Microsoft

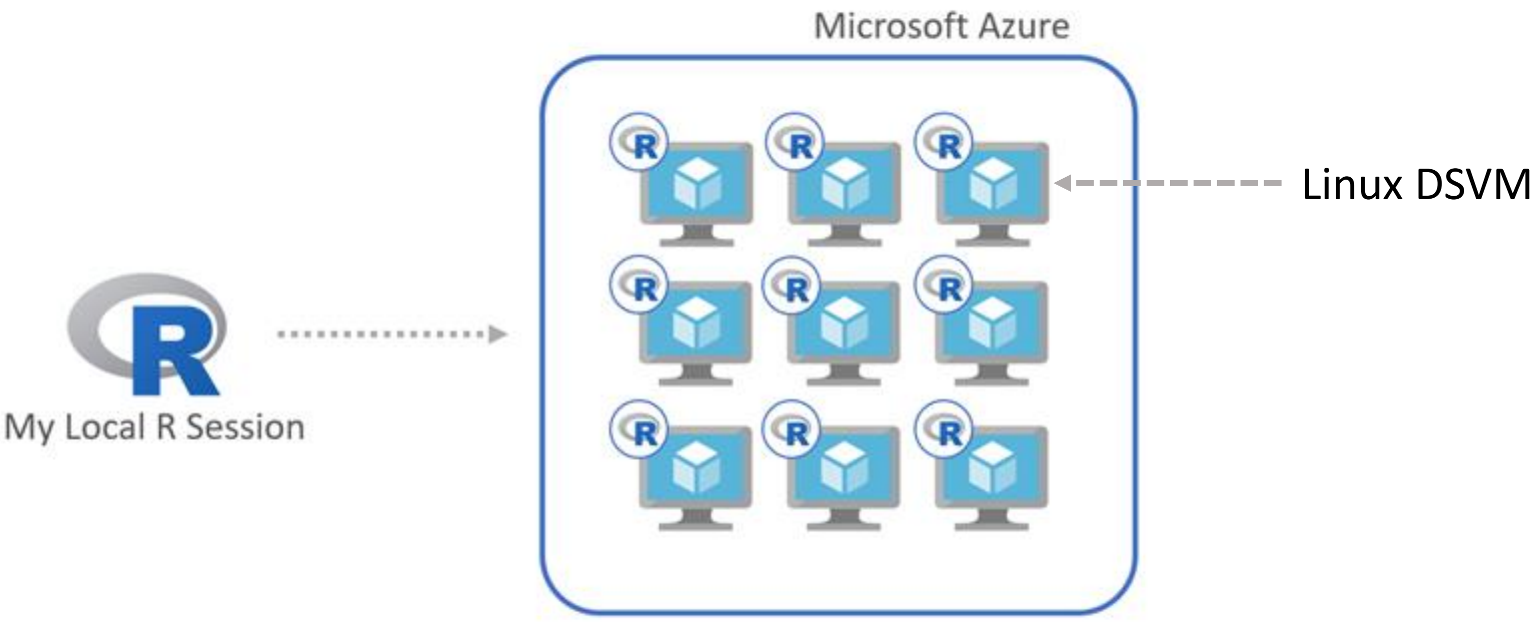
R Parallel Programming in the Azure HPC Batch

Adrian.Fernandez@microsoft.com

@adrianfz10

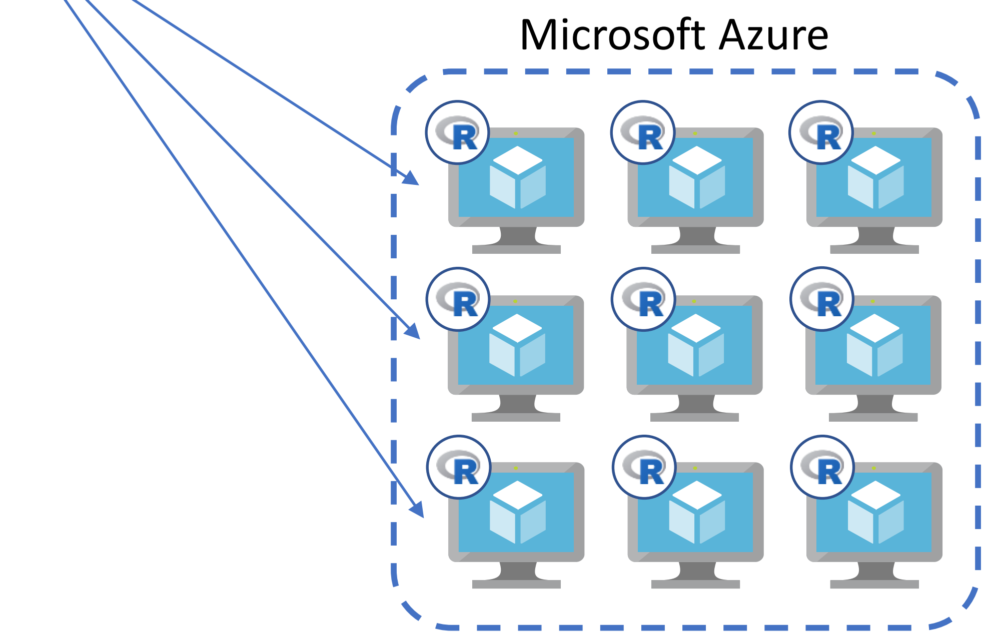
doAzureParallel is... R Package for parallel procs

A simple R package that uses Azure as a parallel-backend for popular open source tools to use –foreach, caret, plyr, etc.



