Here we will download **Spark** binaries on our AWS instance, fire up some **Spark** clusters, and try out **RStudio Server**.

- Creating Clusters
- Connecting to the Master Cluster
- Trying RStudio

Creating Clusters

Now, let's get our clusters up and running. This will call the script to launch 3 clusters, 1 master and 2 dependents in the `us-west` zone. `-k` is your PEM name, `-i` is the path to the PEM file, `-s` is the number of instances needed and `-t` is the server type:

```
# launch clusters
./spark-ec2 -k udemy -i udemy.pem -r us-west-2 -s 2 -t
m1.small launch --copy-aws-credentials my-spark-cluster
```

Building the clusters takes about 15 minutes. The terminal window may spit out some `waiting`, `disconnected`, or `connection refused` messages but as long as it doesn't return the prompt, let it do its thing as it needs to set up each instance and install a lot of software on all of them.

```
[ec2-user@ip-10-0-0-121 ec2]$ # launch clusters
[ec2-user@ip-10-0-0-121 ec2]$ ./spark-ec2 -k demo -i demo.pem -r us-west-2 -s 2
-t m1.small launch --copy-aws-credentials my-spark-cluster
Downloading external libraries that spark-ec2 needs from PyPI to /home/ec2-user/
spark-1.5.0-bin-hadoop2.6/ec2/lib...
This should be a one-time operation.
 - Downloading boto...
 - Finished downloading boto.
Setting up security groups...
Creating security group my-spark-cluster-master
Creating security group my-spark-cluster-slaves
Searching for existing cluster my-spark-cluster in region us-west-2...
Spark AMI: ami-9a6e0daa
Launching instances...
Launched 2 slaves in us-west-2a, regid = r-0a1a52fc
Launched master in us-west-2a, regid = r-a01a5256
Waiting for AWS to propagate instance metadata...
Waiting for cluster to enter 'ssh-ready' state....
```

Connecting to the Master Cluster

One of the last lines in the terminal window will be the master IP address - but you can also get it from the instance viewer on the AWS website.

The master cluster is where we need to run RStudio. It is automatically installed for us (nice!). To check that our clusters are up and running, connect to the `Spark Master` window (swap the master IP address with yours):

```
http://52.13.38.224:8080/
```

Workers

Worker Id	Address	State	Cores	Memory
worker-20151001053248-10.35.132.112-46660	10.35.132.112:46660	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)
worker-20151001053248-10.36.39.199-35928	10.36.39.199:35928	ALIVE	1 (0 Used)	1024.0 MB (0.0 B Used)

We confirm that we have two dependent clusters with status `Alive`. In order to log into RStudio, we need to first SSH into the instance and create a new user.

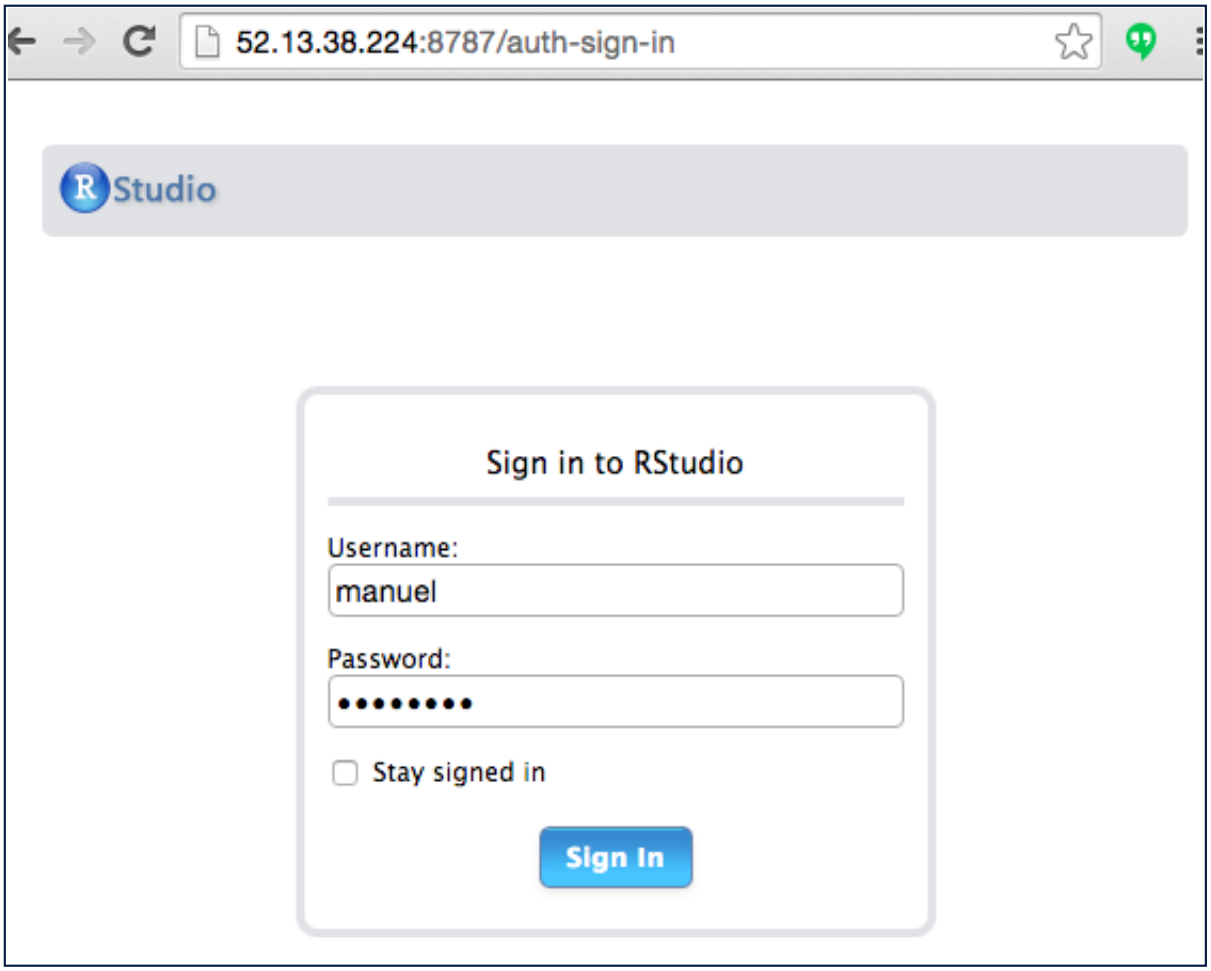
```
# log into master instance
ssh -i "udemy.pem" ec2-52-13-38-224.us-west-
2.compute.amazonaws.com

# add new user (and you don't have to use my name) and set a
passwd
sudo useradd manuel
sudo passwd manuel
```

