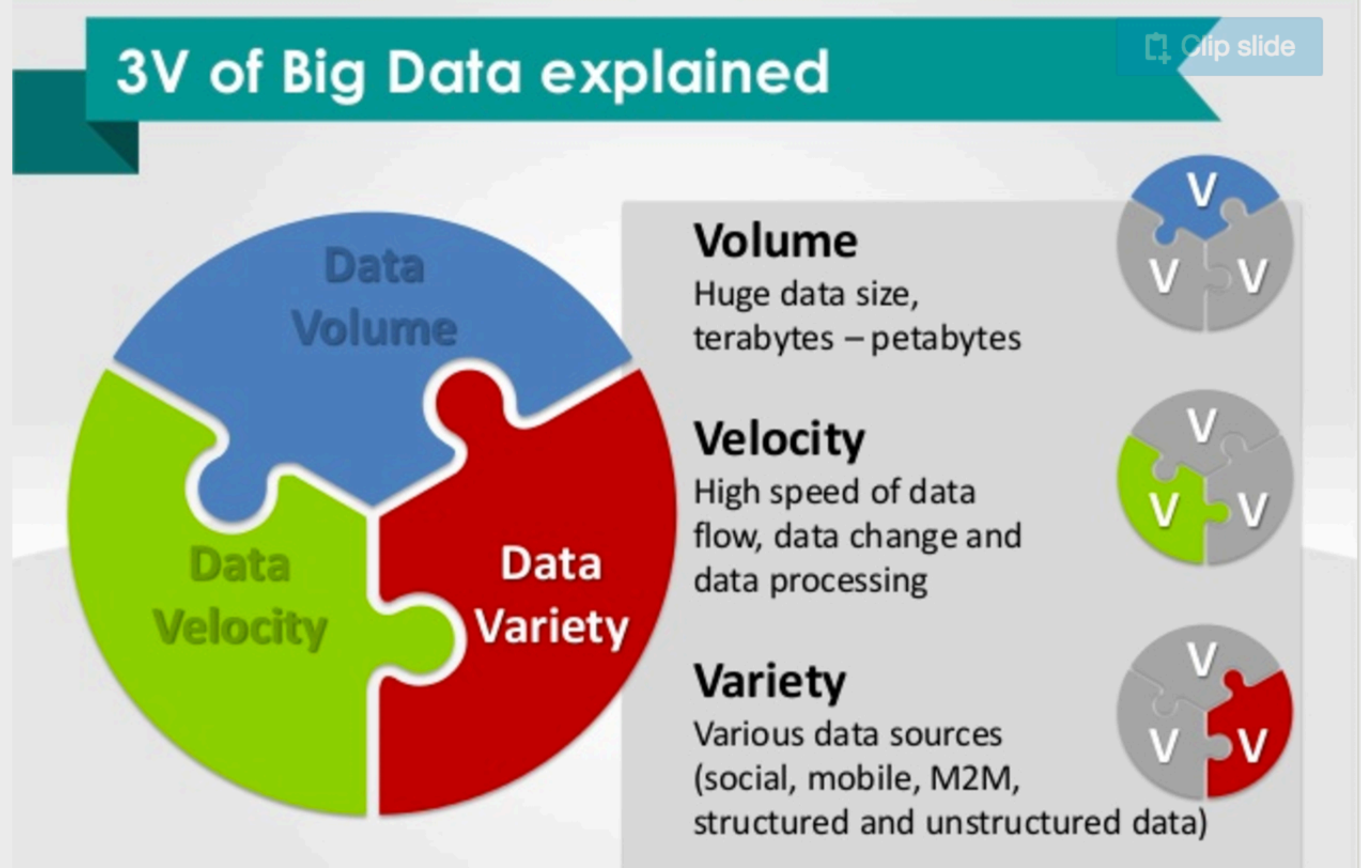
Bigdata generates value from the storage and processing of very large quantities of digital information that cannot be analyzed with traditional computing techniques

Helps to solve new problem or old problem in a better way

Fried Broekhof
November 2013

RAM
Real time Adaptive Marketing

The 3V of Big Data

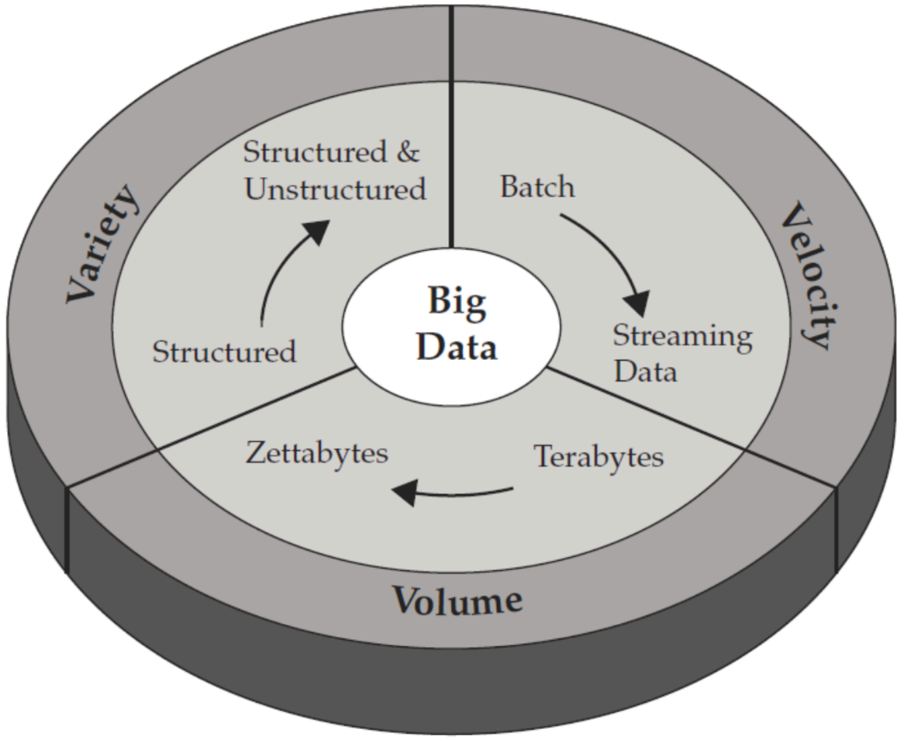


3V of Big Data explained

Volume
Huge data size, terabytes – petabytes

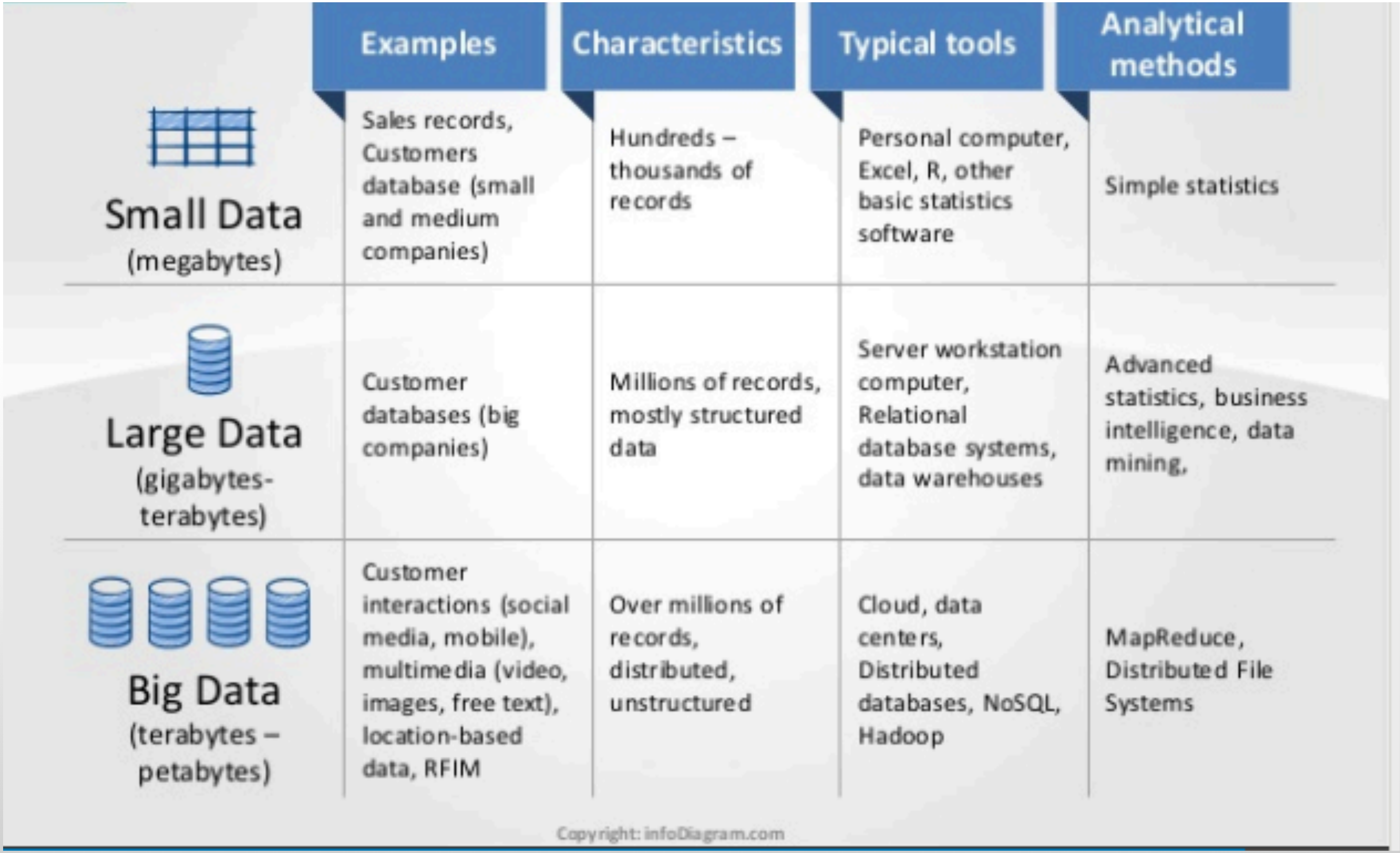
Velocity
High speed of data flow, data change and data processing

Variety
Various data sources (social, mobile, M2M, structured and unstructured data)



- Key enablers for the growth of “Big Data” are:
  - Increase of storage capacities
  - Increase of processing power
  - Availability of data

Data Analytical Tools

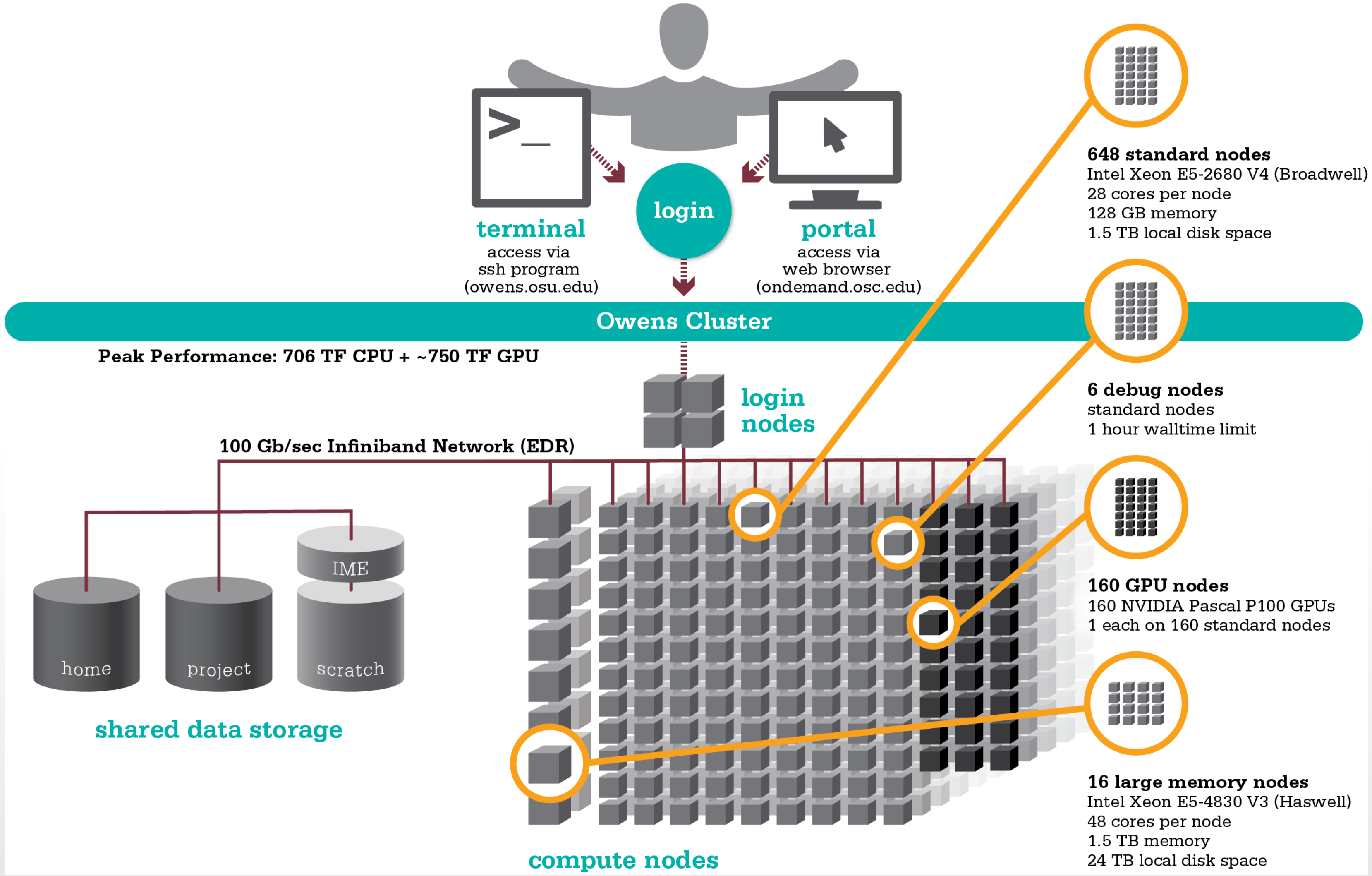


	Examples	Characteristics	Typical tools	Analytical methods
Small Data (megabytes)	Sales records, Customers database (small and medium companies)	Hundreds – thousands of records	Personal computer, Excel, R, other basic statistics software	Simple statistics
Large Data (gigabytes-terabytes)	Customer databases (big companies)	Millions of records, mostly structured data	Server workstation computer, Relational database systems, data warehouses	Advanced statistics, business intelligence, data mining,
Big Data (terabytes – petabytes)	Customer interactions (social media, mobile), multimedia (video, images, free text), location-based data, RFIM	Over millions of records, distributed, unstructured	Cloud, data centers, Distributed databases, NoSQL, Hadoop	MapReduce, Distributed File Systems

Copyright: infoDiagram.com

Data Analytical nodes@OSC

Owens' data analytics environment is comprised of 16 nodes, each with 48 CPU cores, 1.5TB of RAM and 24TB of local disk.



648 standard nodes
Intel Xeon E5-2680 V4 (Broadwell)
28 cores per node
128 GB memory
1.5 TB local disk space

6 debug nodes
standard nodes
1 hour walltime limit

160 GPU nodes
160 NVIDIA Pascal P100 GPUs
1 each on 160 standard nodes

16 large memory nodes
Intel Xeon E5-4830 V3 (Haswell)
48 cores per node
1.5 TB memory
24 TB local disk space

$HOME:
500GB/per user
Backed up daily
Permanent storage

Local disk:$TMPDIR
1.5TB or 24TB
Not backed up
Temporary storage

/fs/scratch:
1200TB
Not backed up
Temporary storage

/fs/project:
Upon request
1-5TB
Backed up daily
1-3 years

OSC OnDemand ondemand.osc.edu

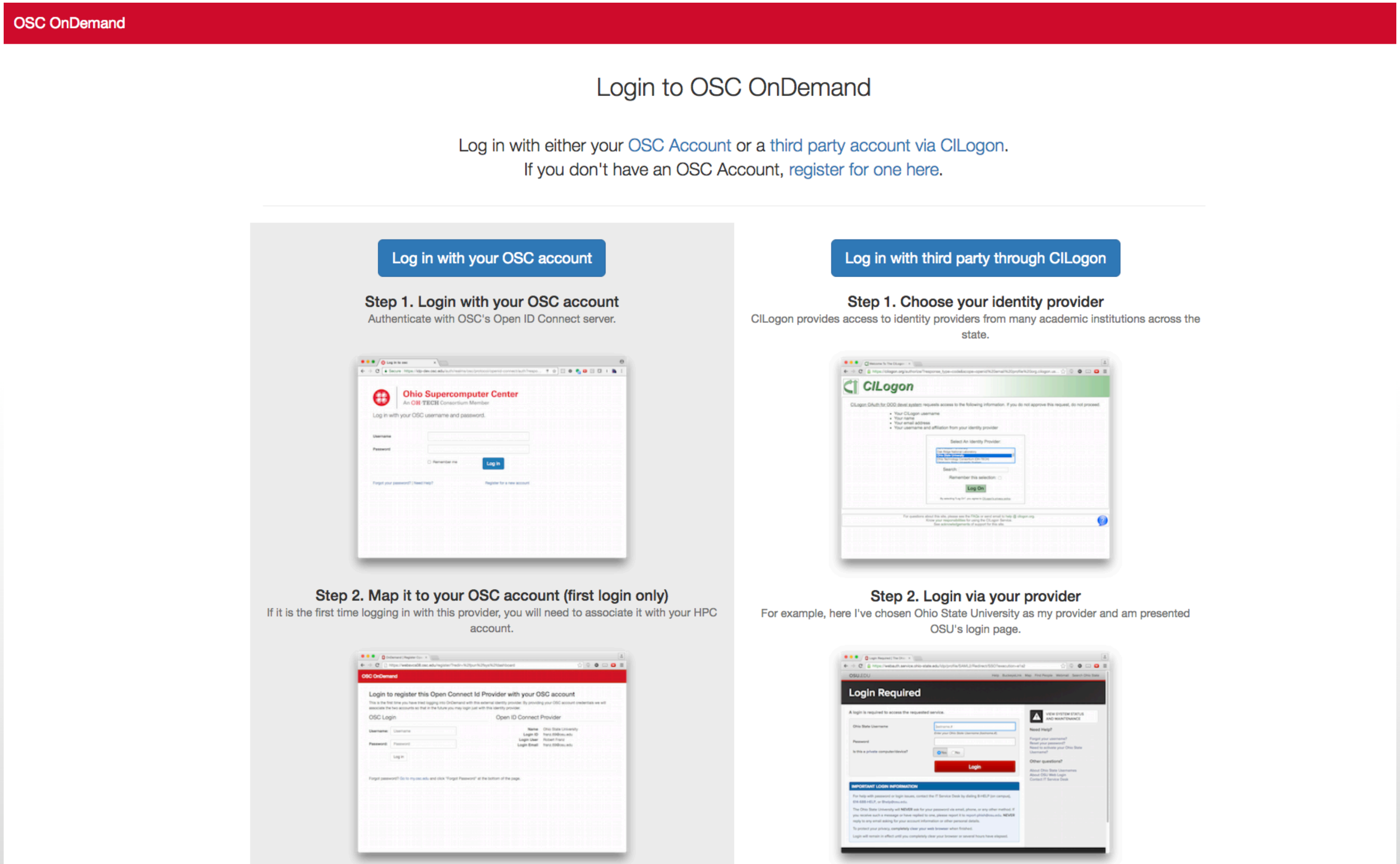
- 1: User Interface
  - Web based
    - Usable from computers, tablets, smartphones
    - Zero installation
  - Single point of entry
    - User needs three things
      - ondemand.osc.edu
      - OSC Username
      - OSC Password
    - Connected to all resources at OSC

- 2: Interactive Services
  - File Access
  - Job Management
  - Visualization Apps
    - Desktop access
    - Single-click apps (Abaqus, Ansys, Comsol, Paraview)
  - Terminal Access

Tutorial available at

osc.edu/ondemand

Go to https://ondemand.osc.edu/



OSC OnDemand

Login to OSC OnDemand

Log in with either your OSC Account or a third party account via CILogon.
If you don't have an OSC Account, register for one here.

Log in with your OSC account

Step 1. Login with your OSC account

Authenticate with OSC's Open ID Connect server.

Step 2. Map it to your OSC account (first login only)

If it is the first time logging in with this provider, you will need to associate it with your HPC account.

