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

- Spark Documentation - Modeling the Iris Dataset
- Lars - Modeling Diabetes Progression

Spark Documentation - Modeling the Iris Dataset

Here is another simple example from the Spark documentation

```
# Create the DataFrame
df <- createDataFrame(sqlContext, iris)

# Fit a linear model over the dataset.
model <- glm(Sepal_Length ~ Sepal_Width + Species, data = df,
family = "gaussian")

# Model coefficients are returned in a similar format to R's
native glm().
summary(model)

# Make predictions based on the model.
predictions <- predict(model, newData = df)
head(select(predictions, "Sepal_Length", "prediction"))
```

Lars - Modeling Diabetes Progression

```
install.packages('lars')
library(lars)
data(diabetes)
diabetes_all <- data.frame(cbind(diabetes$x, y = diabetes$y))
head(diabetes_all)
outcome_name <- 'y'
diabetes_all$sex <- as.numeric(as.factor(diabetes_all$sex ))
dim(diabetes_all)

# split data into training and testing
set.seed(1234)
splitIndex <- base::sample(nrow(diabetes_all),
floor(0.75*nrow(diabetes_all)))
train_diabetes <- diabetes_all[ splitIndex,]
test_diabetes <- diabetes_all[-splitIndex,]

# Convert local data frame/RDD/etc to a SparkR DataFrame
train_diabetes_sp <- createDataFrame(sqlContext,
train_diabetes)
test_diabetes_sp <- createDataFrame(sqlContext,
test_diabetes)

# Fit a linear model over the dataset.
model <- glm(y~age+sex+bmi+map+tc+ldl+hdl+tch+ltg+glu,
        data=train_diabetes_sp, family='gaussian')
summary(model)

# Use test data set to predict y
predictions <- predict(model, newData = test_diabetes_sp)
names(predictions)

predictions_details <- select(predictions, predictions$label,
predictions$prediction)

# collect errors and calculate the mean squared error (MSE)
predictions_details <- collect(predictions_details)
mse <- mean((predictions_details$label -
predictions_details$prediction)^2)
print(mse)
```