Foreach using doAzureParallel

```
library(doAzureParallel)
foreach (i = 1:100) %dopar% {
    myParallelAlgorithm(...)
}
```



doAzureParallel on Azure Batch

Azure Batch is a platform HPC as a Services that provides easy job scheduling and cluster management, allowing applications or algorithms to run in parallel at scale.

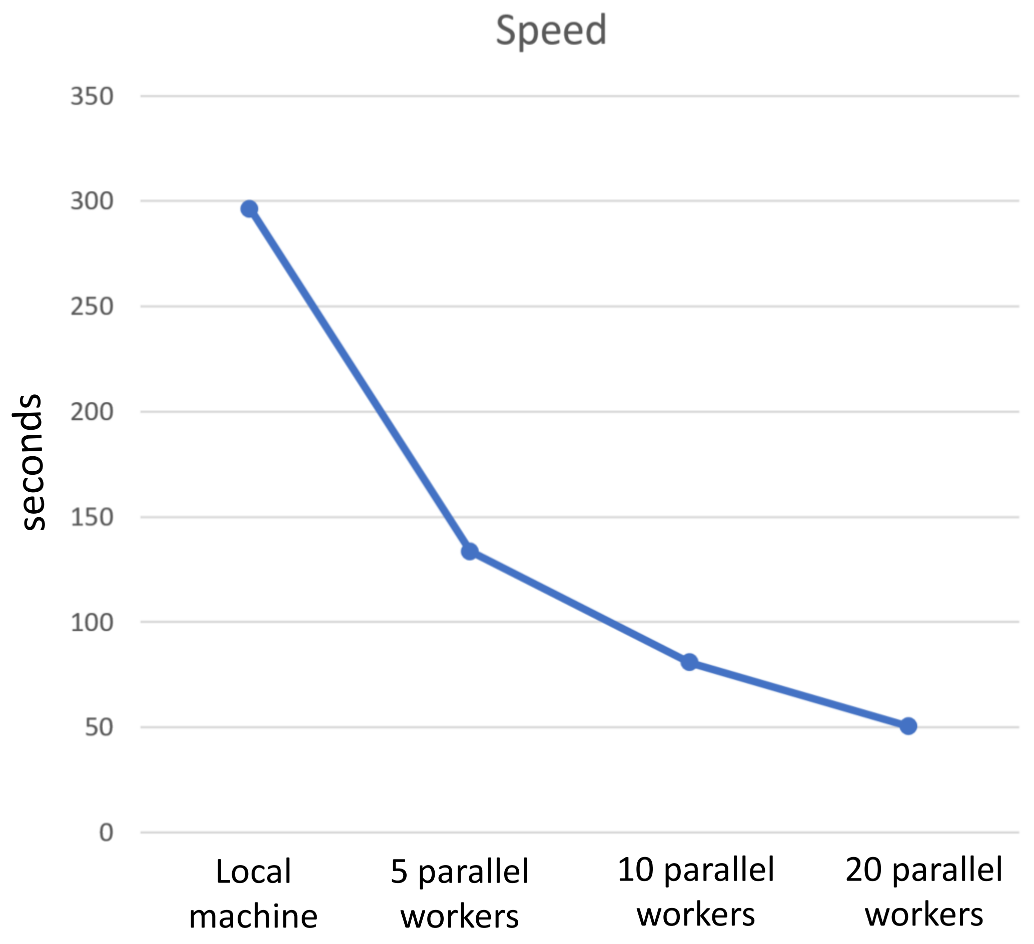
- Capacity on demand; jobs on demand
- Autoscale (more on this later)
- Minimal cluster management (node failure, install, etc)
- Hardware choice – use any VM size
- Pay by the minute
- Cost effective – no charge for using it, you only pay for the VMs
- More cost effective – low priority VMs (more on this later)
- Completely containerized

If you want to run jobs using elastic compute, Batch is a great fit!

Scale

- From 1 to 10,000 VMs for a cluster
- From 1 to millions of tasks
- Your selection of hardware:
  - General compute VMs (A-Series / D-Series)
  - Memory / storage optimized (G-Series)
  - Compute Optimized (F-Series)
  - GPU enabled (N-Series)

- Results from computing the **Mandelbrot** set when scaling up:



Elastic Compute

- Compute on-demand
  - Create/delete your cluster as you need

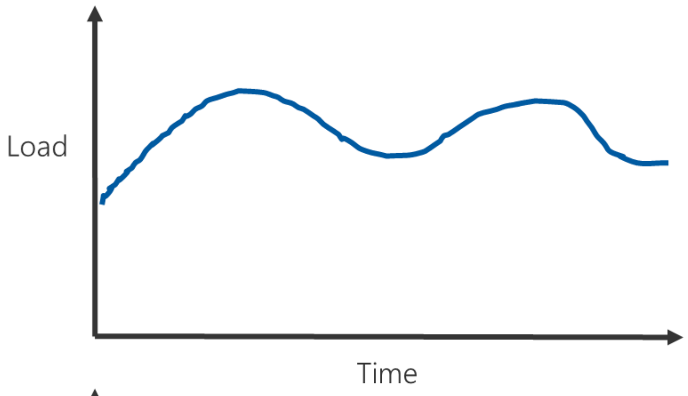
- Autoscaling pool = maximizing cloud elasticity
  - Long running batch jobs / overnight
  - Daily scheduled work – pre-provision cluster so its ready for you at the beginning of the day
  - Bursty work

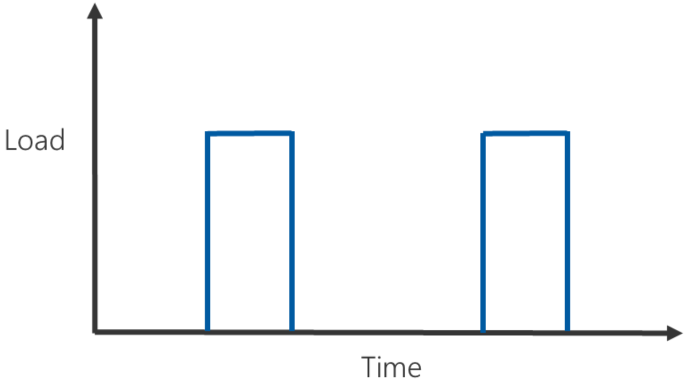
Elasticity & Scale

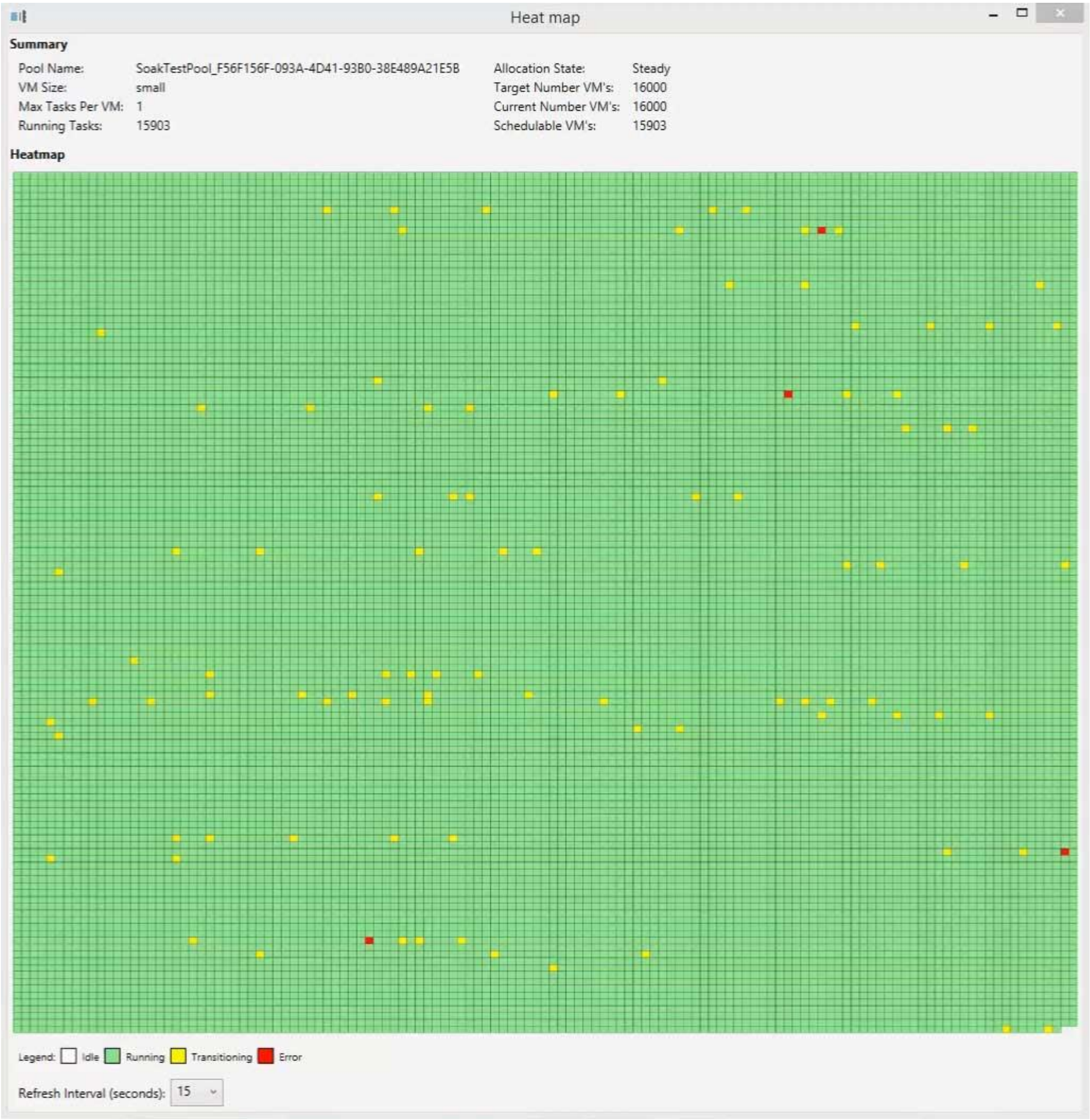
https://blogs.endjin.com/2015/07/spinning-up-16000-a1-virtual-machines-on-azure-batch/