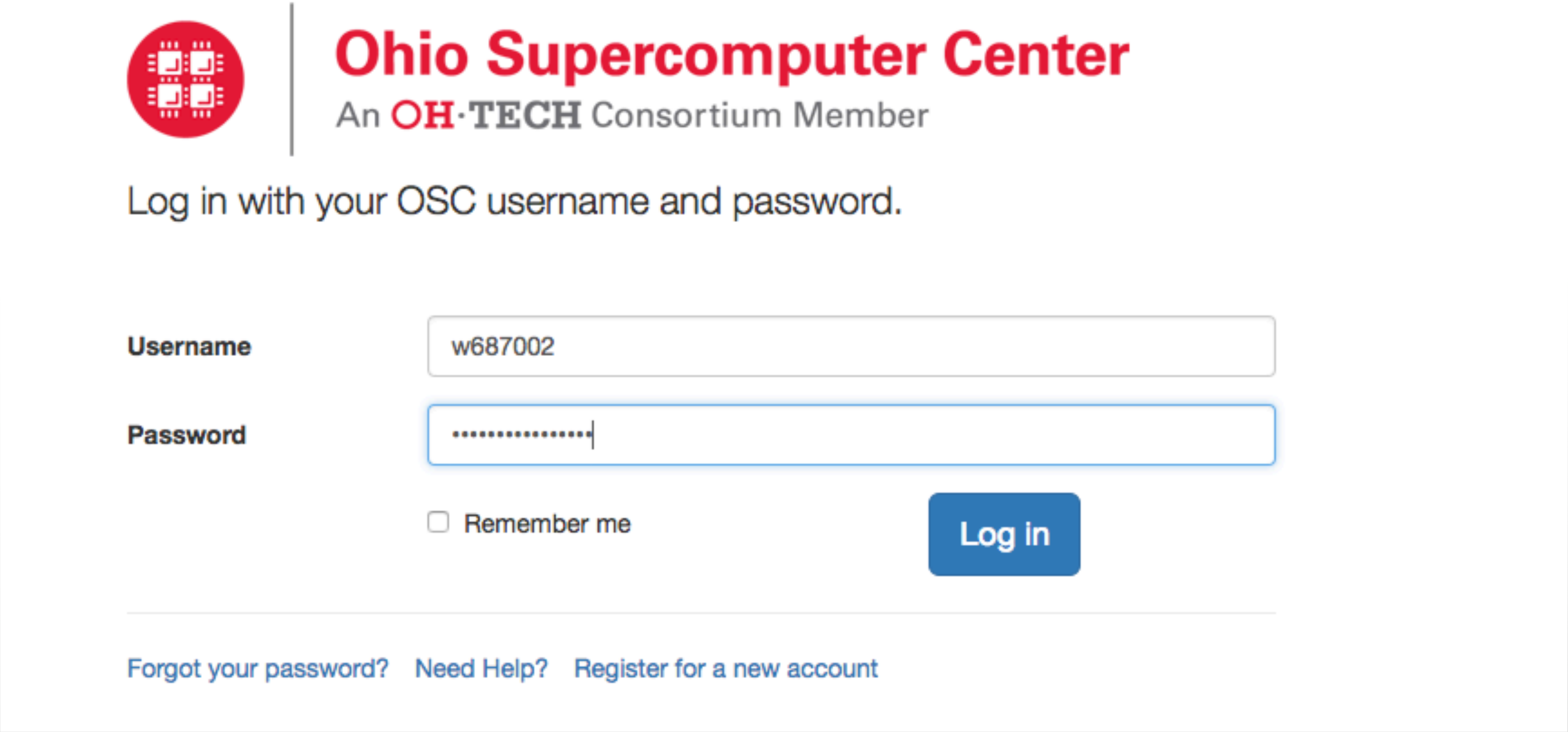
Log in with third party through CILogon

Step 1. Choose your identity provider

CILogon provides access to identity providers from many academic institutions across the state.

Step 2. Login via your provider

For example, here I've chosen Ohio State University as my provider and am presented OSU's login page.

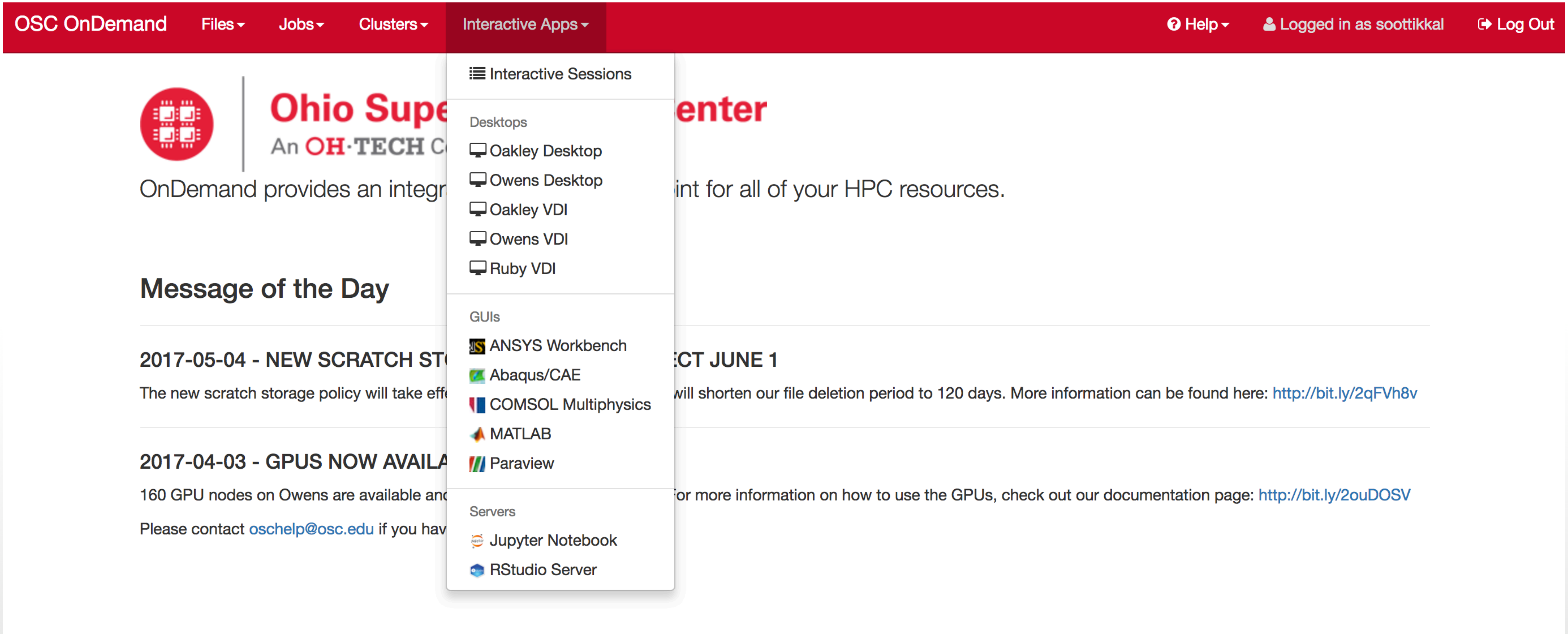


Ohio Supercomputer Center
An OH·TECH Consortium Member

Log in with your OSC username and password.

Username	w687002
Password	••••••••••••••••

☐ Remember me



OnDemand provides an integr int for all of your HPC resources.

Message of the Day

2017-05-04 - NEW SCRATCH ST CT JUNE 1

The new scratch storage policy will take eff will shorten our file deletion period to 120 days. More information can be found here: http://bit.ly/2qFVh8v

2017-04-03 - GPUS NOW AVAILA

160 GPU nodes on Owens are available and or more information on how to use the GPUs, check out our documentation page: http://bit.ly/2ouDOSV

Please contact oschelp@osc.edu if you hav

Exercise

Data Analytics@OSC

Python: A popular general-purpose, high-level programming language with numerous mathematical and scientific packages available for data analytics.

R: A programming language for statistical and machine learning applications with very strong graphical capabilities.

MATLAB:A full featured data analysis toolkit with many advanced algorithms readily available.

Spark and Hadoop: Frameworks for running map reduce algorithms

Intel Compilers: Compilers for generating optimized code for Intel CPUs.

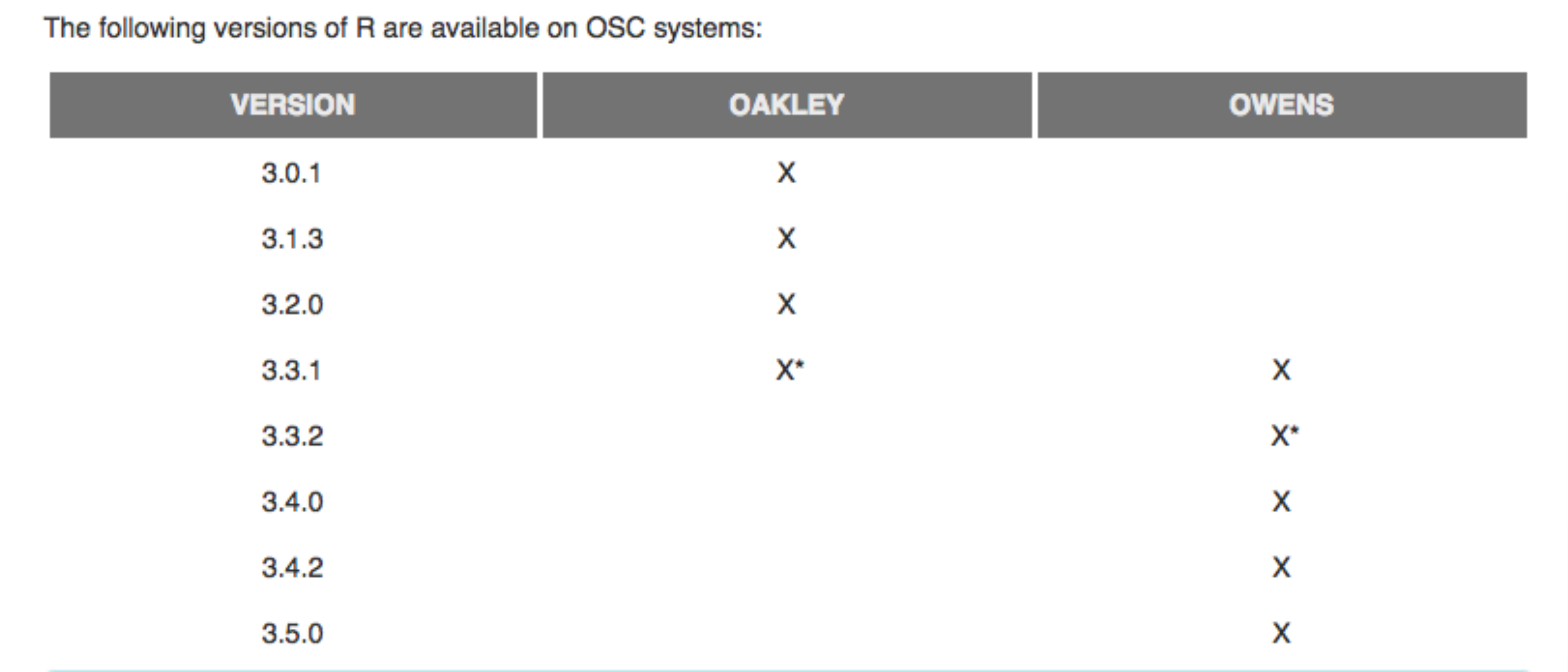
Intel MKL: The Math Kernel Library provides optimized subroutines for common computation tasks such as matrix-matrix calculations.

Statistical software: Octave, Stata, FFTW, ScaLAPACK, MINPACK, sprng2

R and Rstudio

R is a language and environment for statistical computing and graphics. R provides a wide variety of statistical and graphical techniques and is highly extensible.

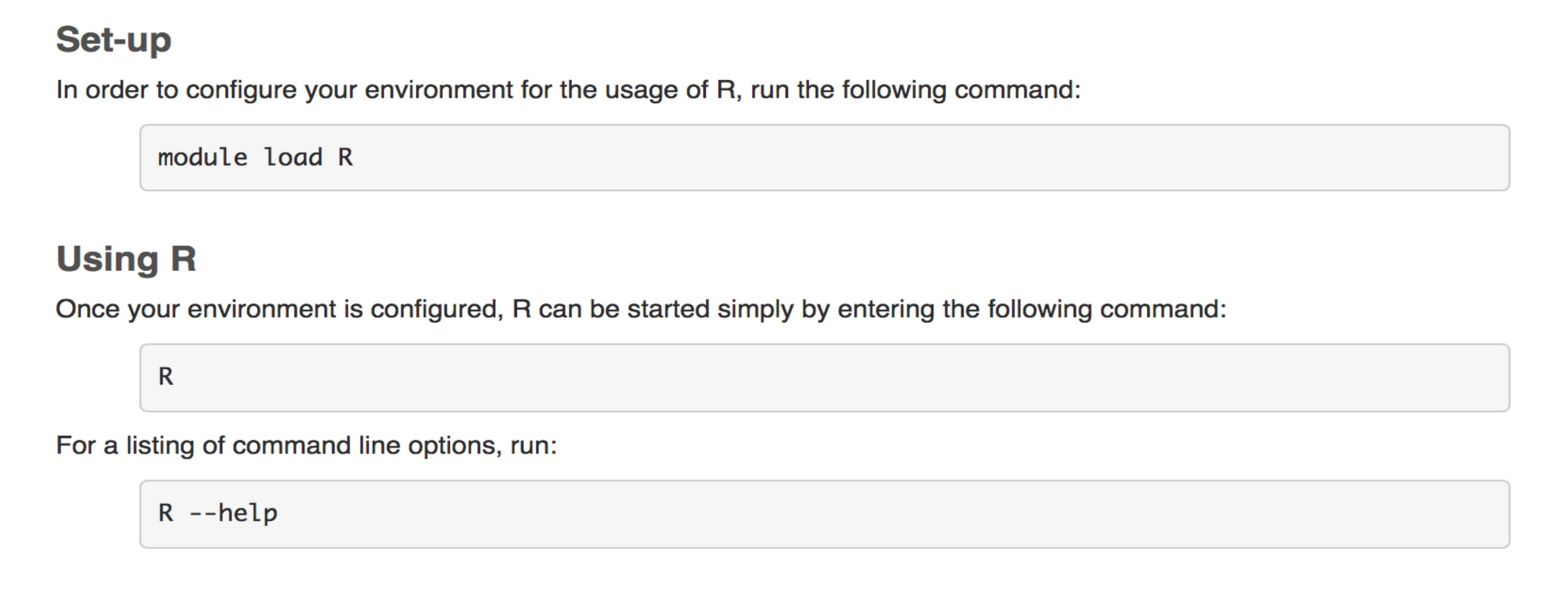
Availability:



The following versions of R are available on OSC systems:

VERSION	OAKLEY	OWENS
3.0.1	X	
3.1.3	X	
3.2.0	X	
3.3.1	X*	X
3.3.2		X*
3.4.0		X
3.4.2		X
3.5.0		X

Running R interactively



Set-up

In order to configure your environment for the usage of R, run the following command:

```
module load R
```

Using R

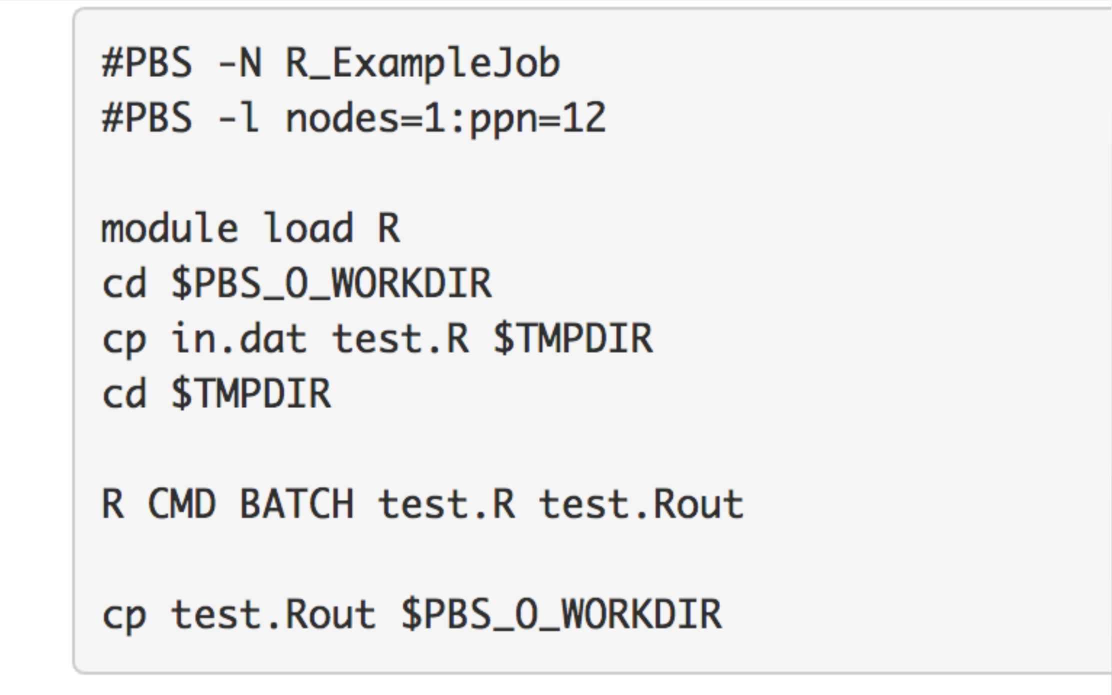
Once your environment is configured, R can be started simply by entering the following command:

```
R
```

For a listing of command line options, run:

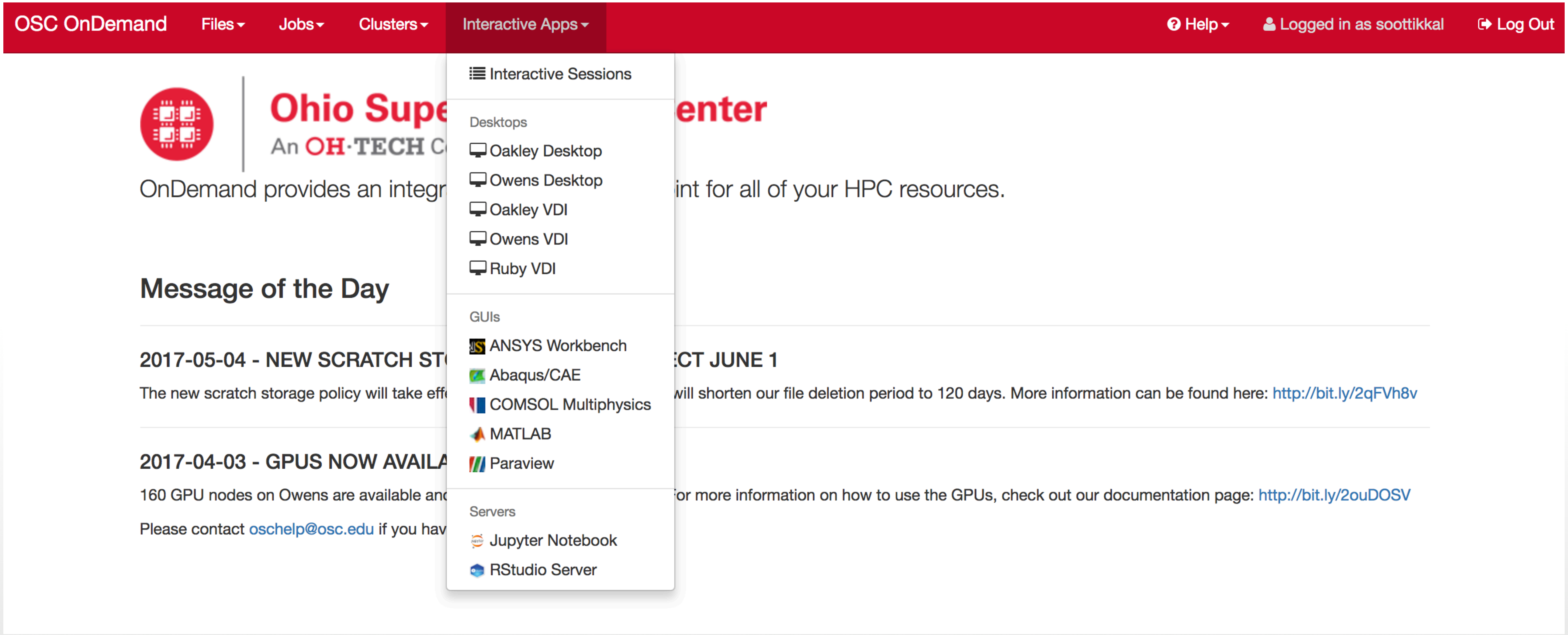
```
R --help
```

Batch Usage



```
#PBS -N R_ExampleJob
#PBS -l nodes=1:ppn=12

module load R
cd $PBS_O_WORKDIR
cp in.dat test.R $TMPDIR
cd $TMPDIR

R CMD BATCH test.R test.Rout

cp test.Rout $PBS_O_WORKDIR
```

Rstudio on Ondemand



OnDemand provides an integr int for all of your HPC resources.

Message of the Day

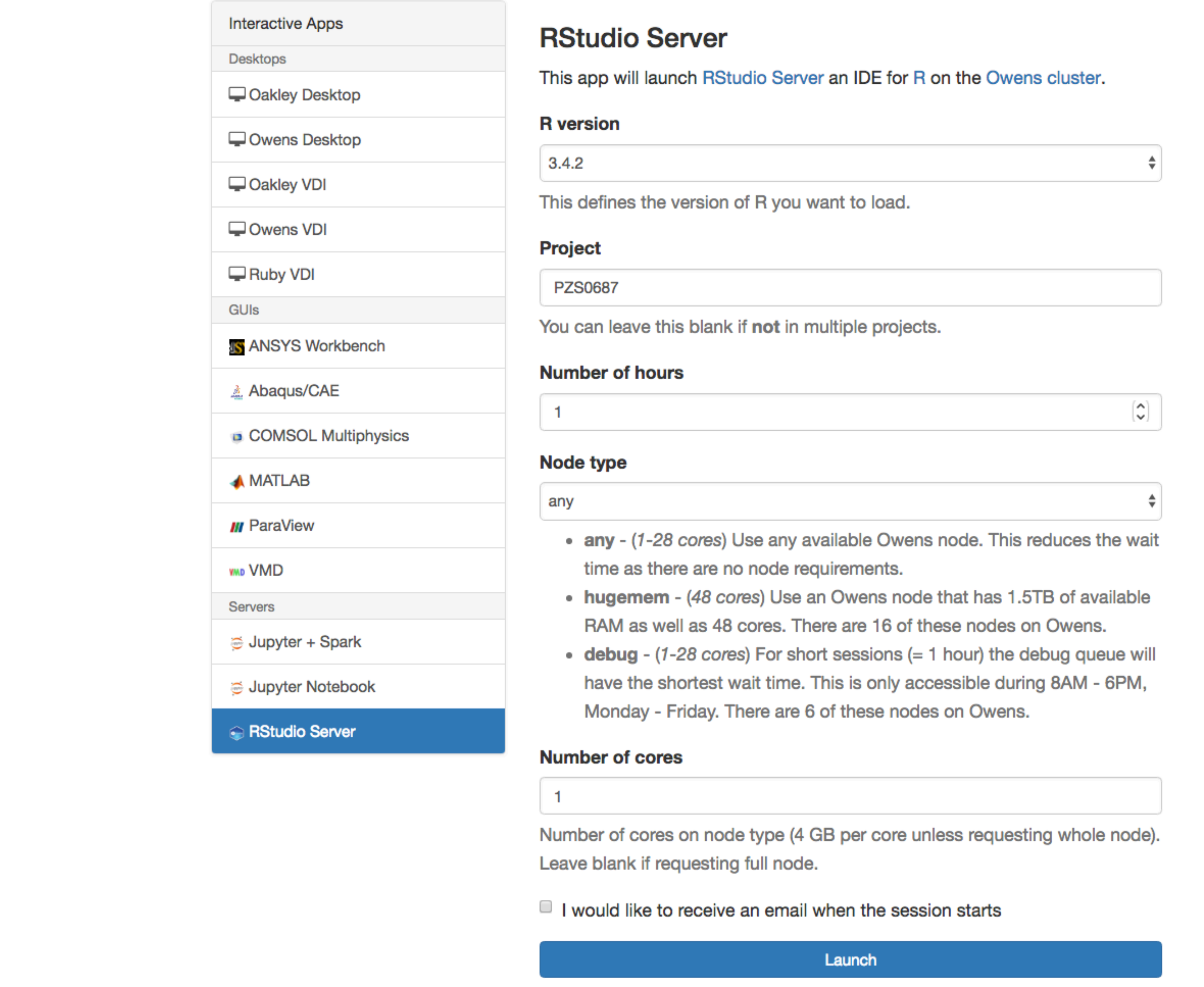
2017-05-04 - NEW SCRATCH ST ECT JUNE 1

The new scratch storage policy will take eff will shorten our file deletion period to 120 days. More information can be found here: http://bit.ly/2qFVh8v

2017-04-03 - GPUS NOW AVAILA

160 GPU nodes on Owens are available and or more information on how to use the GPUs, check out our documentation page: http://bit.ly/2ouDOSV

Please contact oschelp@osc.edu if you hav



Interactive Apps

Desktops

- Oakley Desktop
- Owens Desktop
- Oakley VDI
- Owens VDI
- Ruby VDI

GUIs

- ANSYS Workbench
- Abaqus/CAE
- COMSOL Multiphysics
- MATLAB
- ParaView
- VMD

Servers

- Jupyter + Spark
- Jupyter Notebook
- RStudio Server

RStudio Server

This app will launch RStudio Server an IDE for R on the Owens cluster.