Working with RStudio

The credentials we just created are what we'll use to log into `RStudio Server`:

```
# log into r studio
http://52.13.38.224:8787/
```



Let’s get some data through these clusters

Just a quick example from the 1.5 Spark documentation to show that this works. Paste the following code into RStudio running on your master cluster:

```
library("SparkR", lib.loc="/root/spark/R/lib")
Sys.setenv(SPARK_HOME="/root/spark")
```

```
# initialize the Spark Context
sc <- sparkR.init()
sqlContext <- sparkRSQL.init(sc)

# create a data frame using the createDataFrame object
df <- createDataFrame(sqlContext, faithful)
head(df)
```

```
##   eruptions waiting
## 1     3.600      79
## 2     1.800      54
## 3     3.333      74
## 4     2.283      62
## 5     4.533      85
## 6     2.883      55
```

```
# try simple generalized linear model
model <- glm(waiting ~ eruptions, data = df, family =
"gaussian")
summary(model)
```

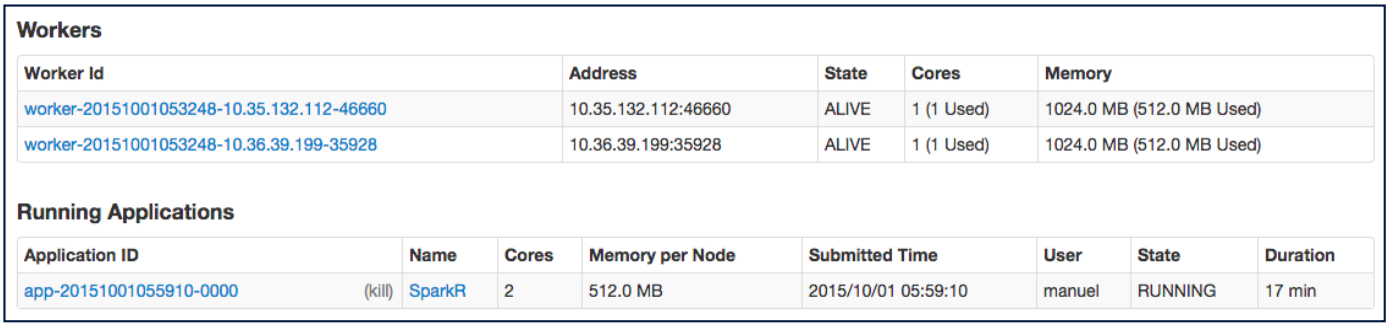
```
## $coefficients
##             Estimate
## (Intercept) 33.47440
## eruptions   10.72964
```

```
# see how well it predicts
predictions <- predict(model, newData = df)
head(select(predictions, "waiting", "prediction"))
```

```
##   waiting prediction
## 1      79   72.10111
## 2      54   52.78775
## 3      74   69.23629
## 4      62   57.97017
## 5      85   82.11186
## 6      55   64.40795
```

And to prove that all clusters went to work, check out the `Spark Master`:

```
http://52.13.38.224:8080/
```



Workers

Worker Id	Address	State	Cores	Memory
worker-20151001053248-10.35.132.112-46660	10.35.132.112:46660	ALIVE	1 (1 Used)	1024.0 MB (512.0 MB Used)
worker-20151001053248-10.36.39.199-35928	10.36.39.199:35928	ALIVE	1 (1 Used)	1024.0 MB (512.0 MB Used)

Running Applications

Application ID		Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20151001055910-0000	(kill)	SparkR	2	512.0 MB	2015/10/01 05:59:10	manuel	RUNNING	17 min

K- we're done for now. Don't forget to terminate all your EC2 instances you created!

```
# handy command to kill all Spark clusters part of my-spark-cluster
./spark-ec2 -k udemy -i udemy.pem -r us-west-2  destroy my-spark-cluster
```

Next section we'll run more **GLM** models on different data sets.