What would you do with 100,000 cores? – Big compute at global scale (16.000 cores)

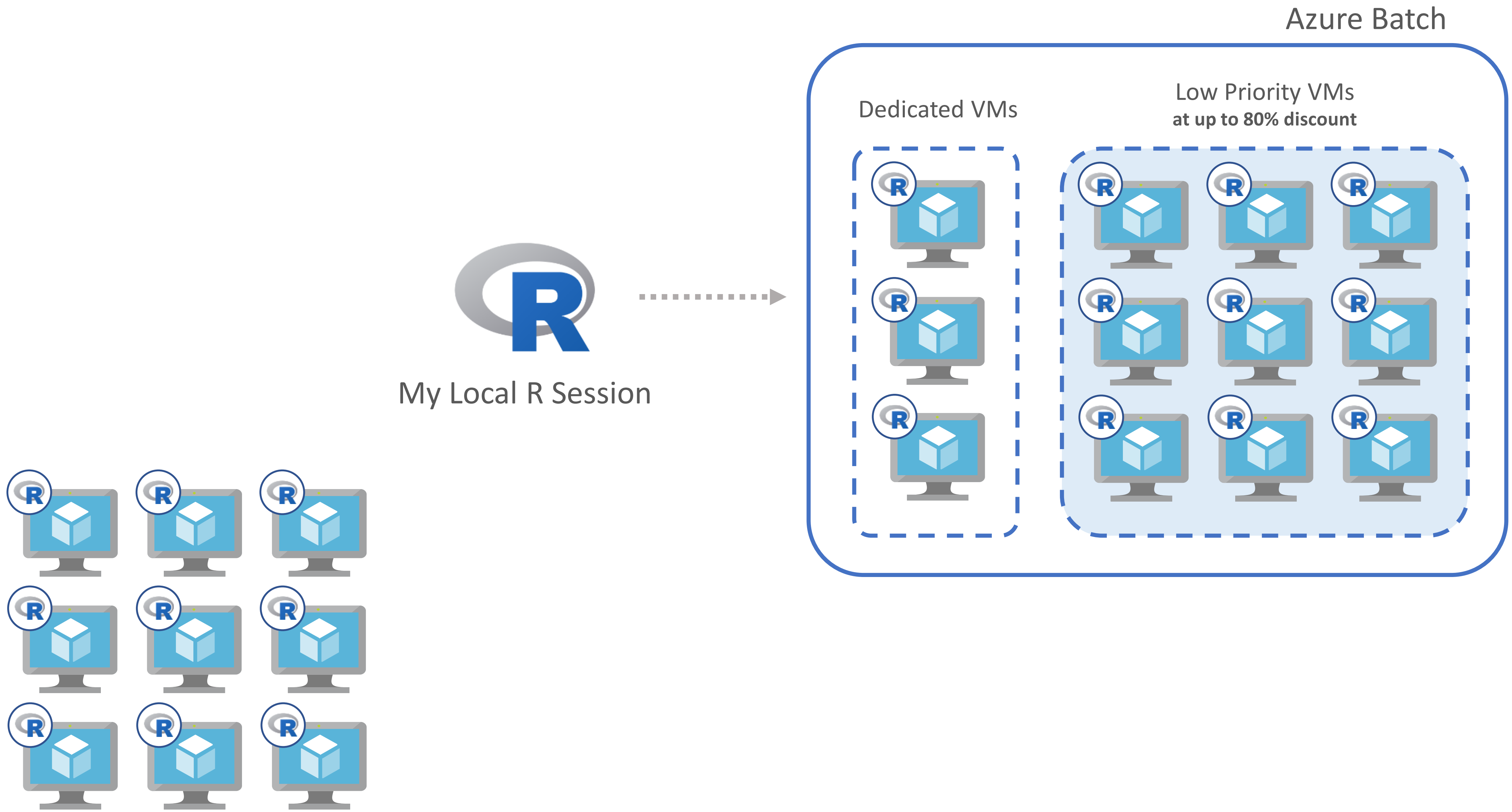
https://azure.microsoft.com/en-us/blog/what-would-you-do-with-100000-cores-big-compute-at-global-scale/







R + Azure DEMO


Azure Batch
Dedicated VMs
Low Priority VMs
at up to 80% discount
My Local R Session