R version

3.4.2

This defines the version of R you want to load.

Project

PZS0687

You can leave this blank if **not** in multiple projects.

Number of hours

1

Node type

any

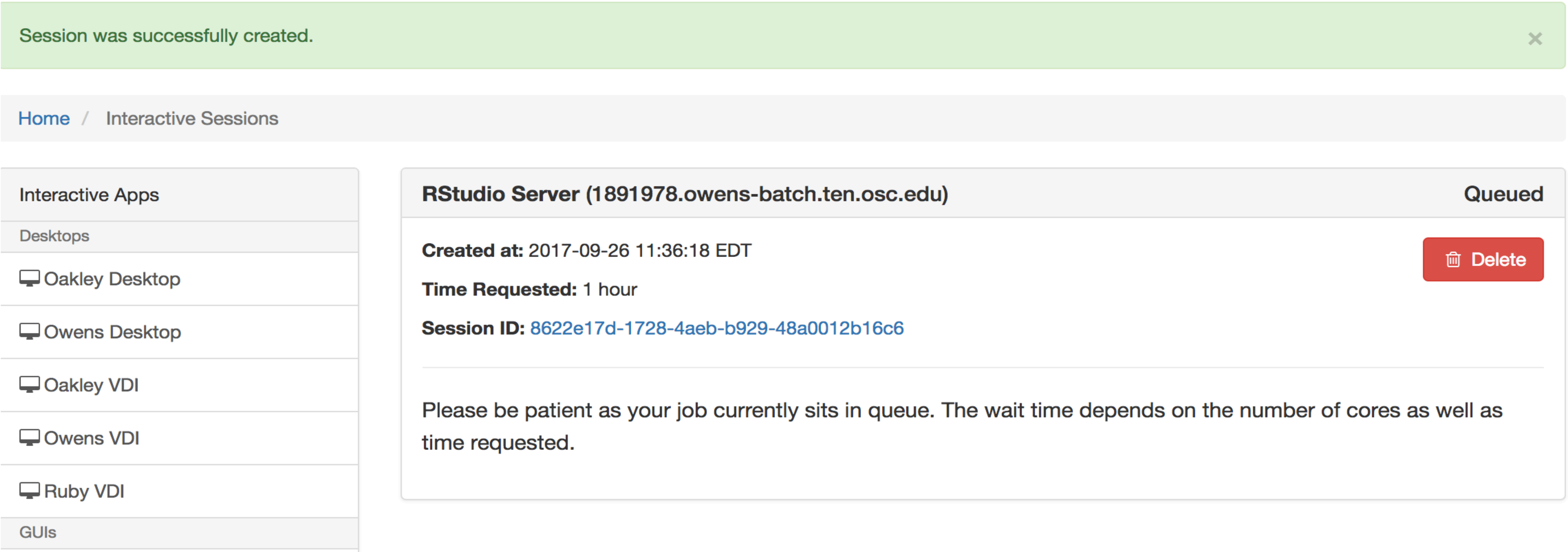
- **any** - (*1-28 cores*) Use any available Owens node. This reduces the wait time as there are no node requirements.
- **hugemem** - (*48 cores*) Use an Owens node that has 1.5TB of available RAM as well as 48 cores. There are 16 of these nodes on Owens.
- **debug** - (*1-28 cores*) For short sessions (= 1 hour) the debug queue will have the shortest wait time. This is only accessible during 8AM - 6PM, Monday - Friday. There are 6 of these nodes on Owens.

Number of cores

1

Number of cores on node type (4 GB per core unless requesting whole node). Leave blank if requesting full node.

☐ I would like to receive an email when the session starts



Session was successfully created.

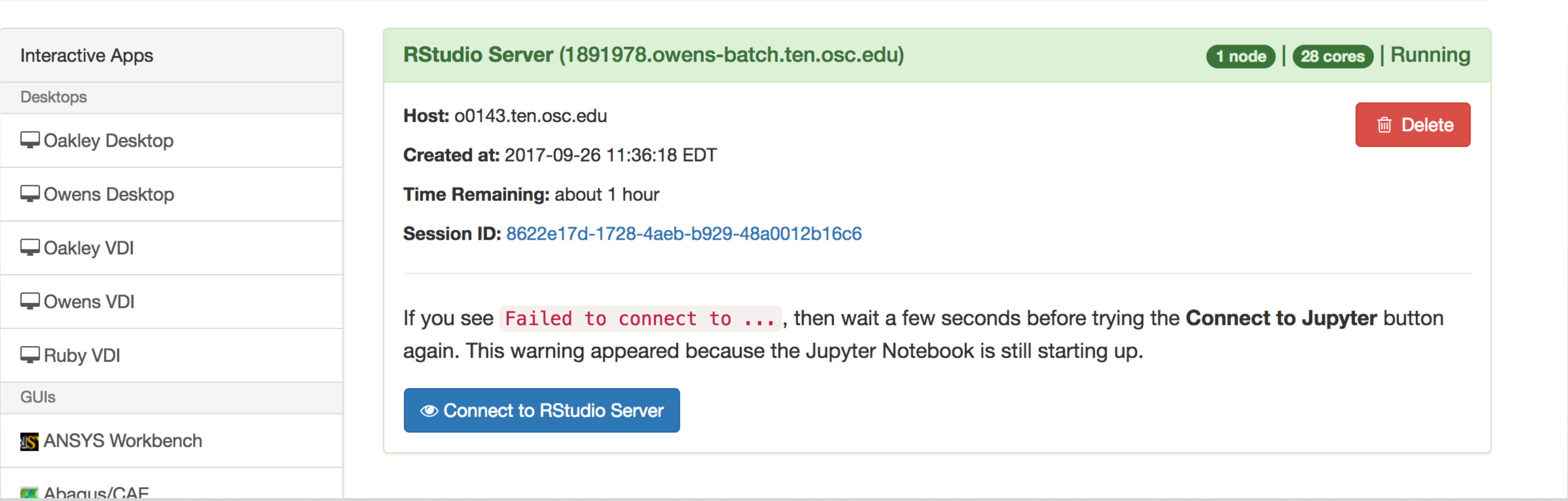
RStudio Server (1891978.owens-batch.ten.osc.edu) Queued

Created at: 2017-09-26 11:36:18 EDT

Time Requested: 1 hour

Session ID: 8622e17d-1728-4aeb-b929-48a0012b16c6

Please be patient as your job currently sits in queue. The wait time depends on the number of cores as well as time requested.



RStudio Server (1891978.owens-batch.ten.osc.edu) 1 node | 28 cores | Running

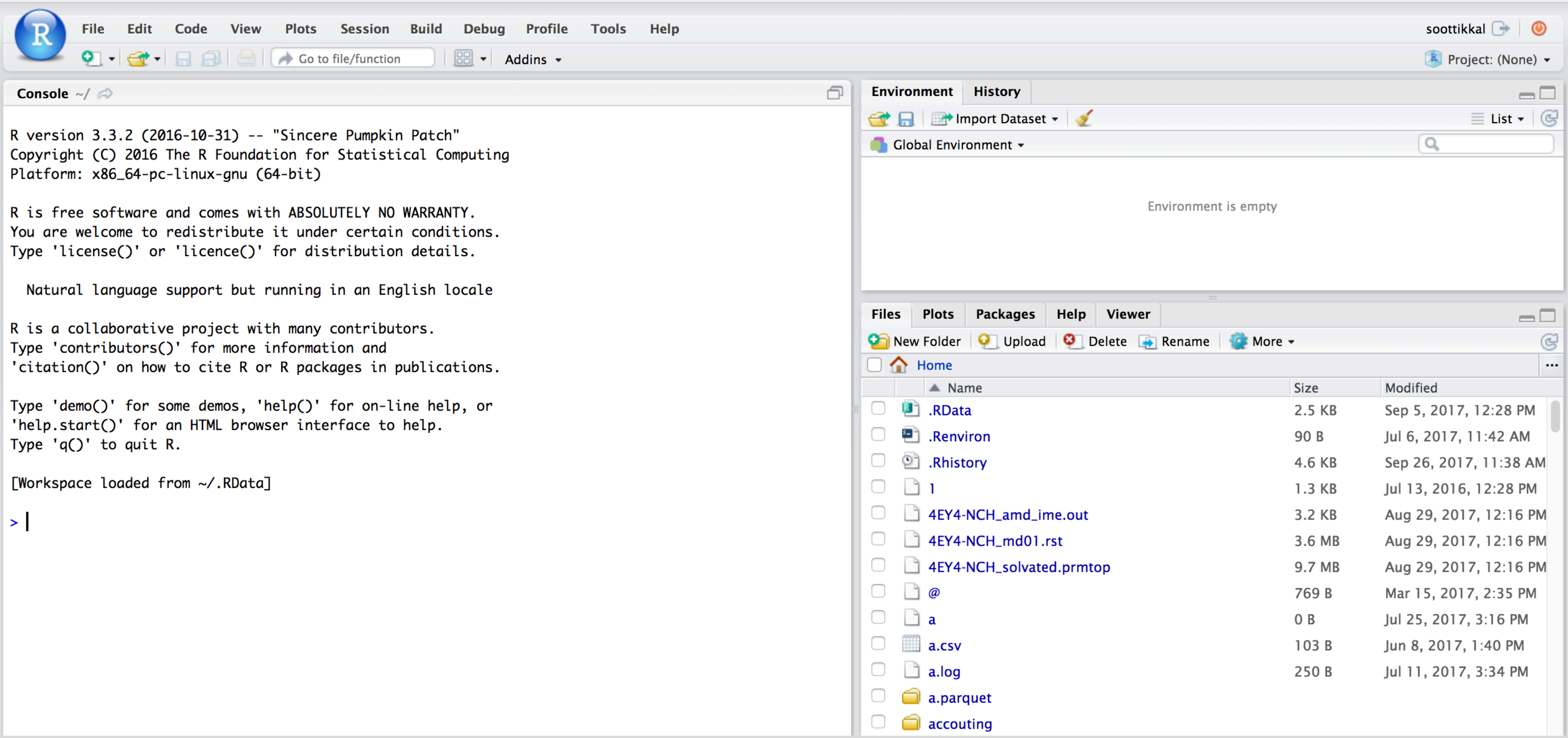
Host: o0143.ten.osc.edu

Created at: 2017-09-26 11:36:18 EDT

Time Remaining: about 1 hour

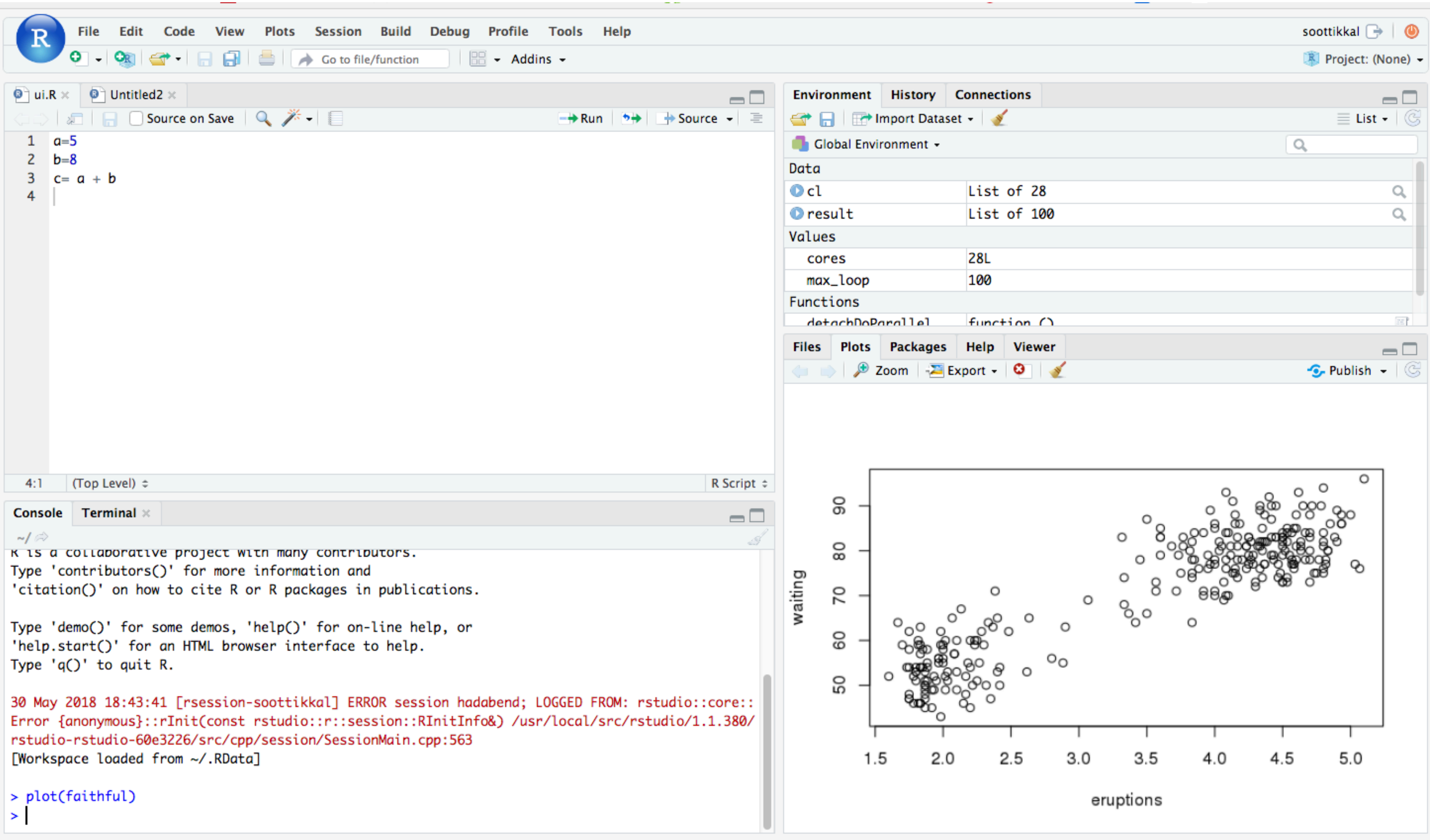
Session ID: 8622e17d-1728-4aeb-b929-48a0012b16c6

If you see `Failed to connect to ...`, then wait a few seconds before trying the **Connect to Jupyter** button again. This warning appeared because the Jupyter Notebook is still starting up.



```
R version 3.3.2 (2016-10-31) -- "Sincere Pumpkin Patch"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> |
```

Name	Size	Modified
.RData	2.5 KB	Sep 5, 2017, 12:28 PM
.Renviron	90 B	Jul 6, 2017, 11:42 AM
.Rhistory	4.6 KB	Sep 26, 2017, 11:38 AM
1	1.3 KB	Jul 13, 2016, 12:28 PM
4EY4-NCH_amd_ime.out	3.2 KB	Aug 29, 2017, 12:16 PM
4EY4-NCH_md01.rst	3.6 MB	Aug 29, 2017, 12:16 PM
4EY4-NCH_solvated.prmtop	9.7 MB	Aug 29, 2017, 12:16 PM
@	769 B	Mar 15, 2017, 2:35 PM
a	0 B	Jul 25, 2017, 3:16 PM
a.csv	103 B	Jun 8, 2017, 1:40 PM
a.log	250 B	Jul 11, 2017, 3:34 PM
a.parquet		
accouting		



```
a=5
b=8
c= a + b
```

Data

cl	List of 28
result	List of 100

Values

cores	28L
max_loop	100

Functions

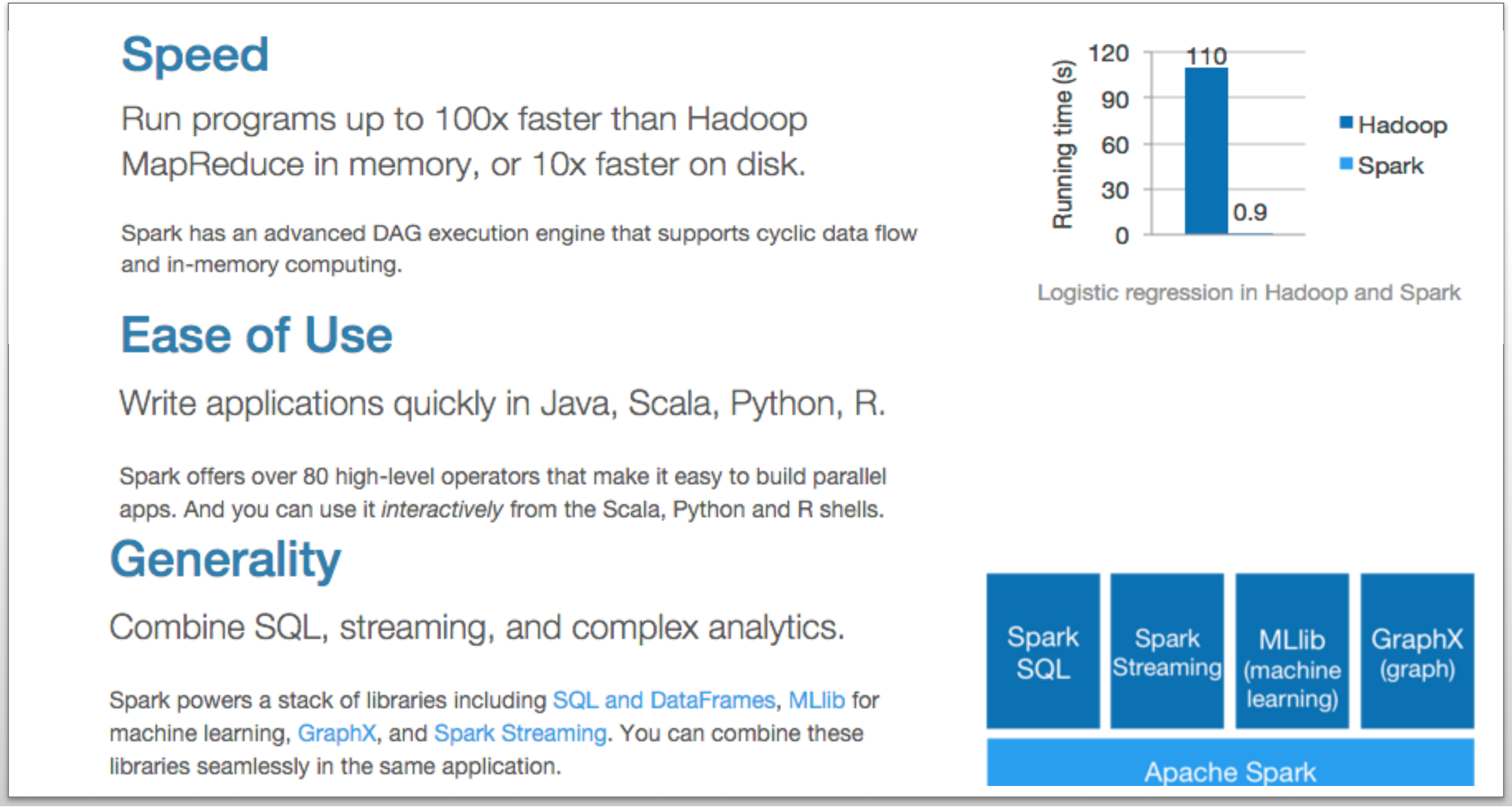
detachDoParallel	function ()

```
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

30 May 2018 18:43:41 [rsession-soottikkal] ERROR session hadabend; LOGGED FROM: rstudio::core::
Error {anonymous}::rInit(const rstudio::r::session::RInitInfo&) /usr/local/src/rstudio/1.1.380/
rstudio-rstudio-60e3226/src/cpp/session/SessionMain.cpp:563
[Workspace loaded from ~/.RData]

> plot(faithful)
> 
```

Exercise

Apache Spark

Apache Spark is an open source cluster computing framework originally developed in the AMPLab at University of California, Berkeley but was later donated to the Apache Software Foundation where it remains today. In contrast to Hadoop's disk-based analytics paradigm, Spark has multi-stage in-memory analytics.



Speed

Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.

Spark has an advanced DAG execution engine that supports cyclic data flow and in-memory computing.

Logistic regression in Hadoop and Spark

Ease of Use

Write applications quickly in Java, Scala, Python, R.

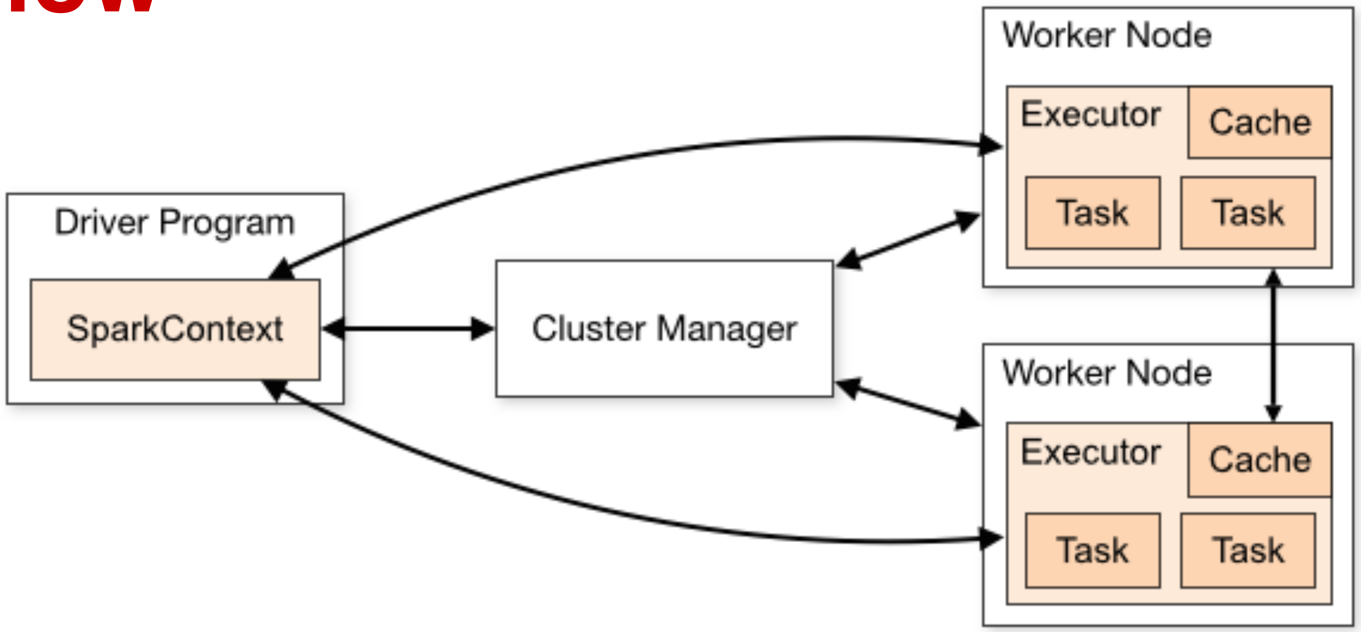
Spark offers over 80 high-level operators that make it easy to build parallel apps. And you can use it *interactively* from the Scala, Python and R shells.

Generality

Combine SQL, streaming, and complex analytics.

Spark powers a stack of libraries including SQL and DataFrames, MLlib for machine learning, GraphX, and Spark Streaming. You can combine these libraries seamlessly in the same application.

Spark workflow



Spark applications run as independent sets of processes on a cluster, coordinated by the SparkContext object in your main program (called the driver program).

Requires cluster managers which allocate resources across applications.

Once connected, Spark acquires executors on nodes in the cluster, which are processes that run computations and store data for your application.

Next, it sends your application code (defined by JAR or Python files passed to SparkContext) to the executors. Finally, SparkContext sends tasks to the executors to run.

RDD- Resilient Distributed Datasets

RDD (Resilient Distributed Dataset) is the main logical data unit in Spark. They are

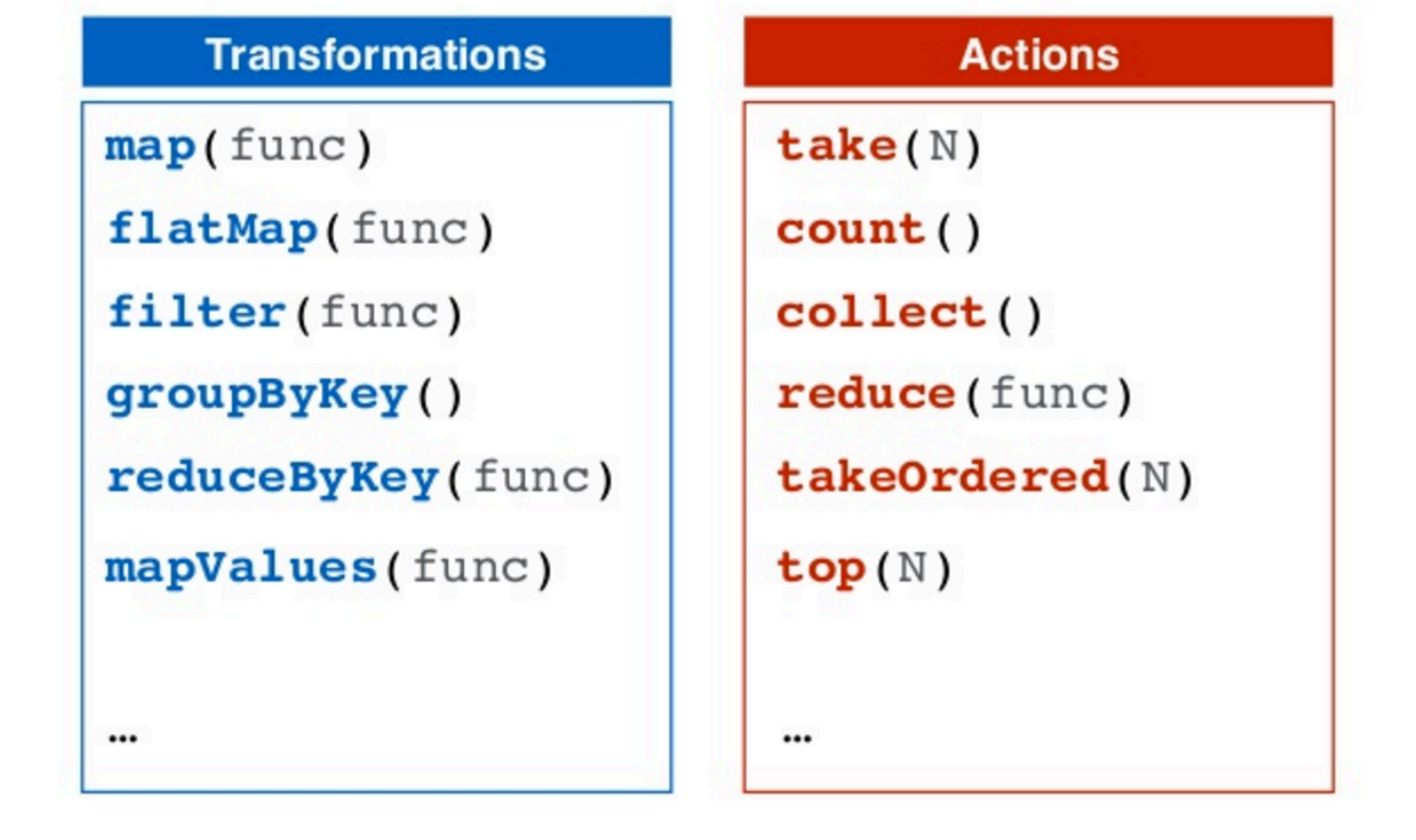
- Distributed and partitioned
- Stored in memory
- Immutable
- Partitions recomputed on failure

RDD- Transformations and Actions

Transformations are executed on demand. That means they are computed lazily. Eg: filter, join, sort

Actions return final results of RDD computations. Actions triggers execution using lineage graph to load the data into original RDD, carry out all intermediate transformations and return final results to Driver program or write it out to file system. Eg: collect(), count(), take()

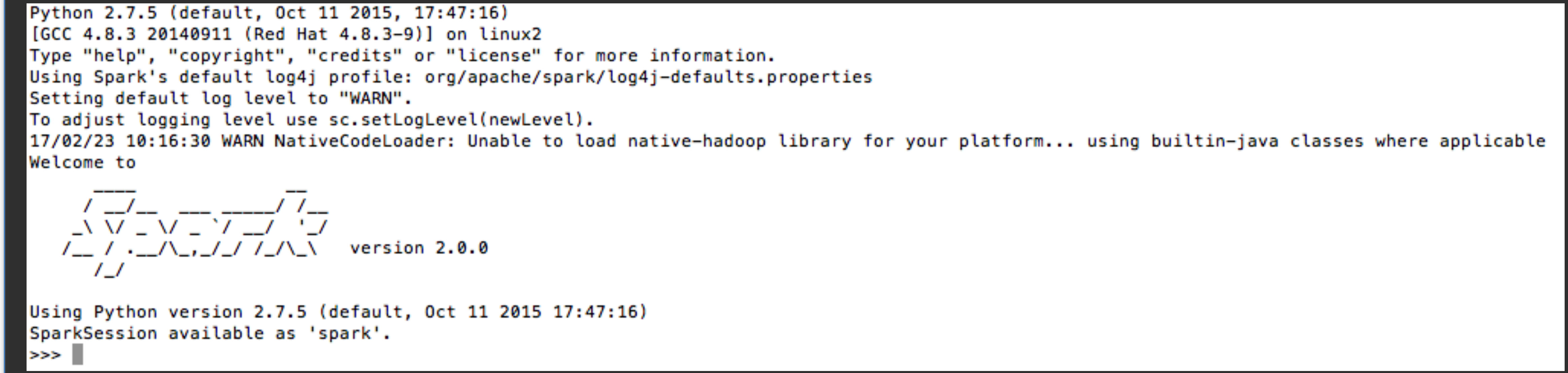
RDD Operations



Transformations	Actions
map(func)	take(N)
flatMap(func)	count()
filter(func)	collect()
groupByKey()	reduce(func)
reduceByKey(func)	takeOrdered(N)
mapValues(func)	top(N)
…	…

Interactive Analysis with the Spark Shell

$SPARK_HOME/bin/pyspark   # Opens SparkContext



```
Python 2.7.5 (default, Oct 11 2015, 17:47:16)
[GCC 4.8.3 20140911 (Red Hat 4.8.3-9)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
17/02/23 10:16:30 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.0.0
      /_/

Using Python version 2.7.5 (default, Oct 11 2015 17:47:16)
SparkSession available as 'spark'.
>>> 
```

1. Create a RDD

```
>>> data = sc.textFile("README.md")
```

2. Transformation of RDD

```
>>>linesWithSpark = data.filter(lambda line: "Spark" in line)
```

3. Action on RDD

```
>>> linesWithSpark.count()   # Number of items in this RDD
12
```

4. Combining Transformation and Actions

```
>>> data.filter(lambda line: "Spark" in line).count() # How many lines contain "Spark"?
12
```

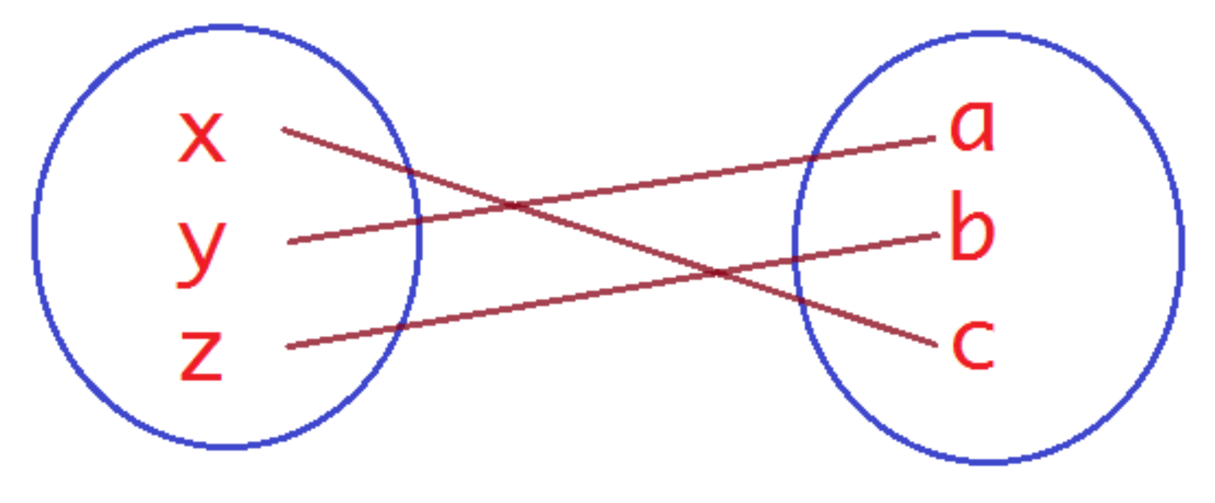
Ohio Supercomputer Center

OH·TECH | Ohio Technology Consortium
A Division of the Ohio Board of Regents

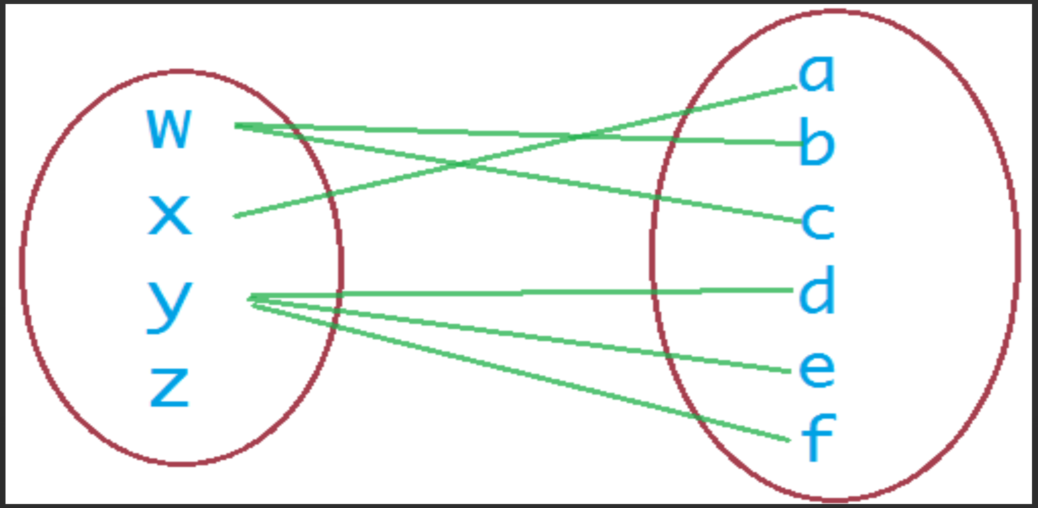
Word count Example

Map: One element in input gets mapped to only one element in output.
Flatmap: One element in input maps to zero or more elements in the output.

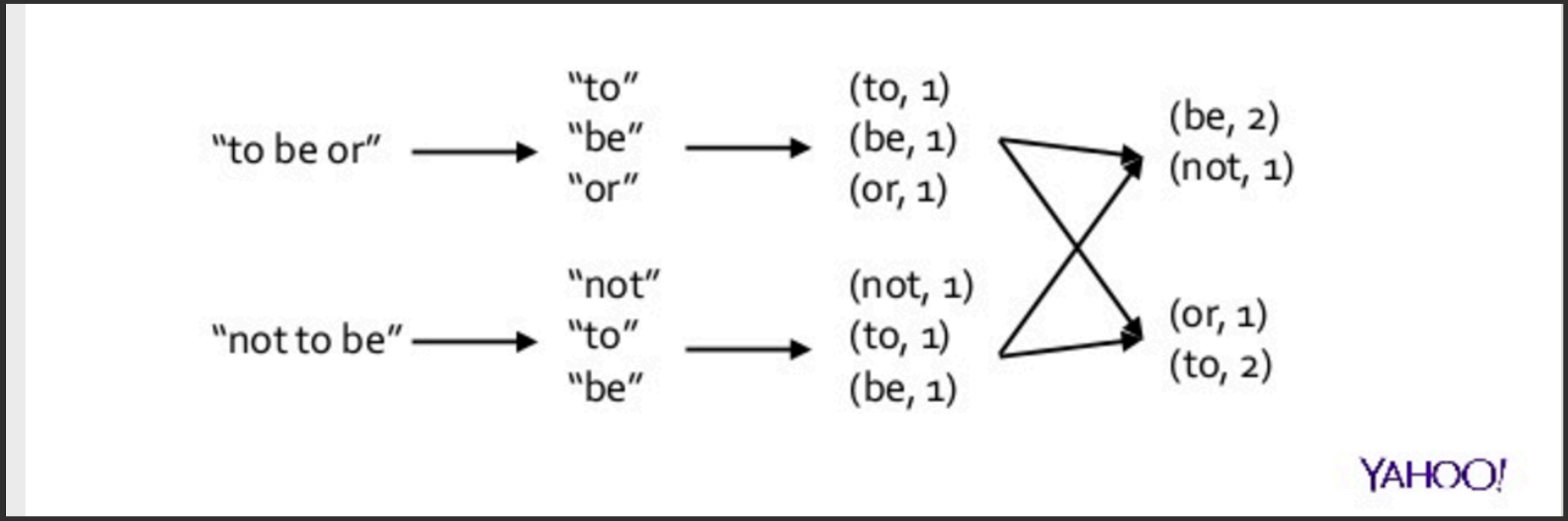
Map



Flatmap



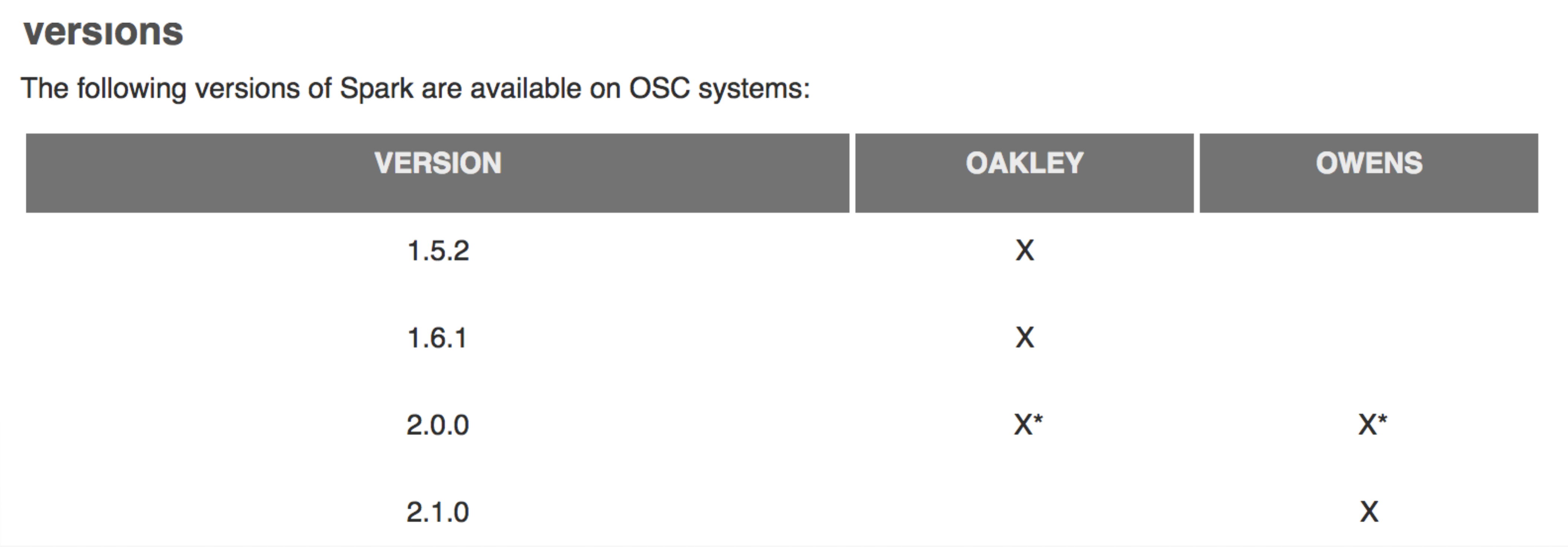
Word count Example



```
>>>wordCounts = data.flatMap(lambda line: line.split()).map(lambda word:
(word,1)).reduceByKey(lambda a, b: a+b)

>>> wordCounts.collect()
```

Spark documentation at OSC

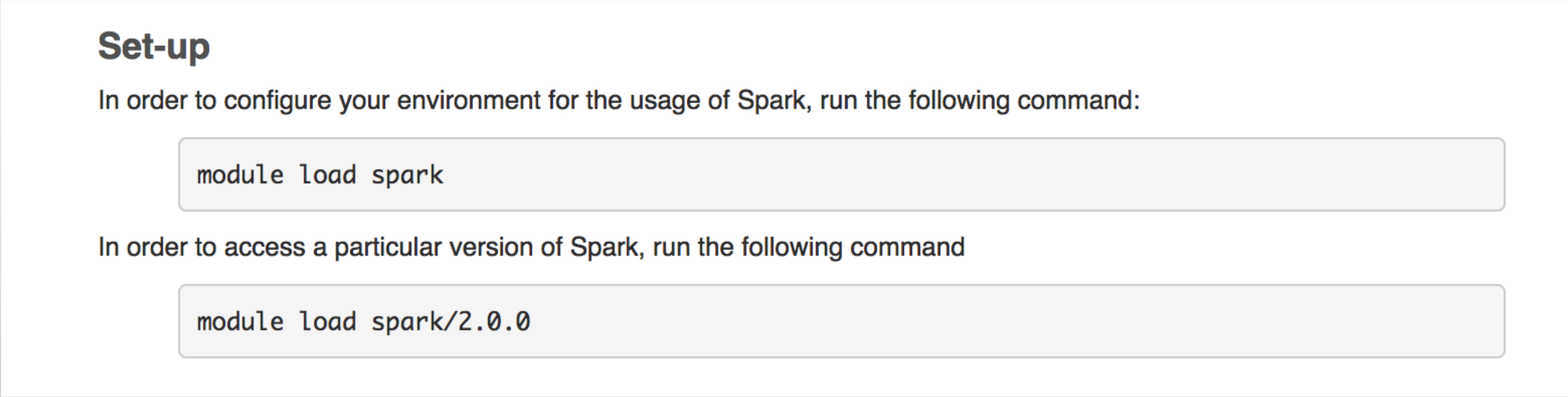
https://www.osc.edu/resources/available_software/software_list/spark



versions

The following versions of Spark are available on OSC systems:

VERSION	OAKLEY	OWENS
1.5.2	X	
1.6.1	X	
2.0.0	X*	X*
2.1.0		X



Set-up

In order to configure your environment for the usage of Spark, run the following command:

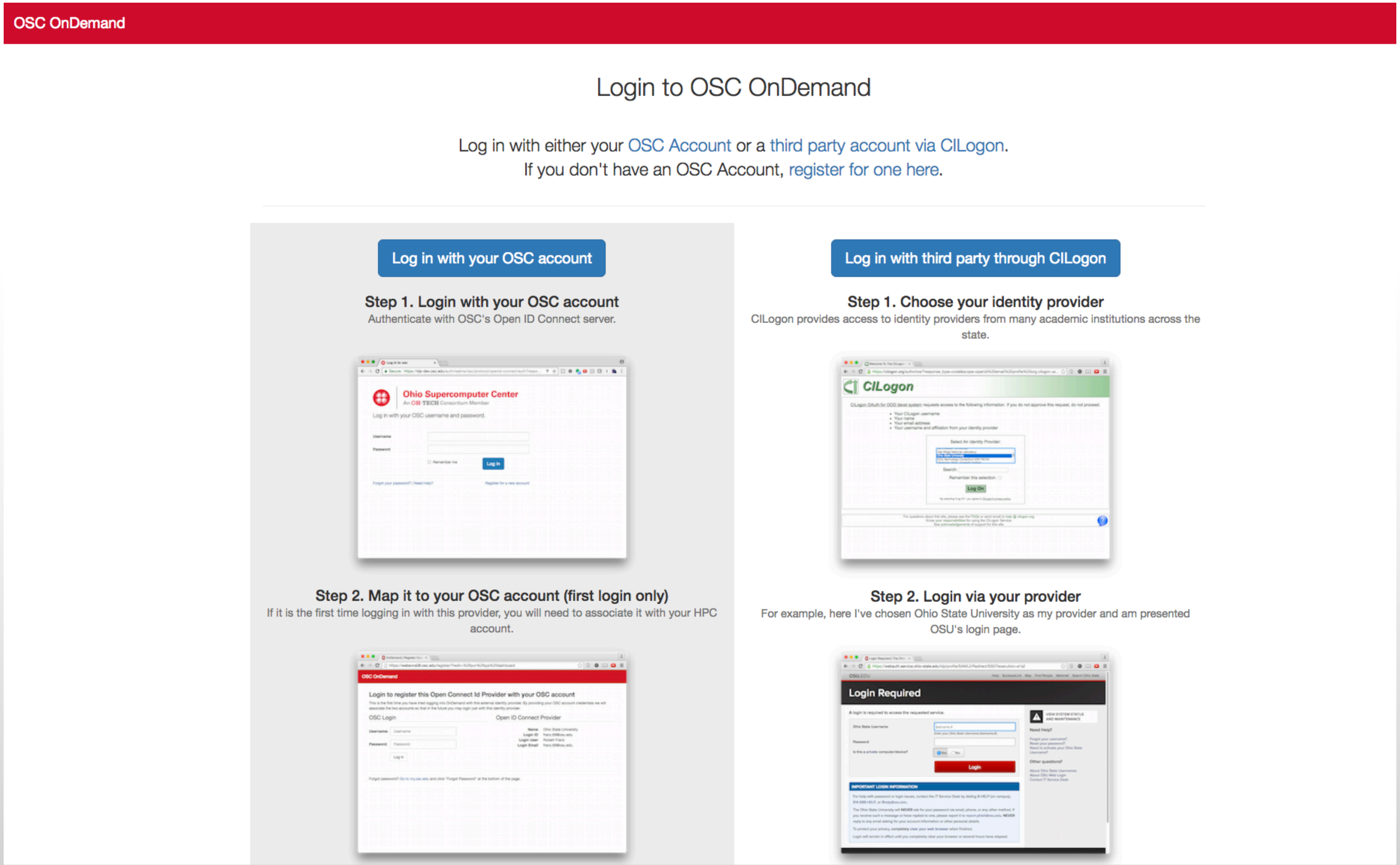
```
module load spark
```

In order to access a particular version of Spark, run the following command

```
module load spark/2.0.0
```

Running Spark interactively: Jupyter+Spark App

Go to https://ondemand.osc.edu/



OSC OnDemand

Login to OSC OnDemand

Log in with either your OSC Account or a third party account via CILogon.
If you don't have an OSC Account, register for one here.

Log in with your OSC account

Step 1. Login with your OSC account
Authenticate with OSC's Open ID Connect server.

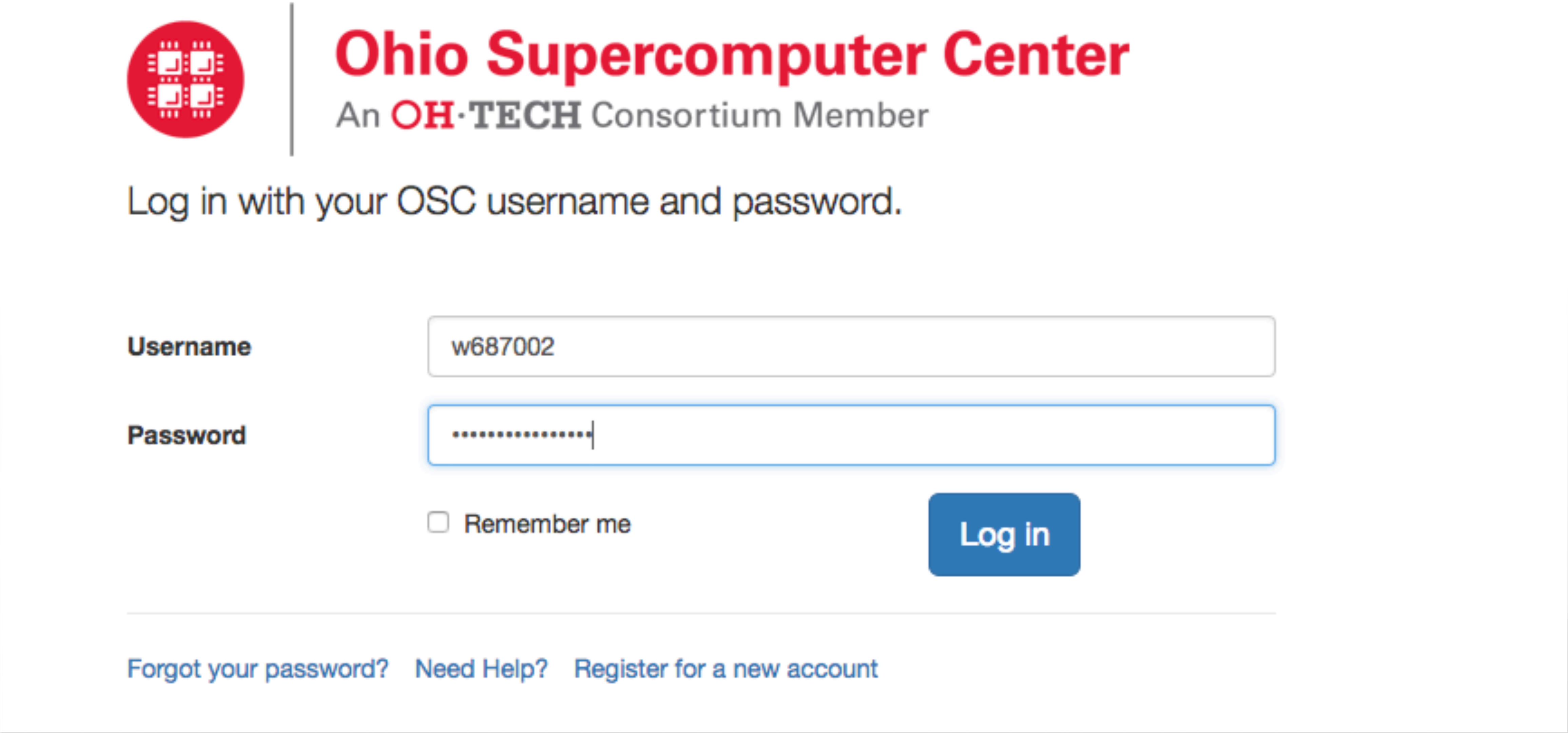
Step 2. Map it to your OSC account (first login only)
If it is the first time logging in with this provider, you will need to associate it with your HPC account.

Log in with third party through CILogon

Step 1. Choose your identity provider
CILogon provides access to identity providers from many academic institutions across the state.

Step 2. Login via your provider
For example, here I've chosen Ohio State University as my provider and am presented OSU's login page.

Log on with your OSC credentials.



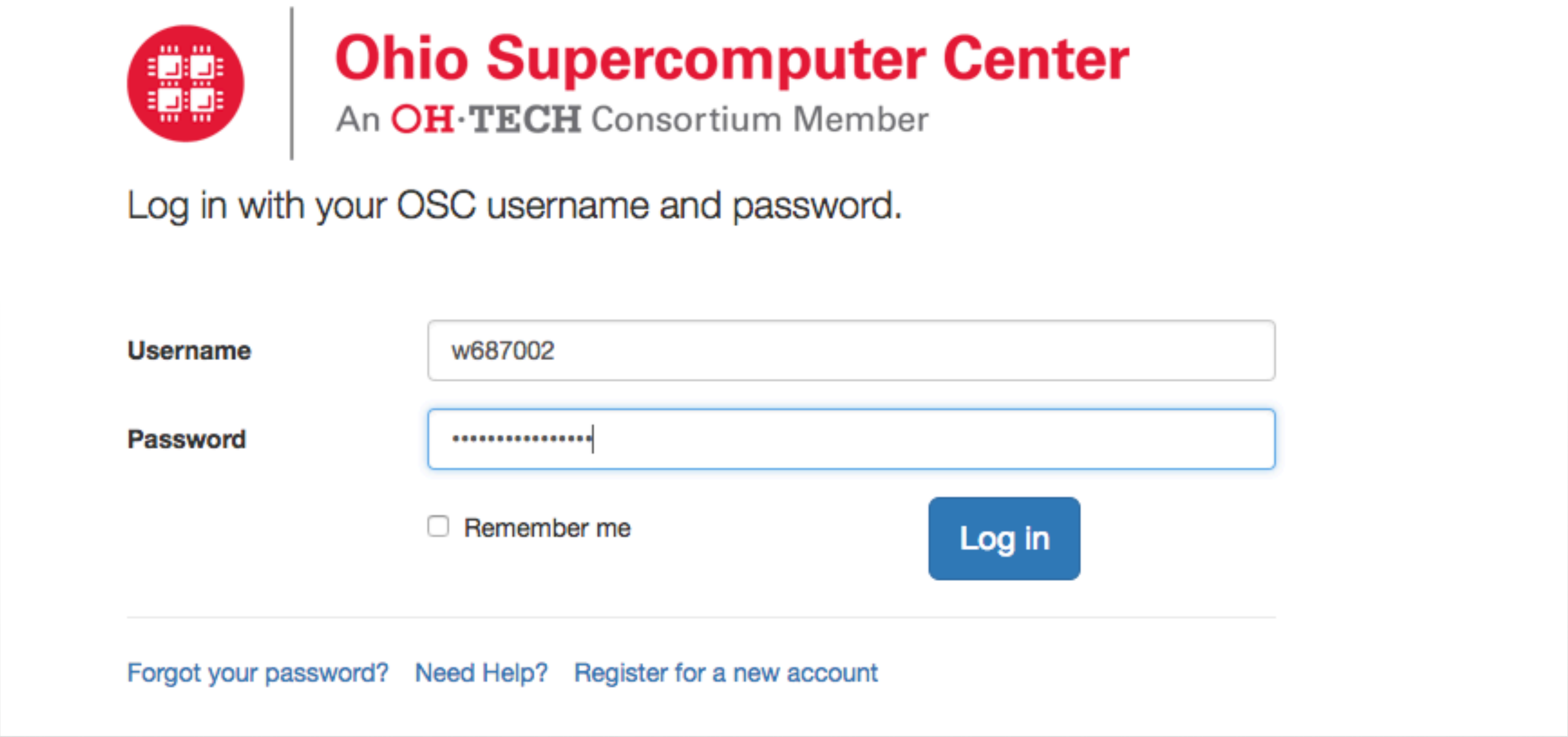
Ohio Supercomputer Center

An OH·TECH Consortium Member

Log in with your OSC username and password.

Username: w687002

Password: ••••••••••••••••



Ohio Supercomputer Center
An OH·TECH Consortium Member

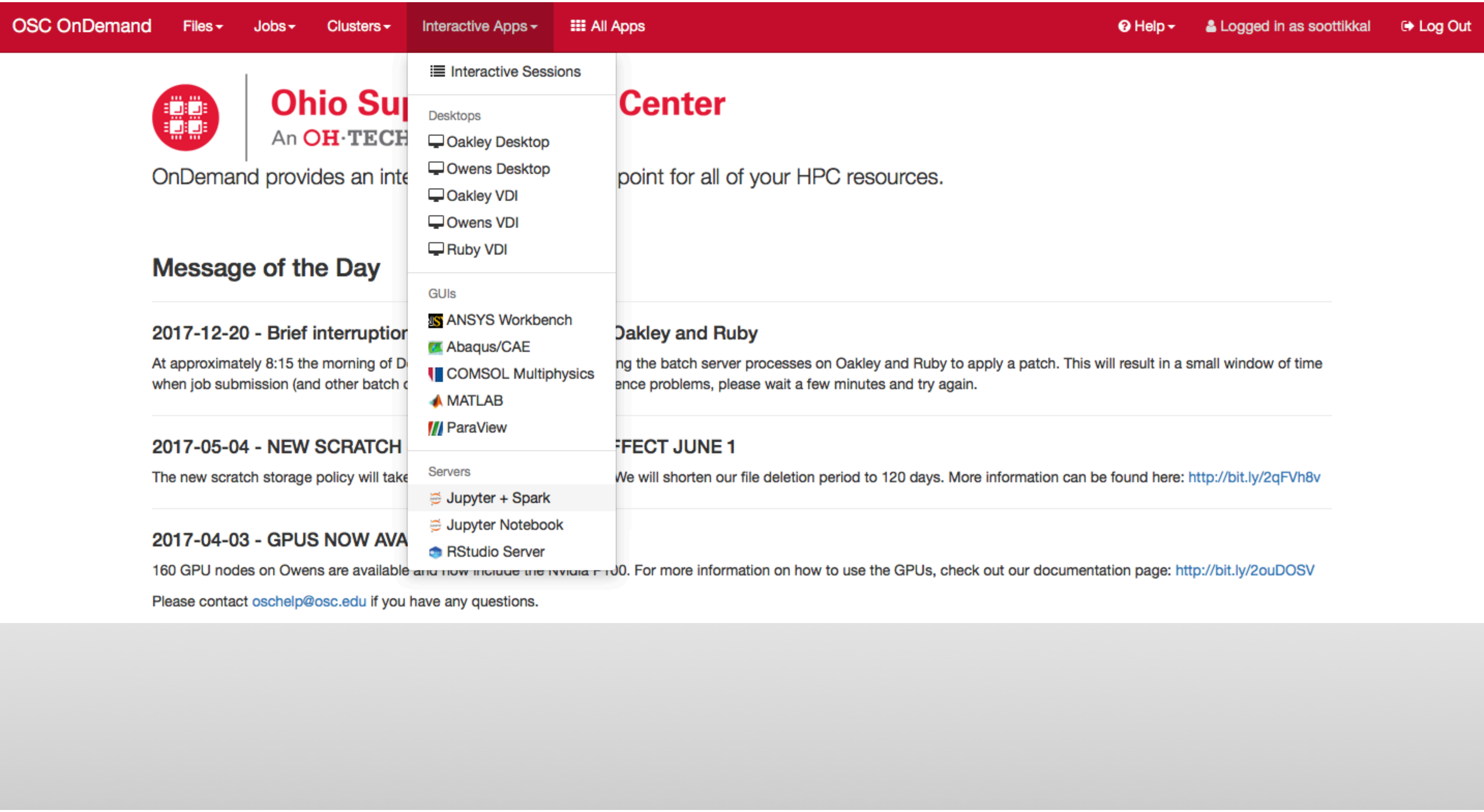
Log in with your OSC username and password.

Username: w687002

Password: ••••••••••••••••

□ Remember me

Choose Jupyter+Spark app from the Interactive Apps option.



OnDemand provides an inte point for all of your HPC resources.

Message of the Day

2017-12-20 - Brief interruption Oakley and Ruby

At approximately 8:15 the morning of D ng the batch server processes on Oakley and Ruby to apply a patch. This will result in a small window of time when job submission (and other batch c ence problems, please wait a few minutes and try again.

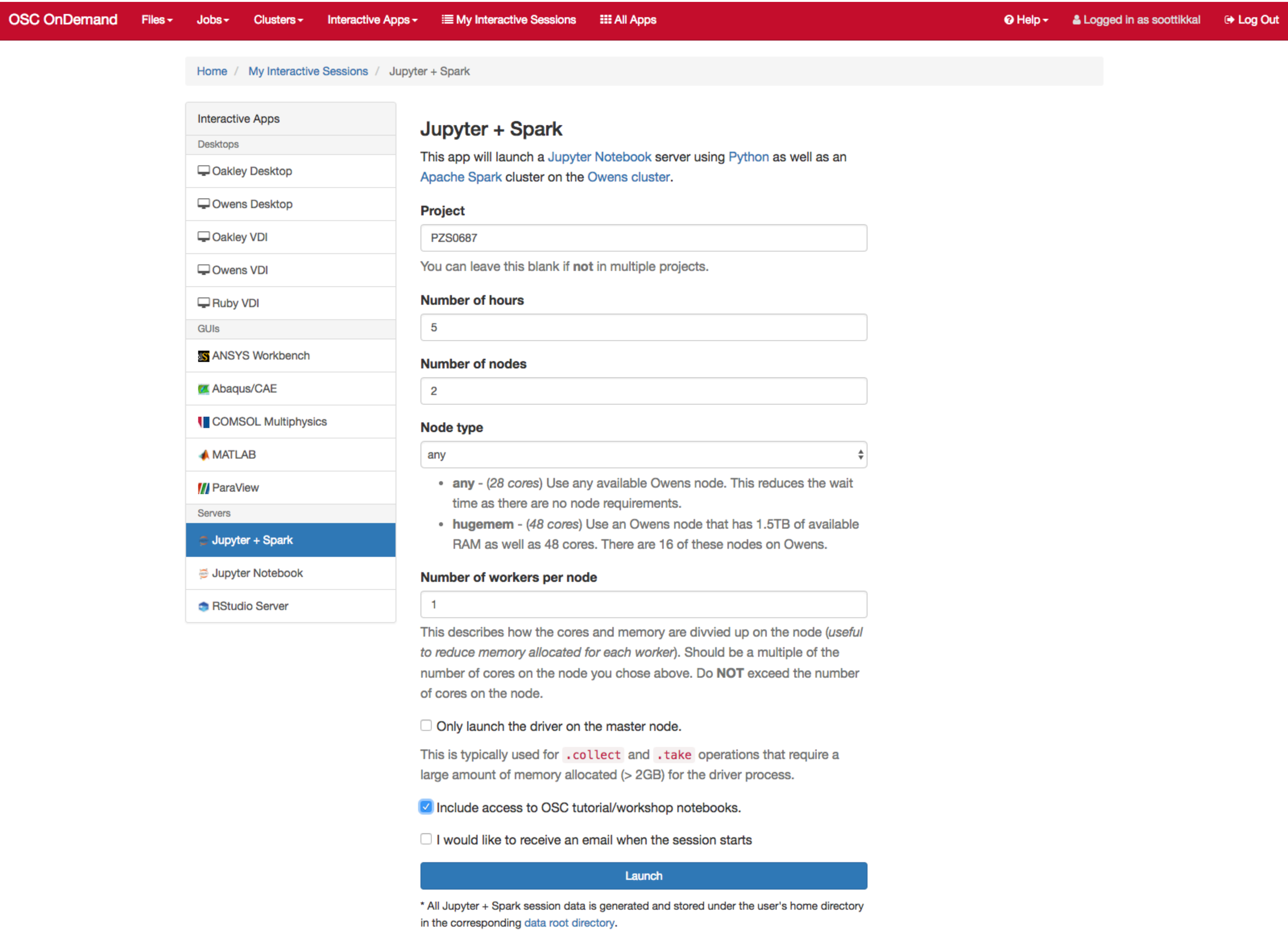
2017-05-04 - NEW SCRATCH FFECT JUNE 1

The new scratch storage policy will take We will shorten our file deletion period to 120 days. More information can be found here: http://bit.ly/2qFVh8v

2017-04-03 - GPUS NOW AVA

160 GPU nodes on Owens are available 00. For more information on how to use the GPUs, check out our documentation page: http://bit.ly/2ouDOSV

Please contact oschelp@osc.edu if you have any questions.



Jupyter + Spark

This app will launch a Jupyter Notebook server using Python as well as an Apache Spark cluster on the Owens cluster.

Project

PZS0687

You can leave this blank if **not** in multiple projects.

Number of hours

5

Number of nodes

2

Node type

any

- **any** - (*28 cores*) Use any available Owens node. This reduces the wait time as there are no node requirements.
- **hugemem** - (*48 cores*) Use an Owens node that has 1.5TB of available RAM as well as 48 cores. There are 16 of these nodes on Owens.

Number of workers per node

1

This describes how the cores and memory are divvied up on the node (*useful to reduce memory allocated for each worker*). Should be a multiple of the number of cores on the node you chose above. Do **NOT** exceed the number of cores on the node.

☐ Only launch the driver on the master node.