Monte Carlo Pricing - HPC Simulation - %dopar%

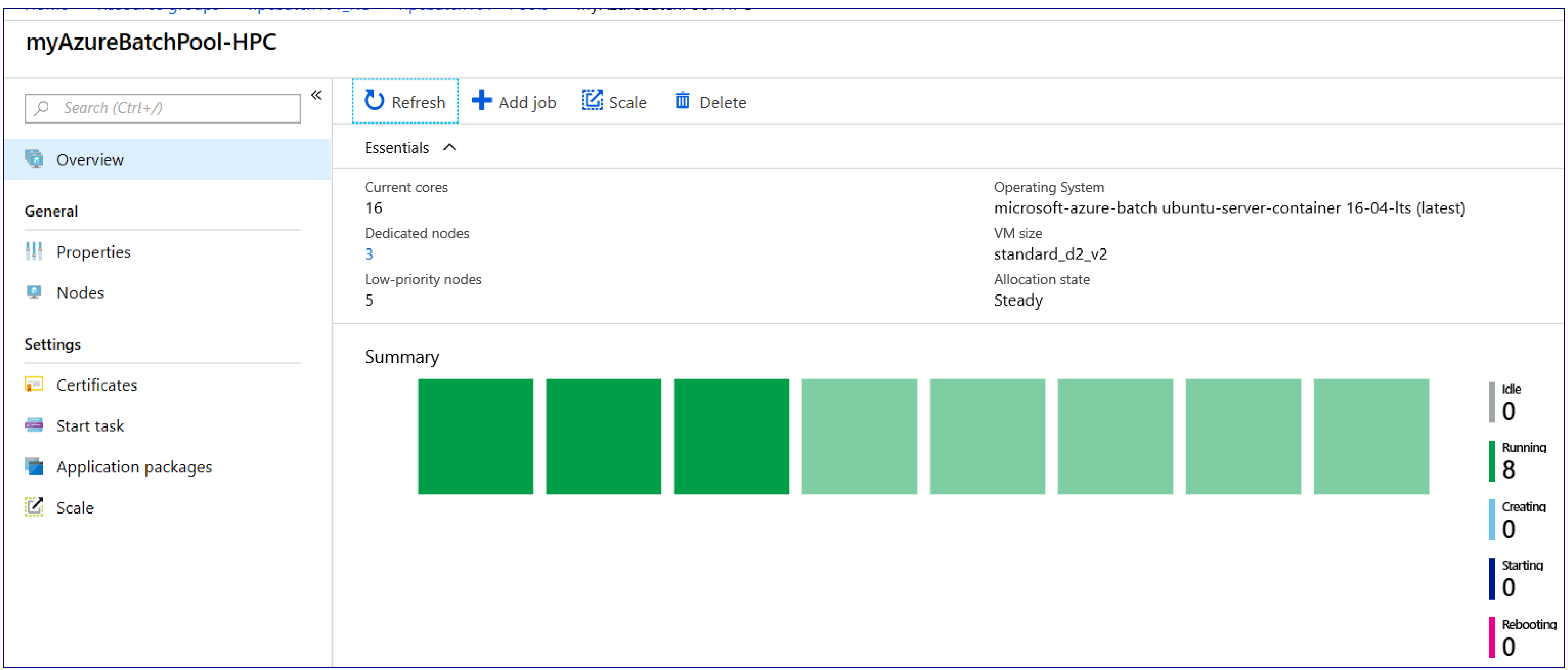
8-node cluster (standard D2v2: 2 vCPU, 7 Gb)

- specify VM class in `cluster.json`
- specify credentials for Azure Batch and Azure Storage in `credentials.json`

```r
# Estimate runtime for 1 million (linear approximation) (1000 x 1000)
1000 * difftime(end_s, start_s, unit = "min")

# Run 1 million simulations with doAzureParallel
# We will run 100 iterations where each iteration executes 10,000 simulations
opt <- list(chunkSize = 20) # optimizie runtime. Chunking allows us to run multiple iteratio

## %dopar% ## AZURE BATCH COMPUTATION
start_p <- Sys.time()
closingPrices_p <- foreach(i = 1:1000, .combine='c', .options.azure = opt) %dopar% {
  replicate(1000, getClosingPrice())
}
end_p <- Sys.time()
```

```json
{
  "name": "myAzureBatchPool-HPC",
  "vmSize": "Standard_D2_v2",
  "maxTasksPerNode": 4,
  "poolSize": {
    "dedicatedNodes": {
      "min": 3,
      "max": 5
    },
    "lowPriorityNodes": {
      "min": 5,
      "max": 5
    },
    "autoscaleFormula": "QUEUE"
  },
  "containerImage": "rocker/tidyverse:latest",
  "rPackages": {
    "cran": [],
    "github": [],
```

45 seconds (more than **5 times faster**) on a warm start


myAzureBatchPool-HPC
Search (Ctrl+/)
Refresh
Add job
Scale
Delete
Overview
General
Properties
Nodes
Settings
Certificates
Start task
Application packages
Scale
Essentials
Current cores
16
Dedicated nodes
3
Low-priority nodes
5
Operating System
microsoft-azure-batch ubuntu-server-container 16-04-lts (latest)
VM size
standard_d2_v2
Allocation state
Steady
Summary
Idle
0
Running
8
Creating
0
Starting
0
Rebooting
0

Cross-validation with caret

- Most predictive modeling algorithms have “tuning parameters”
- Example: Boosted Trees
  - Boosting iterations
  - Max Tree Depth
  - Shrinkage
- Parameters affect model performance
- Try ‘em out: **cross-validate**

```
grid <-
 data.frame(
  nrounds = …,
  max_depth = …,
  gamma = …,
  colsample_bytree = …,
  min_child_weight = …,
  subsample = …)
 )
```