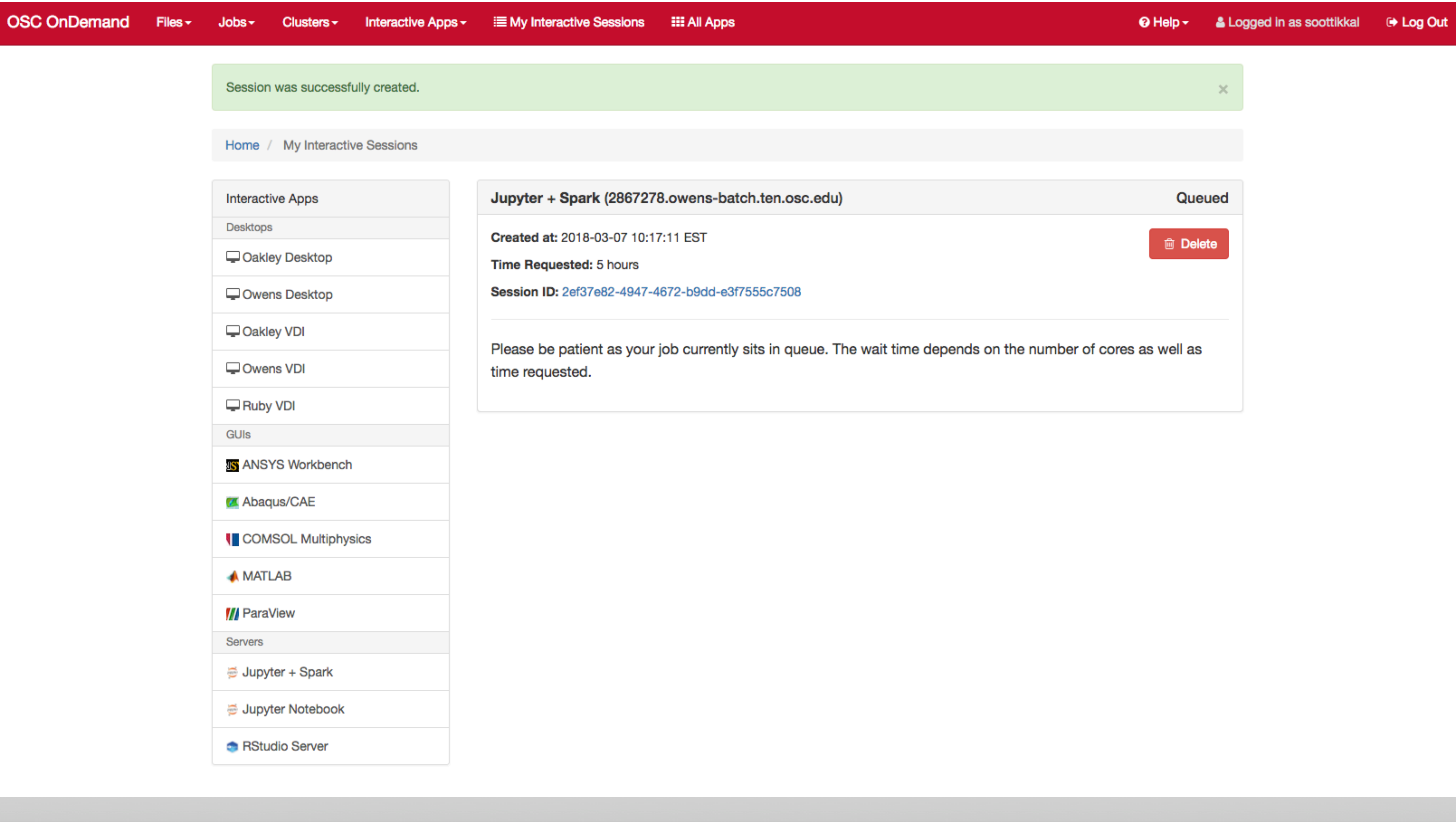
This is typically used for `.collect` and `.take` operations that require a large amount of memory allocated (> 2GB) for the driver process.

☑ Include access to OSC tutorial/workshop notebooks.

☐ I would like to receive an email when the session starts

Launch

* All Jupyter + Spark session data is generated and stored under the user's home directory in the corresponding data root directory.



Session was successfully created.

Home / My Interactive Sessions

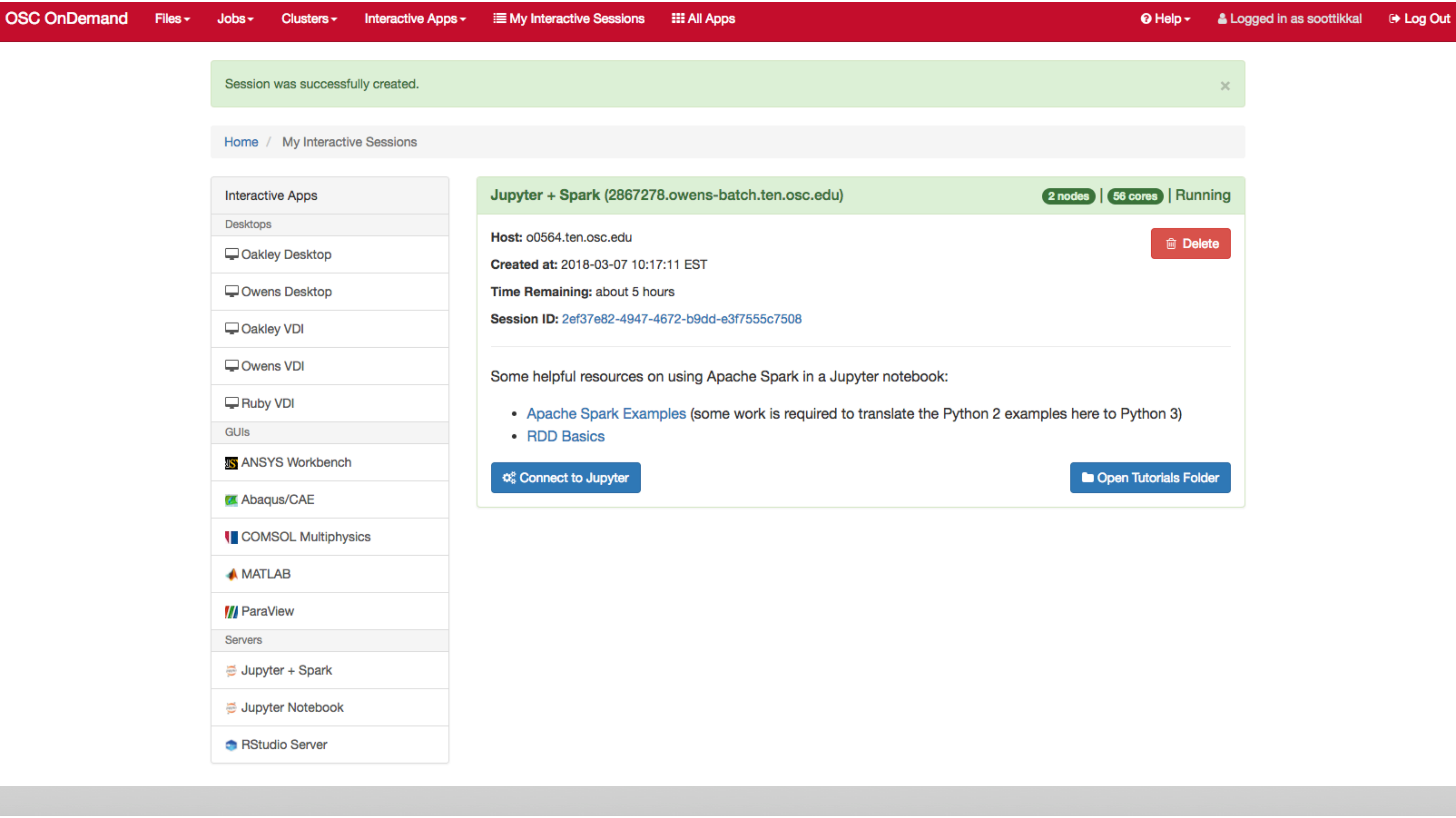
Jupyter + Spark (2867278.owens-batch.ten.osc.edu) Queued

Created at: 2018-03-07 10:17:11 EST

Time Requested: 5 hours

Session ID: 2ef37e82-4947-4672-b9dd-e3f7555c7508

Please be patient as your job currently sits in queue. The wait time depends on the number of cores as well as time requested.



Session was successfully created. ×

Home / My Interactive Sessions

Jupyter + Spark (2867278.owens-batch.ten.osc.edu) 2 nodes | 56 cores | Running

Host: o0564.ten.osc.edu

Created at: 2018-03-07 10:17:11 EST

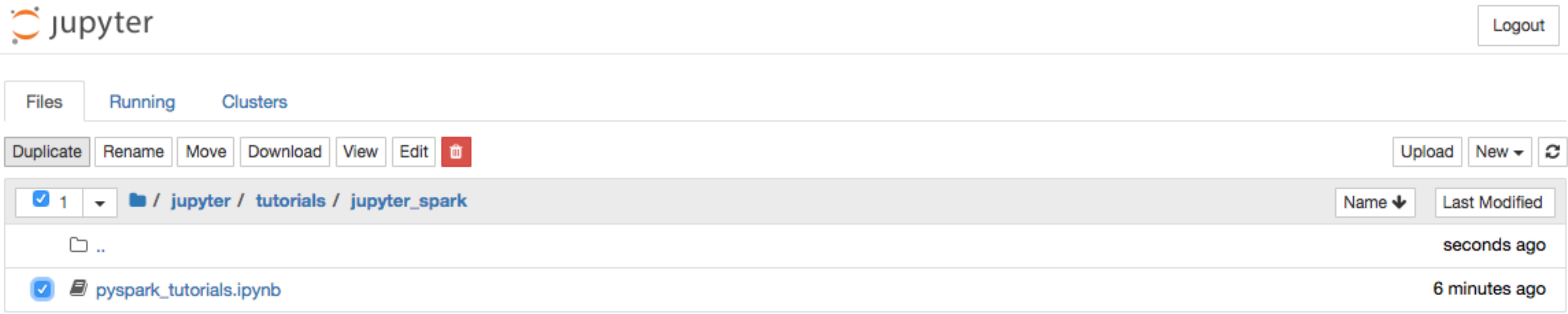
Time Remaining: about 5 hours

Session ID: 2ef37e82-4947-4672-b9dd-e3f7555c7508

Some helpful resources on using Apache Spark in a Jupyter notebook:

- Apache Spark Examples (some work is required to translate the Python 2 examples here to Python 3)
- RDD Basics

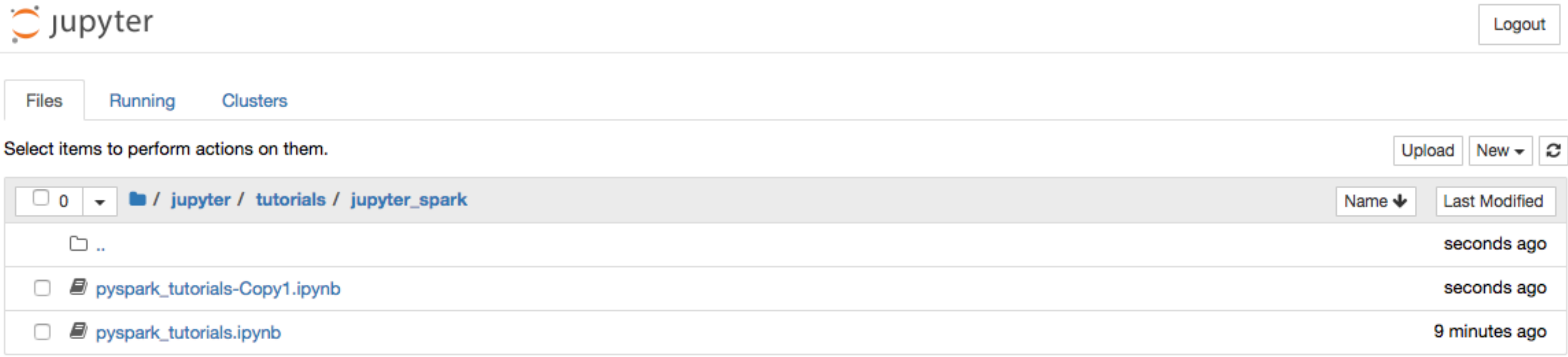
You will see a file called pyspark_tutorials.ipynb. Please check on the file and click on duplicate to make a copy of the file.



☑ 1 / jupyter / tutorials / jupyter_spark

	..	seconds ago
☑	pyspark_tutorials.ipynb	6 minutes ago

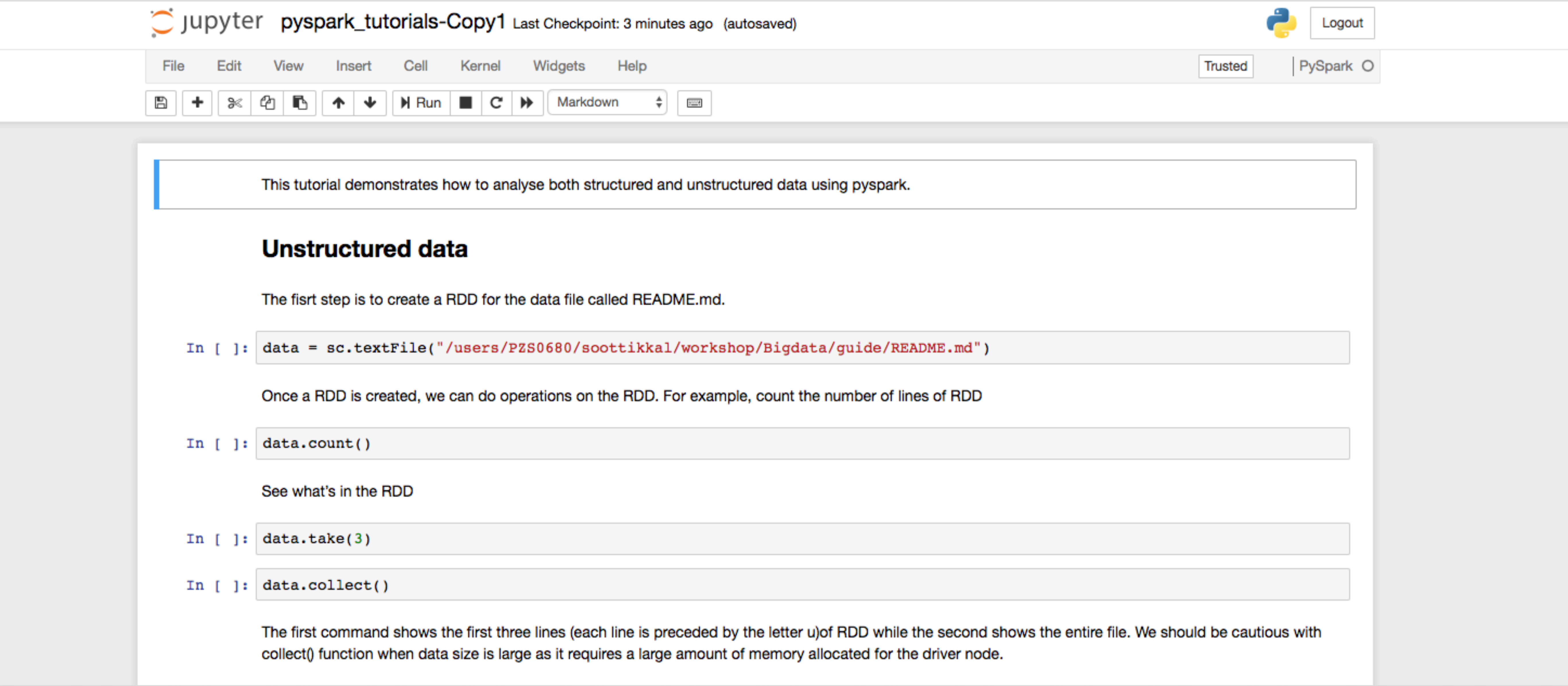
You will see a new file pyspark_tutorials-Copy1.ipynb is created. Double-click on the pyspark_tutorials-Copy1.ipynb file will launch Jupyter interface for Spark to proceed with the tutorials.



Select items to perform actions on them.

□ 0 / jupyter / tutorials / jupyter_spark

	..	seconds ago
□	pyspark_tutorials-Copy1.ipynb	seconds ago
□	pyspark_tutorials.ipynb	9 minutes ago



This tutorial demonstrates how to analyse both structured and unstructured data using pyspark.

Unstructured data

The fisrt step is to create a RDD for the data file called README.md.

```
In [ ]: data = sc.textFile("/users/PZS0680/soottikkal/workshop/Bigdata/guide/README.md")
```

Once a RDD is created, we can do operations on the RDD. For example, count the number of lines of RDD

```
In [ ]: data.count()
```

See what's in the RDD

```
In [ ]: data.take(3)
```

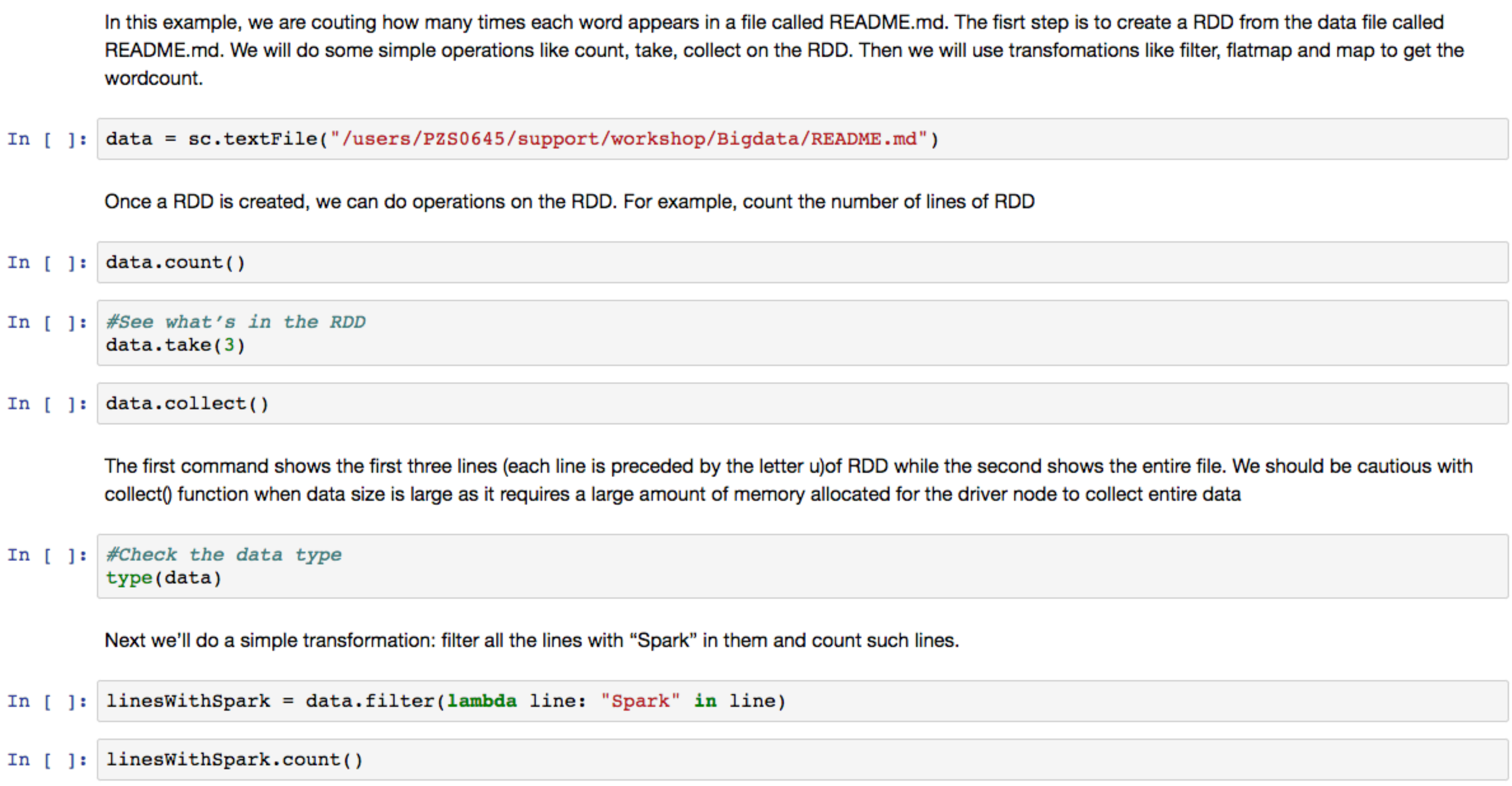
```
In [ ]: data.collect()
```

The first command shows the first three lines (each line is preceded by the letter u)of RDD while the second shows the entire file. We should be cautious with collect() function when data size is large as it requires a large amount of memory allocated for the driver node.

Exercise

https://www.osc.edu/content/launching_jupyter_spark_app

Spark RDD



In this example, we are couting how many times each word appears in a file called README.md. The fisrt step is to create a RDD from the data file called README.md. We will do some simple operations like count, take, collect on the RDD. Then we will use transfomations like filter, flatmap and map to get the wordcount.

```
In [ ]: data = sc.textFile("/users/PZS0645/support/workshop/Bigdata/README.md")
```

Once a RDD is created, we can do operations on the RDD. For example, count the number of lines of RDD

```
In [ ]: data.count()
```

```
In [ ]: #See what’s in the RDD
        data.take(3)
```

```
In [ ]: data.collect()
```

The first command shows the first three lines (each line is preceded by the letter u)of RDD while the second shows the entire file. We should be cautious with collect() function when data size is large as it requires a large amount of memory allocated for the driver node to collect entire data

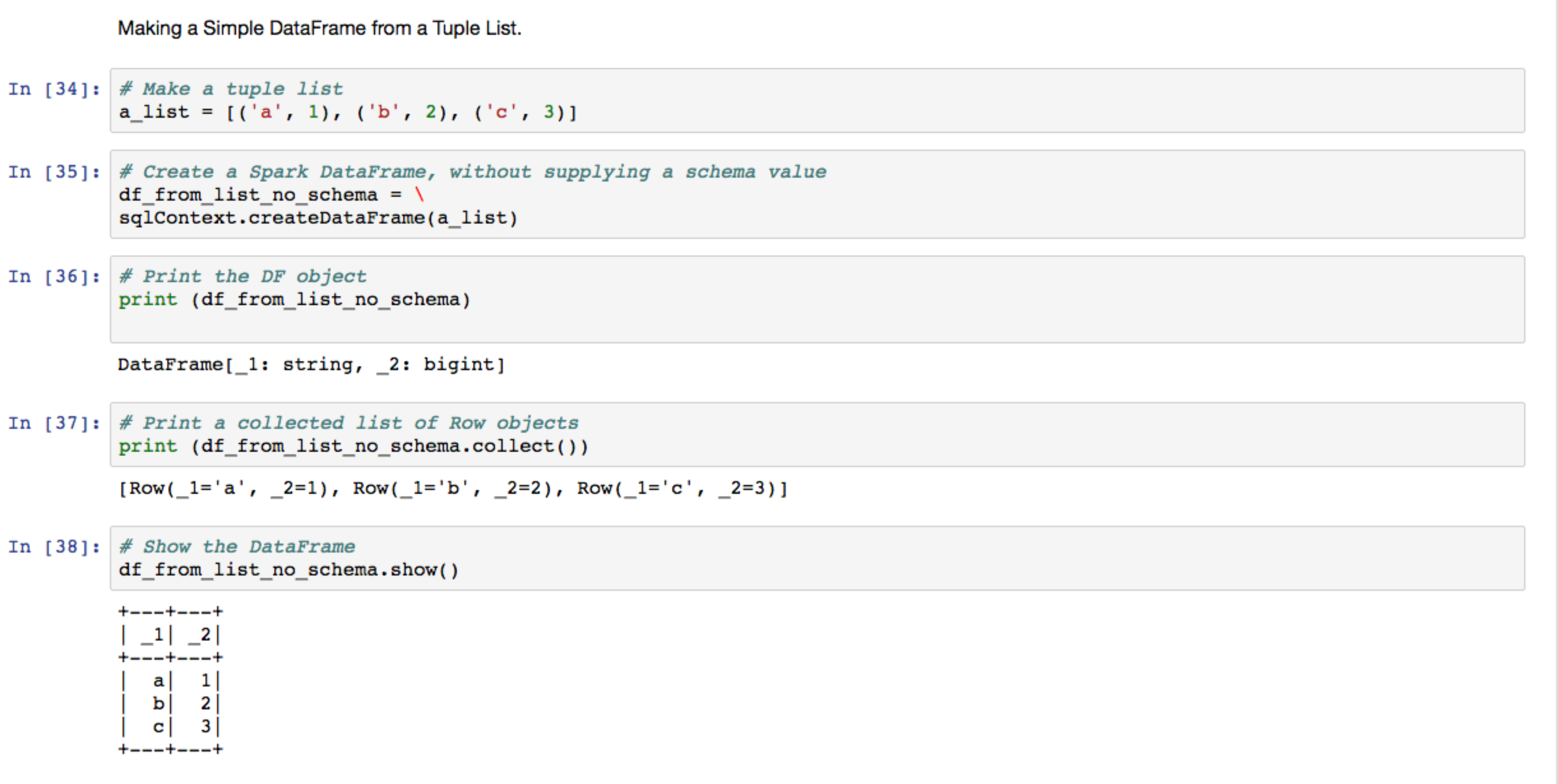
```
In [ ]: #Check the data type
        type(data)
```

Next we’ll do a simple transformation: filter all the lines with “Spark” in them and count such lines.

```
In [ ]: linesWithSpark = data.filter(lambda line: "Spark" in line)
```

```
In [ ]: linesWithSpark.count()
```

Spark DataFrame



Making a Simple DataFrame from a Tuple List.

```
In [34]: # Make a tuple list
         a_list = [('a', 1), ('b', 2), ('c', 3)]
```

```
In [35]: # Create a Spark DataFrame, without supplying a schema value
         df_from_list_no_schema = \
         sqlContext.createDataFrame(a_list)
```

```
In [36]: # Print the DF object
         print (df_from_list_no_schema)
```

```
DataFrame[_1: string, _2: bigint]
```

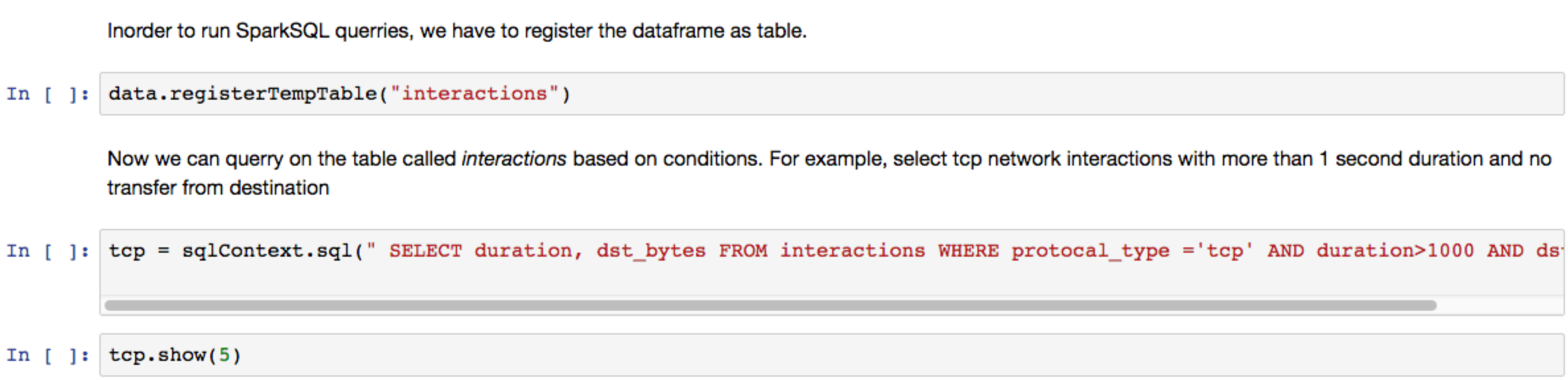
```
In [37]: # Print a collected list of Row objects
         print (df_from_list_no_schema.collect())
```

```
[Row(_1='a', _2=1), Row(_1='b', _2=2), Row(_1='c', _2=3)]
```

```
In [38]: # Show the DataFrame
         df_from_list_no_schema.show()
```

```
+---+---+
| _1| _2|
+---+---+
|  a|  1|
|  b|  2|
|  c|  3|
+---+---+
```

Spark SQL



Inorder to run SparkSQL querries, we have to register the dataframe as table.

```
In [ ]: data.registerTempTable("interactions")
```

Now we can querry on the table called *interactions* based on conditions. For example, select tcp network interactions with more than 1 second duration and no transfer from destination

```
In [ ]: tcp = sqlContext.sql(" SELECT duration, dst_bytes FROM interactions WHERE protocal_type ='tcp' AND duration>1000 AND ds
```

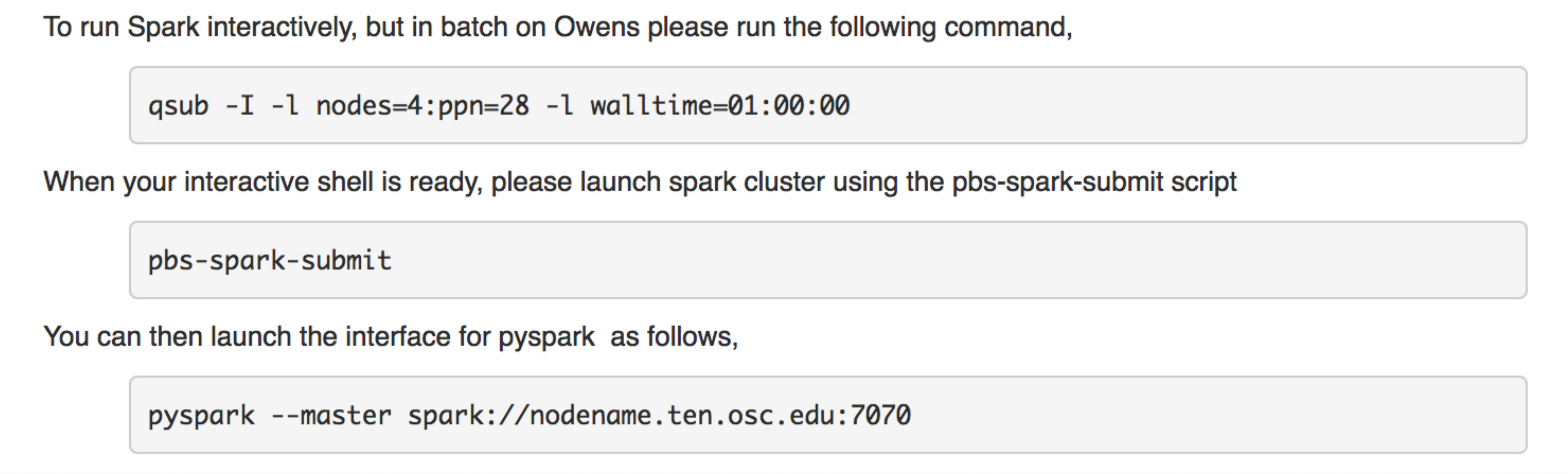
```
In [ ]: tcp.show(5)
```

Spark Mllib

1. Logistic regression: to predict a binary response
2. Kmeans clustering: to clusters the data points into a predefined number of clusters

Exercise

Running Spark interactively in batch



To run Spark interactively, but in batch on Owens please run the following command,

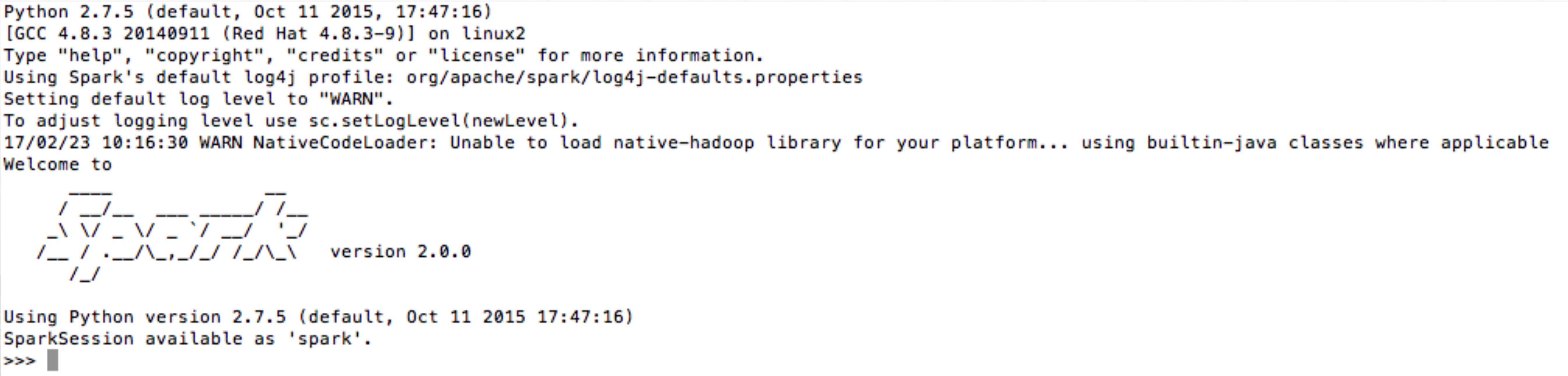
```
qsub -I -l nodes=4:ppn=28 -l walltime=01:00:00
```

When your interactive shell is ready, please launch spark cluster using the pbs-spark-submit script

```
pbs-spark-submit
```

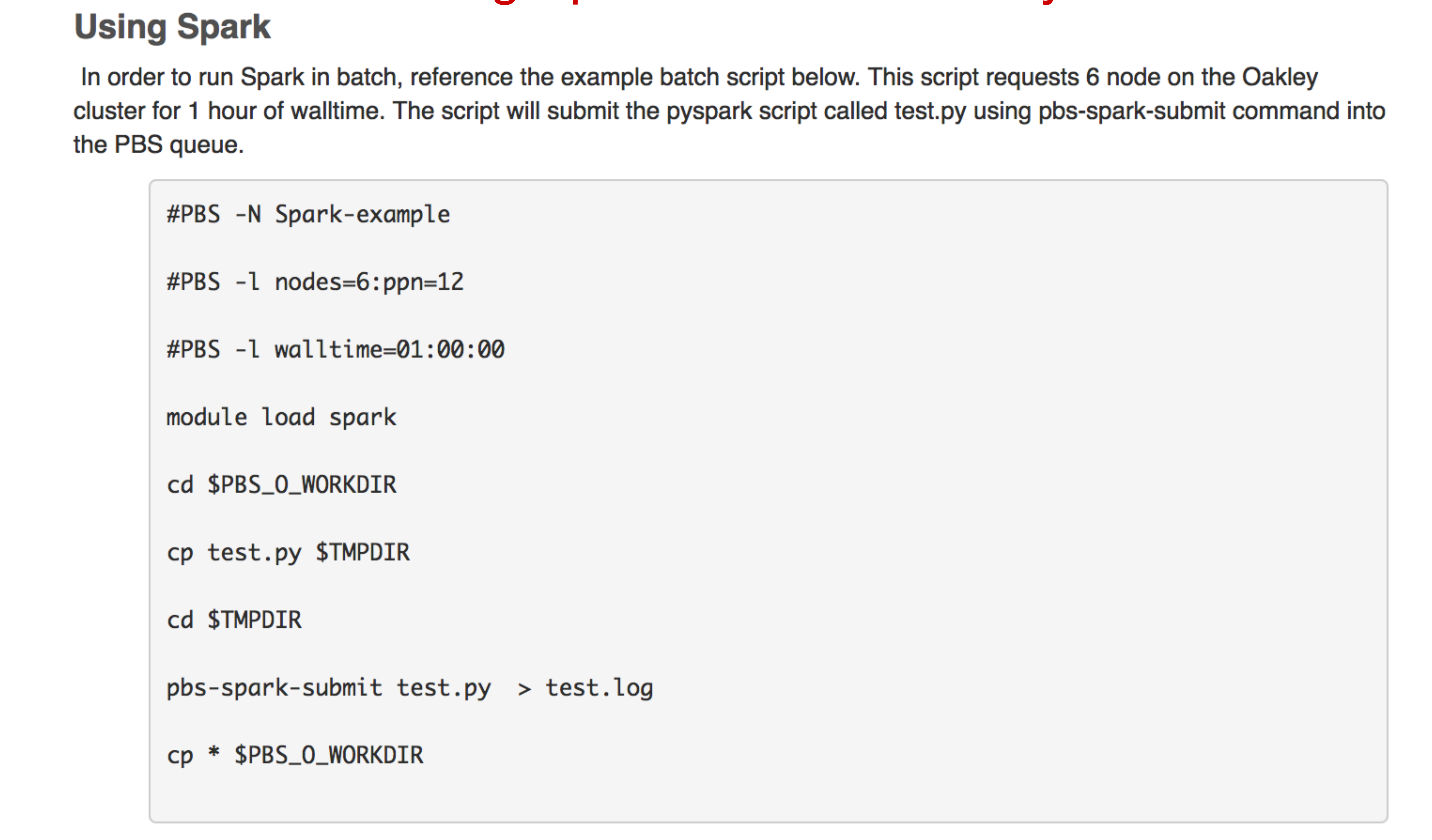
You can then launch the interface for pyspark as follows,

```
pyspark --master spark://nodename.ten.osc.edu:7070
```



```
Python 2.7.5 (default, Oct 11 2015, 17:47:16)
[GCC 4.8.3 20140911 (Red Hat 4.8.3-9)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel).
17/02/23 10:16:30 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.0.0
      /_/

Using Python version 2.7.5 (default, Oct 11 2015 17:47:16)
SparkSession available as 'spark'.
>>> 
```

Running Spark non-interactively



Using Spark

In order to run Spark in batch, reference the example batch script below. This script requests 6 node on the Oakley cluster for 1 hour of walltime. The script will submit the pyspark script called test.py using pbs-spark-submit command into the PBS queue.

```
#PBS -N Spark-example

#PBS -l nodes=6:ppn=12

#PBS -l walltime=01:00:00

module load spark

cd $PBS_O_WORKDIR

cp test.py $TMPDIR

cd $TMPDIR

pbs-spark-submit test.py  > test.log

cp * $PBS_O_WORKDIR
```

Running Spark using PBS script

1. Create an App in python: stati.py

```
from pyspark import SparkContext
import urllib
f = urllib.urlretrieve ("http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data.gz","kddcup.data.gz")

data_file = "./kddcup.data.gz"
sc = SparkContext(appName="Stati")
raw_data = sc.textFile(data_file)

import numpy as np

def parse_interaction(line):
    line_split = line.split(",")
    symbolic_indexes = [1,2,3,41]
    clean_line_split=[item for i, item in enumerate(line_split) if i not in symbolic_indexes]
    return np.array([float(x) for x in clean_line_split])

vector_data=raw_data.map(parse_interaction)


from pyspark.mllib.stat import Statistics
from math import sqrt

summary = Statistics.colStats(vector_data)


print ("Duration Statistics:")
print (" Mean %f" % (round(summary.mean()[0],3)))
print ("St. deviation : %f"%(round(sqrt(summary.variance()[0]),3)))
print (" Max value: %f"%(round(summary.max()[0],3)))
print (" Min value: %f"%(round(summary.min()[0],3)))
```

2. Create a PBS script: stati.pbs

```
#PBS -N spark-statistics
#PBS -l nodes=18:ppn=28
#PBS -l walltime=00:10:00
module load spark/2.0.0
cp stati.py $TMPDIR
cd $TMPDIR
pbs-spark-submit stati.py > stati.log
cp *  $PBS_O_WORKDIR
```

3. Run Spark job

```
qsub stati.pbs
```

4. Output: stati.log

```
sync from spark://n0381.ten.osc.edu:7077
starting org.apache.spark.deploy.master.Master, logging to
/nfs/15/soottikkal/spark/kdd/spark-soottikkal-org.apache.spark.deploy.master.Master-1-
n0381.ten.osc.edu.out
failed to launch org.apache.spark.deploy.master.Master:
full log in /nfs/15/soottikkal/spark/kdd/spark-soottikkal-
org.apache.spark.deploy.master.Master-1-n0381.ten.osc.edu.out

Duration Statistics:
Mean 48.342000
St. deviation : 723.330000
Max value: 58329.000000
Min value: 0.000000
Total value count: 4898431.000000
Number of non-zero values: 118939.000000


SPARK_MASTER=spark://n0381.ten.osc.edu:7077
```

Exercise

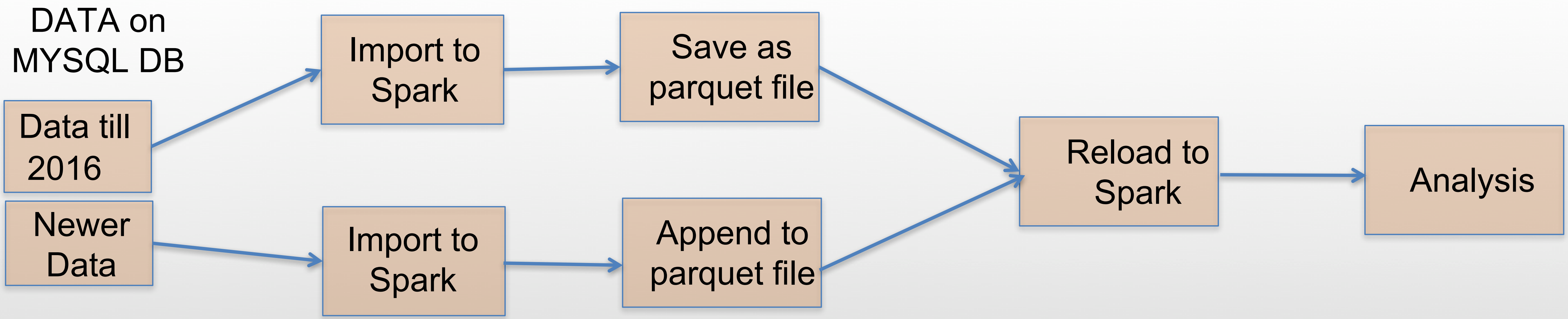
https://www.osc.edu/content/submitting_non_interactive_jobs

CASE STUDY

Data mining of historical jobs records of OSC's clusters

Aim: To understand client utilizations of OSC recourses.

Data: Historical records of every Job that ran on any OSC clusters that includes information's such as number of nodes, software, CPU time and timestamp.