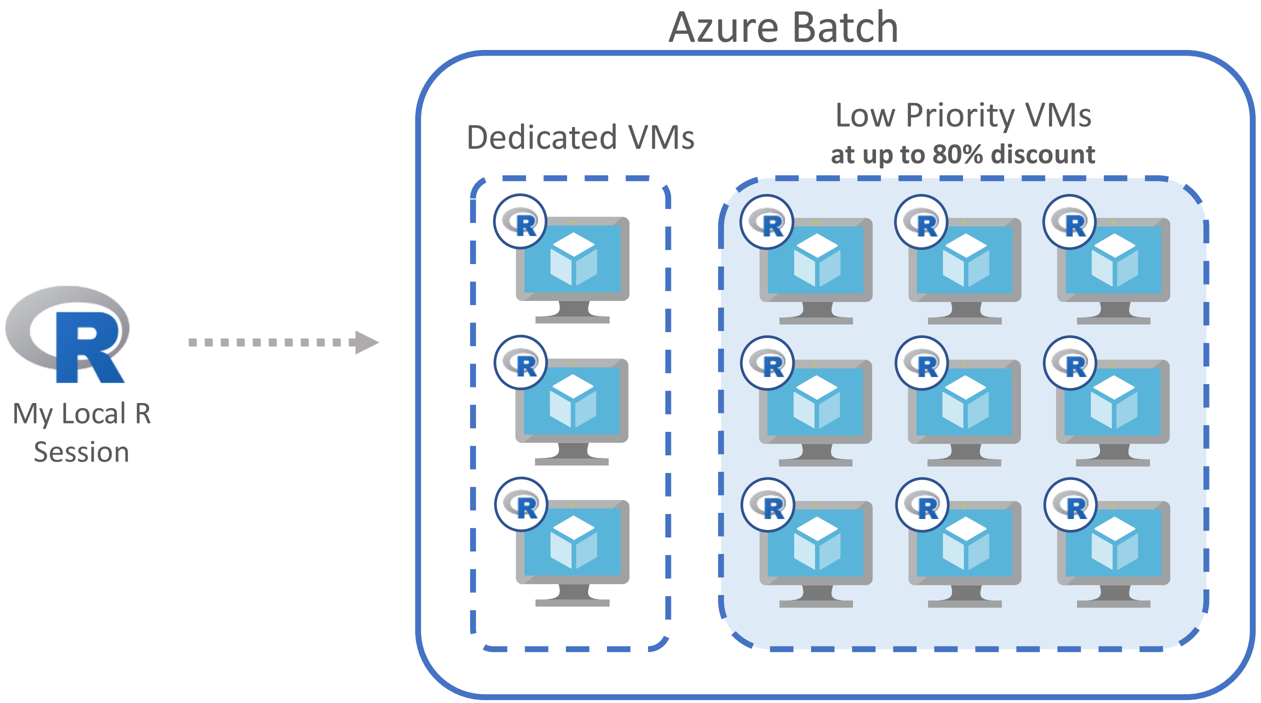
Cross-validation in parallel

- Caret’s train function will **automatically** use the registered foreach backend
- Just register your cluster first:
  `registerDoAzureParallel(cluster)`
- Handles sending objects, packages to nodes

```
mod <- train(
 Class ~ .,
 data = dat,
 method = "xgbTree",
 trControl = ctrl,
 tuneGrid = grid,
 nthread = 1
)
```