DATA on MYSQL DB

Data till 2016

Newer Data

Import to Spark

Import to Spark

Save as parquet file

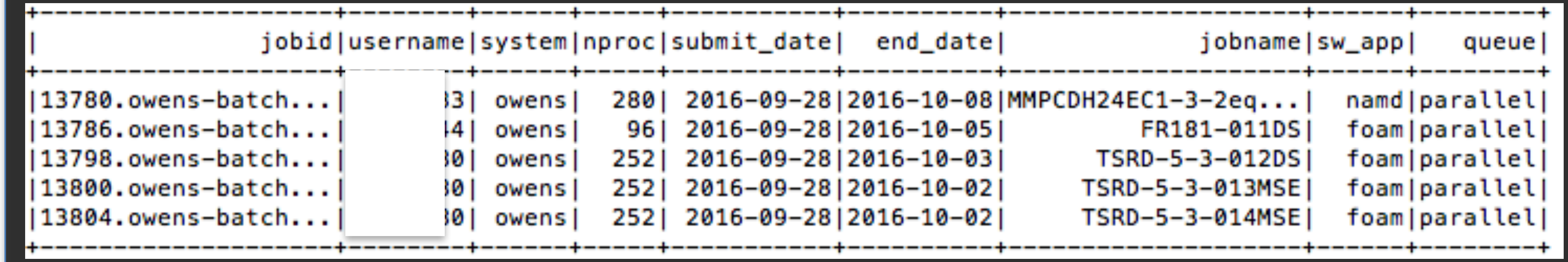
Append to parquet file

Reload to Spark

Analysis

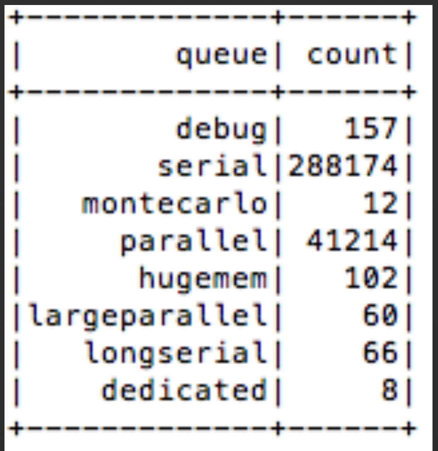
Pyspark code for data analysis

```
#importing data
df=sqlContext.read.parquet("/fs/scratch/pbsacct/Jobs.parquet")
df.show(5)
```



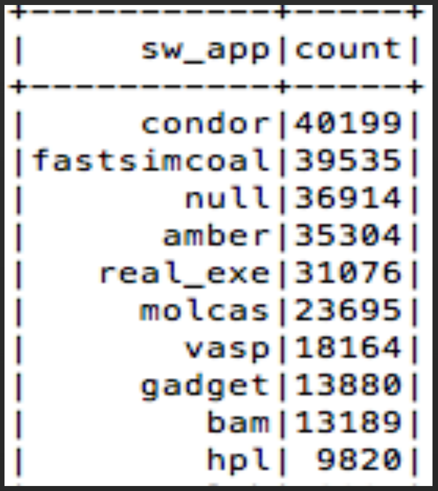
```
+--------------------+--------+------+-----+-----------+----------+--------------------+------+--------+
|               jobid|username|system|nproc|submit_date|  end_date|             jobname|sw_app|   queue|
+--------------------+--------+------+-----+-----------+----------+--------------------+------+--------+
|13780.owens-batch...|[redacted]3| owens|  280| 2016-09-28|2016-10-08|MMPCDH24EC1-3-2eq...|  namd|parallel|
|13786.owens-batch...|[redacted]4| owens|   96| 2016-09-28|2016-10-05|          FR181-011DS|  foam|parallel|
|13798.owens-batch...|[redacted]0| owens|  252| 2016-09-28|2016-10-03|        TSRD-5-3-012DS|  foam|parallel|
|13800.owens-batch...|[redacted]0| owens|  252| 2016-09-28|2016-10-02|       TSRD-5-3-013MSE|  foam|parallel|
|13804.owens-batch...|[redacted]0| owens|  252| 2016-09-28|2016-10-02|       TSRD-5-3-014MSE|  foam|parallel|
+--------------------+--------+------+-----+-----------+----------+--------------------+------+--------+
```

```
#Which types of queue is mostly used
df.select("jobid",”queue").groupBy("queue").count().show()
```



```
+-------------+------+
|        queue| count|
+-------------+------+
|        debug|   157|
|       serial|288174|
|   montecarlo|    12|
|     parallel| 41214|
|      hugemem|   102|
|largeparallel|    60|
|   longserial|    66|
|    dedicated|     8|
+-------------+------+
```

```
#Which software is used most?
df.select("jobid","sw_app").groupBy
("sw_app").count().sort(col("count").desc()) .show()
```

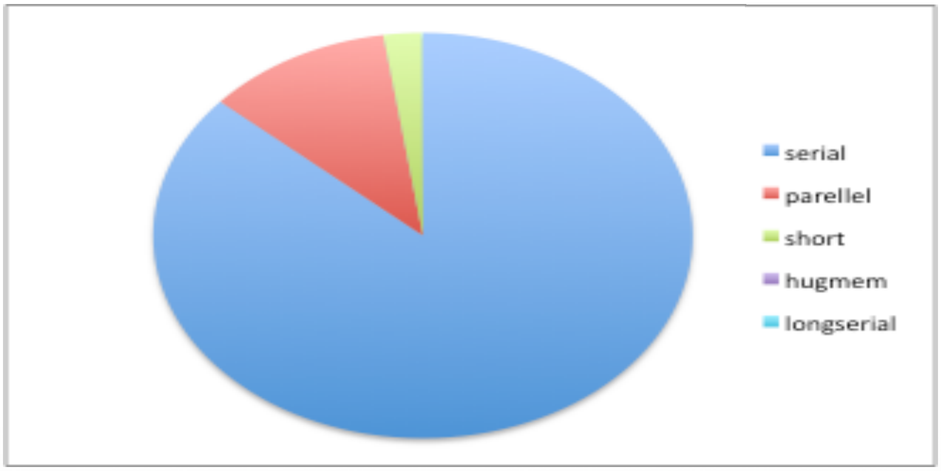


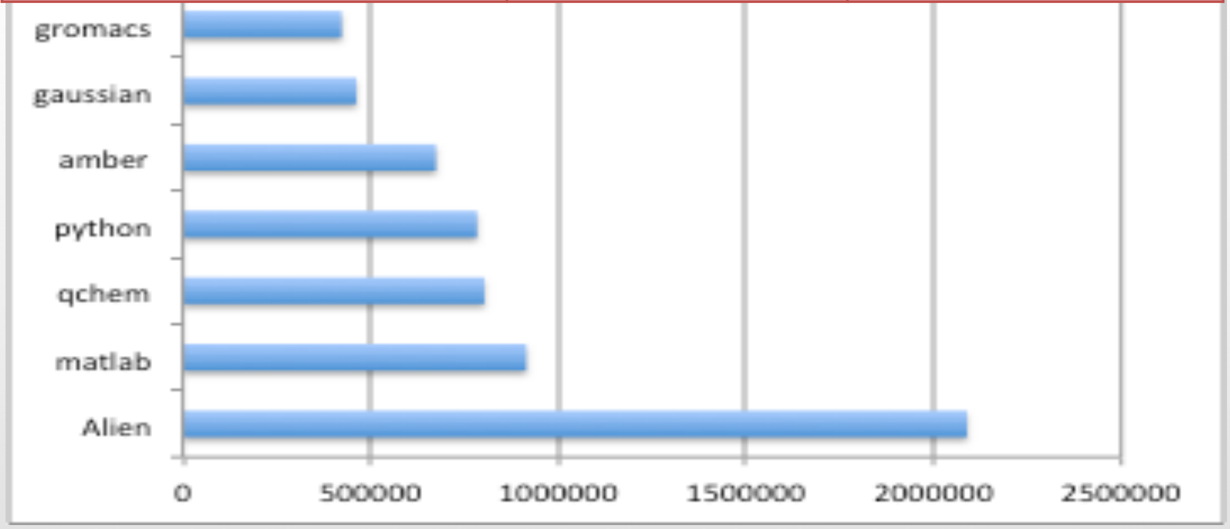
```
+-----------+-----+
|     sw_app|count|
+-----------+-----+
|     condor|40199|
|fastsimcoal|39535|
|       null|36914|
|      amber|35304|
|   real_exe|31076|
|     molcas|23695|
|       vasp|18164|
|     gadget|13880|
|        bam|13189|
|        hpl| 9820|
```

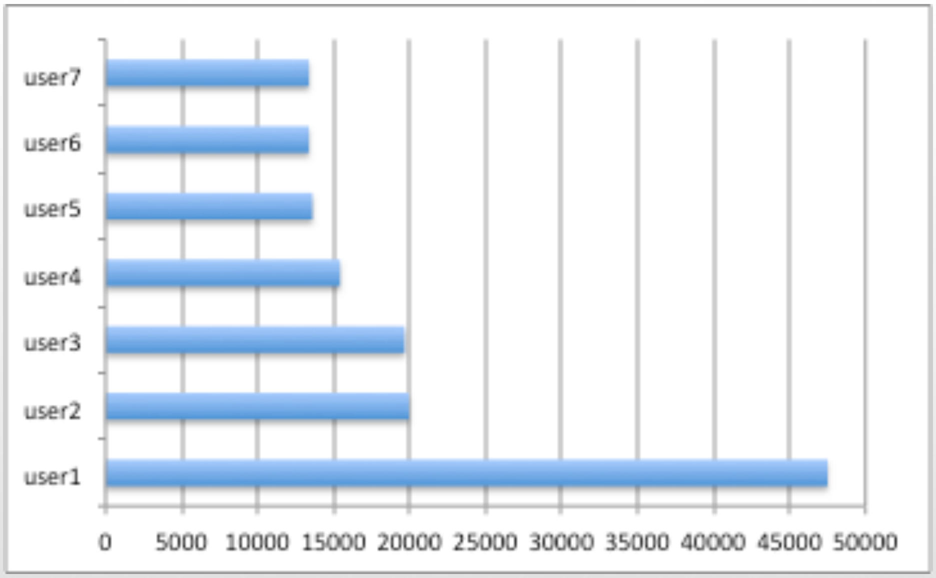
```
#who uses gaussian software most?
df.registerTempTable(”Jobs”)
sqlContext.sql(" SELECT username FROM
Jobs WHERE sw_app='gaussian’ " ).show()
```

Results

Statistics	MYSQL	SPARK
Job vs CPU	1 hour	5 sec
CPU vs Account	1.25 hour	5 sec
Walltime vs user	1.40 hour	5 sec

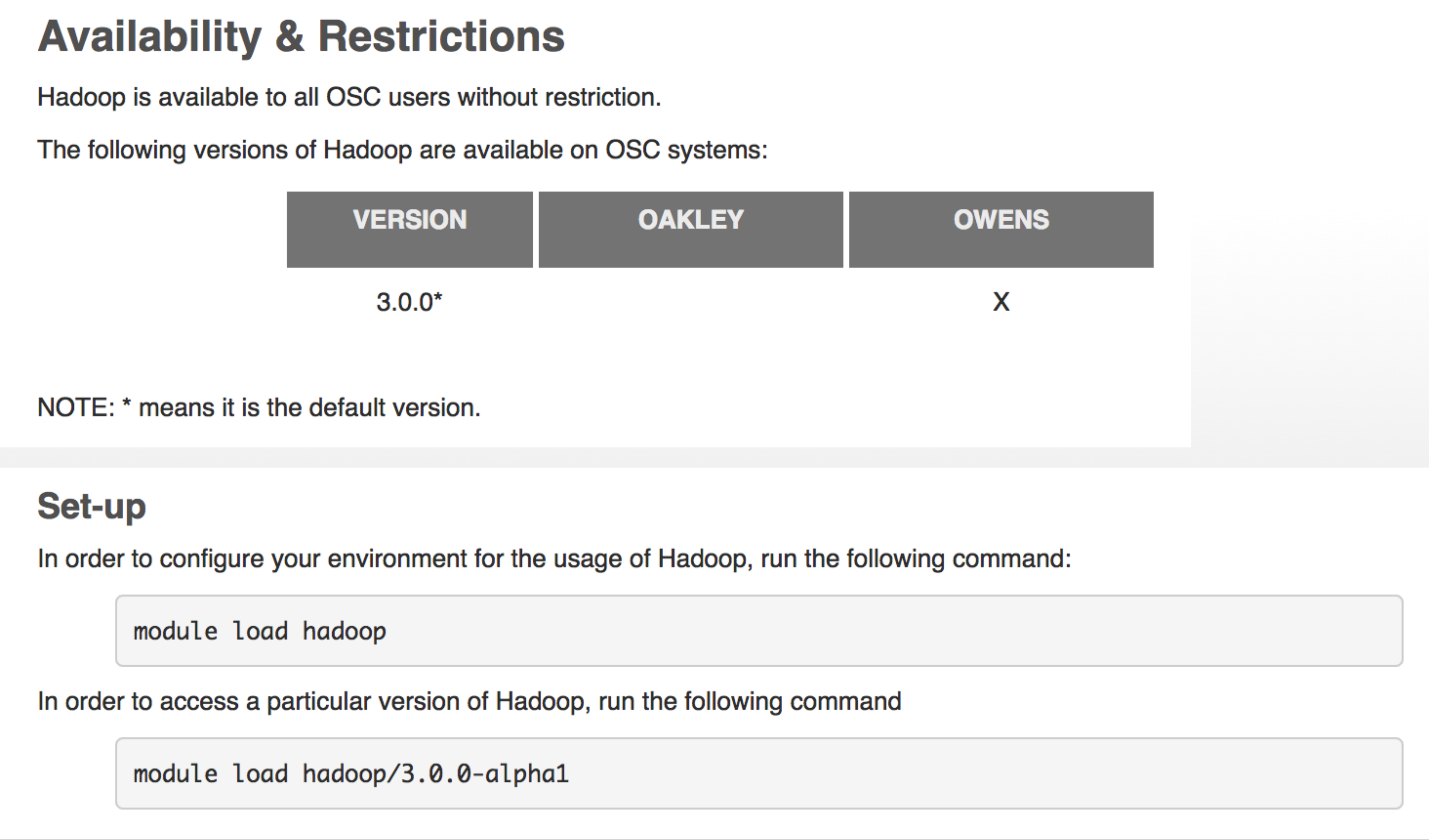






Running Hadoop at OSC

A Hadoop cluster can be launched within the HPC environment, but managed by the PBS job scheduler using Myhadoop framework developed by San Diego Supercomputer Center. (Please see http://www.sdsc.edu/~allans/MyHadoop.pdf)



Availability & Restrictions

Hadoop is available to all OSC users without restriction.

The following versions of Hadoop are available on OSC systems:

VERSION	OAKLEY	OWENS
3.0.0*		X

NOTE: * means it is the default version.

Set-up

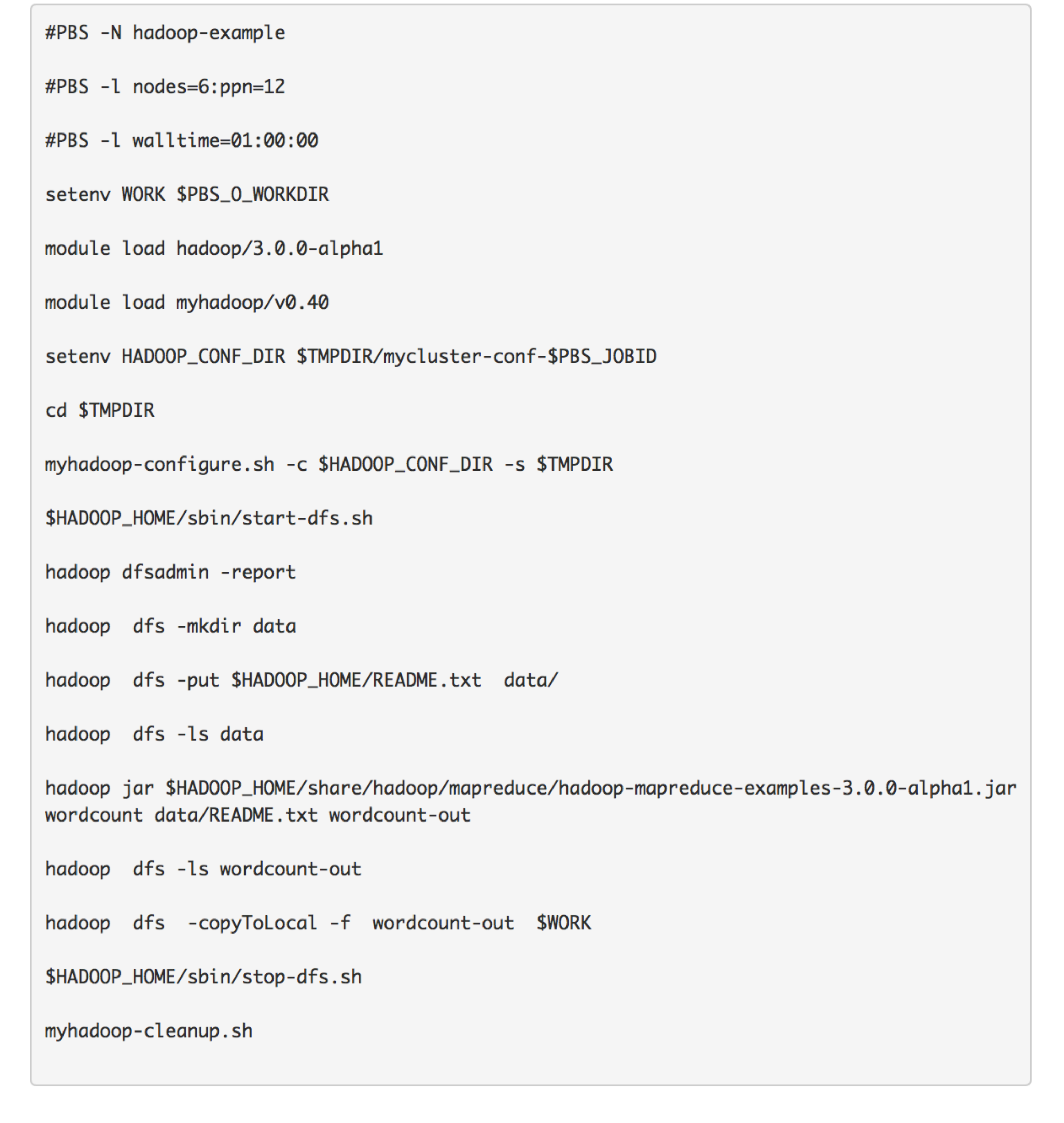
In order to configure your environment for the usage of Hadoop, run the following command:

```
module load hadoop
```

In order to access a particular version of Hadoop, run the following command

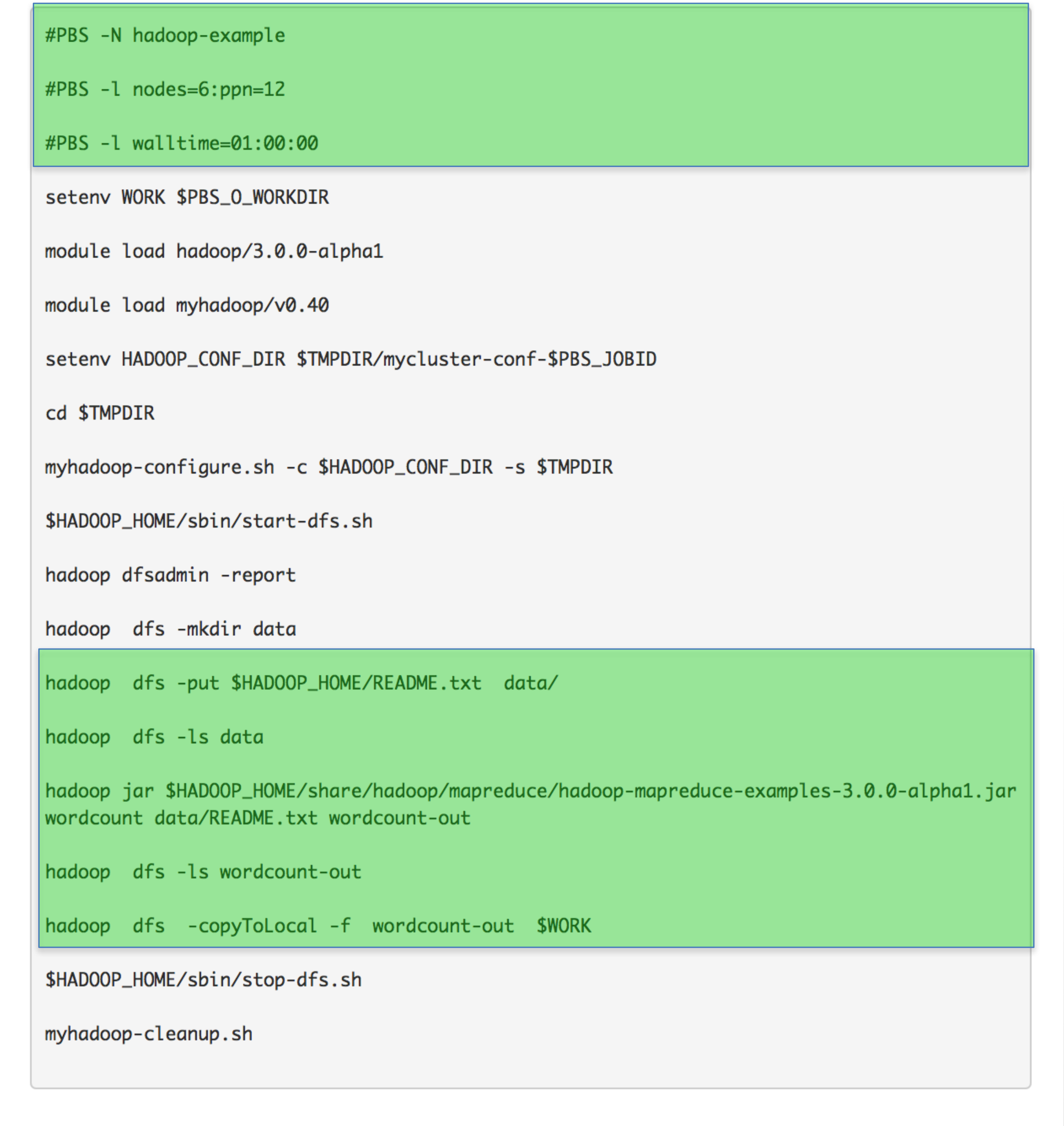
```
module load hadoop/3.0.0-alpha1
```

Using Hadoop: Sample PBS Script



```
#PBS -N hadoop-example

#PBS -l nodes=6:ppn=12

#PBS -l walltime=01:00:00

setenv WORK $PBS_O_WORKDIR

module load hadoop/3.0.0-alpha1

module load myhadoop/v0.40

setenv HADOOP_CONF_DIR $TMPDIR/mycluster-conf-$PBS_JOBID

cd $TMPDIR

myhadoop-configure.sh -c $HADOOP_CONF_DIR -s $TMPDIR

$HADOOP_HOME/sbin/start-dfs.sh

hadoop dfsadmin -report

hadoop  dfs -mkdir data

hadoop  dfs -put $HADOOP_HOME/README.txt  data/

hadoop  dfs -ls data

hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.0.0-alpha1.jar
wordcount data/README.txt wordcount-out

hadoop  dfs -ls wordcount-out

hadoop  dfs  -copyToLocal -f  wordcount-out  $WORK

$HADOOP_HOME/sbin/stop-dfs.sh

myhadoop-cleanup.sh
```

Using Hadoop: Sample PBS Script



```
#PBS -N hadoop-example

#PBS -l nodes=6:ppn=12

#PBS -l walltime=01:00:00

setenv WORK $PBS_O_WORKDIR

module load hadoop/3.0.0-alpha1

module load myhadoop/v0.40

setenv HADOOP_CONF_DIR $TMPDIR/mycluster-conf-$PBS_JOBID

cd $TMPDIR

myhadoop-configure.sh -c $HADOOP_CONF_DIR -s $TMPDIR

$HADOOP_HOME/sbin/start-dfs.sh

hadoop dfsadmin -report

hadoop  dfs -mkdir data

hadoop  dfs -put $HADOOP_HOME/README.txt  data/

hadoop  dfs -ls data

hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.0.0-alpha1.jar
wordcount data/README.txt wordcount-out

hadoop  dfs -ls wordcount-out

hadoop  dfs  -copyToLocal -f  wordcount-out  $WORK

$HADOOP_HOME/sbin/stop-dfs.sh

myhadoop-cleanup.sh
```

Exercise

References

1. Spark Programming Guide

https://spark.apache.org/docs/2.0.0/programming-guide.html

-Programming with Scala, Java and Python

2. Data Exploration with Spark

http://www.cs.berkeley.edu/~rxin/ampcamp-ecnu/data-exploration-using-spark.html

3. Hadoop

http://hadoop.apache.org/

4. OSC Documentation

https://www.osc.edu/documentation/software_list/spark_documentation
https://www.osc.edu/resources/available_software/software_list/hadoop

Thank you!

- Questions or comments: soottikkal@osc.edu
- General questions about OSC service: oschelp@osc.edu