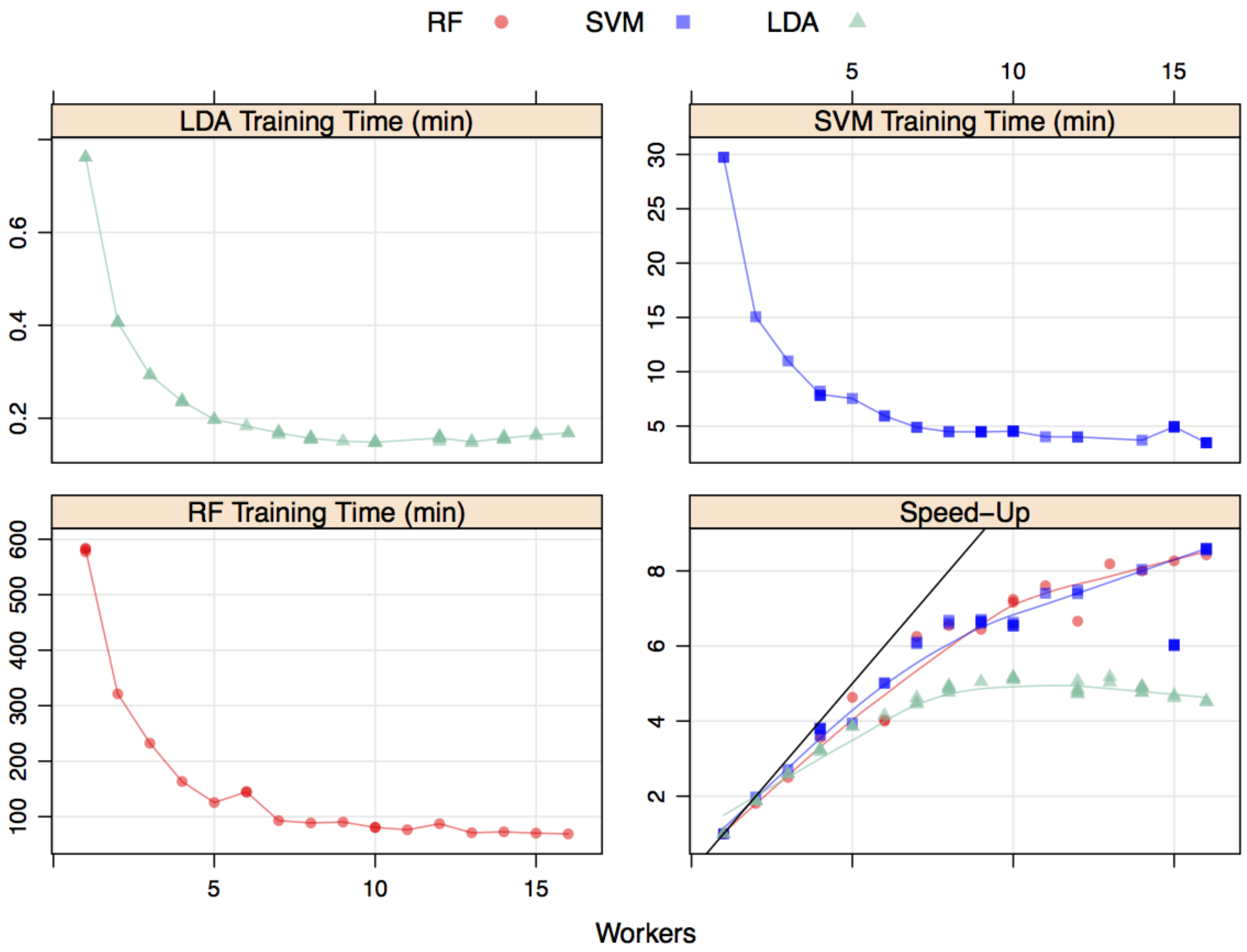
Low Priority Nodes

- Low-Priority = (very) Low Costs VMs from surplus capacity
  - up to 80% discount
- Clusters can mix dedicated VMs and low-priority VMs




```
"poolSize": {
    "dedicatedNodes": {
      "min": 3,
      "max": 3
    },
    "lowPriorityNodes": {
      "min": 9,
      "max": 9
    },
```

caret speed-ups



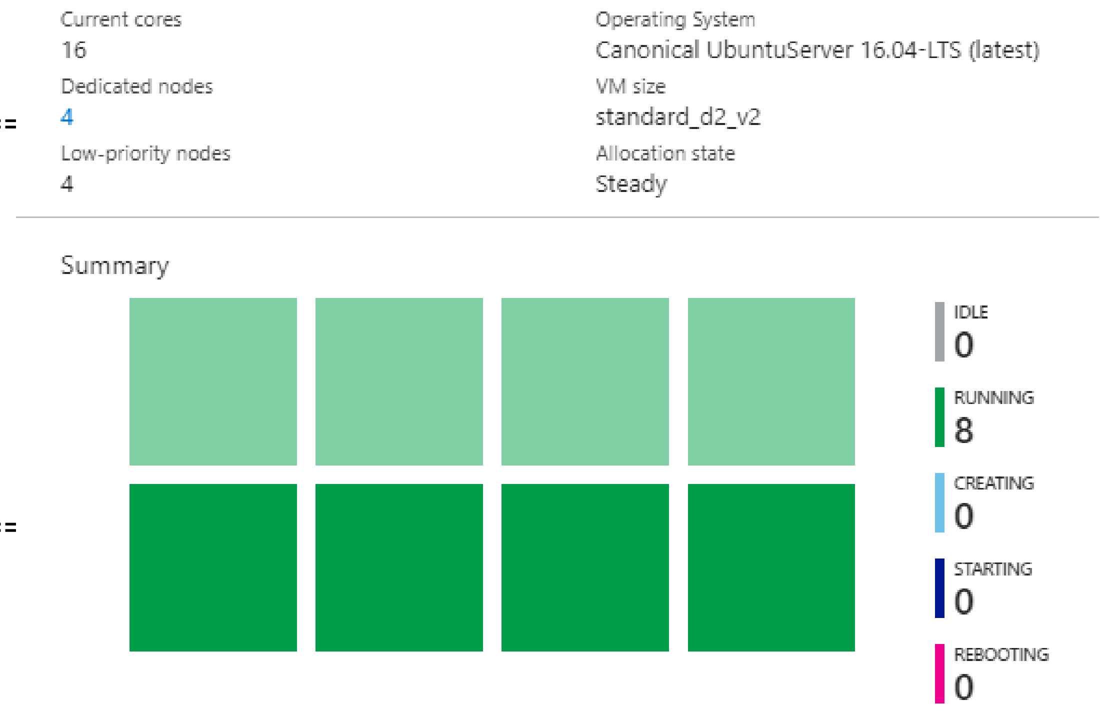
Source: `cda.ms/6V`

- Max Kuhn benchmarked various hardware and OS for local parallel
  - [`cda.ms/6V`](cda.ms/6V)
- Let's see how it works with doAzureParallel

Packages and Containers

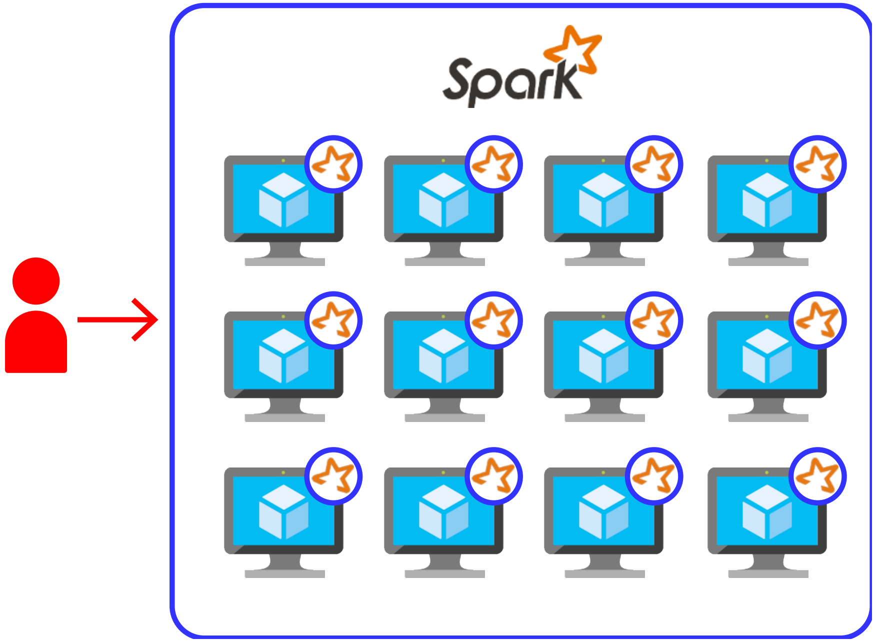
- Docker images used to spawn nodes
  - Default: `rocker/tidyverse:latest`
  - Lots of R packages pre-installed
- But this cross-validation also needs:
  - **xgboost**, **e1071**
- Easy fix: add to `cluster.json`

```
{
  "name": "davidsmi8caret",
  "vmSize": "Standard_D2_v2",
  "maxTasksPerNode": 8,
  "poolSize": {
    "dedicatedNodes": {
      "min": 4,
      "max": 4
    },
    "lowPriorityNodes": {
      "min": 4,
      "max": 4
    },
    "autoscaleFormula": "QUEUE"
  },
  "containerImage":
          "rocker/tidyverse:latest",
  "rPackages": {
    "cran": ["xgboost","e1071"],
    "github": [],
    "bioconductor": []
  },
  "commandLine": []
}
```




```
===============================================================
Id: job20180126022301
chunkSize: 1
enableCloudCombine: TRUE
packages:
         caret;
errorHandling: stop
wait: TRUE
autoDeleteJob: TRUE
===============================================================
Submitting tasks (1250/1250)
Submitting merge task. . .
Job Preparation Status: Package(s) being installed.......

Waiting for tasks to complete. . .
| Progress: 13.84% (173/1250) | Running: 59 | Queued: 1018 | Completed: 173 | Failed: 0 |
```

MY LAPTOP: 78 minutes
THIS CLUSTER: 16 minutes
(almost 5x faster)



For when it’s not complex parallel:

DISTRIBUTED DATA PROCESSING WITH SPARKLYR

What is Spark?

- Distributed data processing engine
    - Store and analyze massive volumes in a robust, scalable cluster
- Successor to Hadoop
    - in-memory engine 100x faster than map-reduce
- Highly extensible, with machine-learning capabilities
    - Supports Scala, Java, Python, R …
- Managed cloud services available
    - Azure Databricks & HDInsight, AWS EMR, GCP Dataproc
- Largest open-source data project
    - Apache project with 1000+ contributors

R and Spark: Sparklyr

- sparklyr: R interface to Spark
  - open-source R package from RStudio
- Move data between R and Spark
- “References” to Spark Data Frames
  - Familiar R operations, including dplyr syntax
  - Computations offloaded to Spark cluster, and deferred until needed
    - CPU/RAM/Disk consumed in cluster, not by R
- Interfaces to Spark ML algorithms

`spark.rstudio.com`

Provisioning clusters for Sparklyr

- **Azure:** Command-line interface to provision Spark-ready (and Sparklyr-ready) clusters in Azure Batch
  - www.github.com/azure/aztk
- **AWS**: Configure EMR cluster with software components
  - Guide: spark.rstudio.com/examples/yarn-cluster-emr
- **H20**: via the **rsparkling** package

dplyr with Sparklyr

Connect to the Spark cluster:

```
library(sparklyr)
cluster_url <- paste0("spark://", system("hostname -i", intern = TRUE), ":7077")
sc <- spark_connect(master = cluster_url)
```

Load in some data:

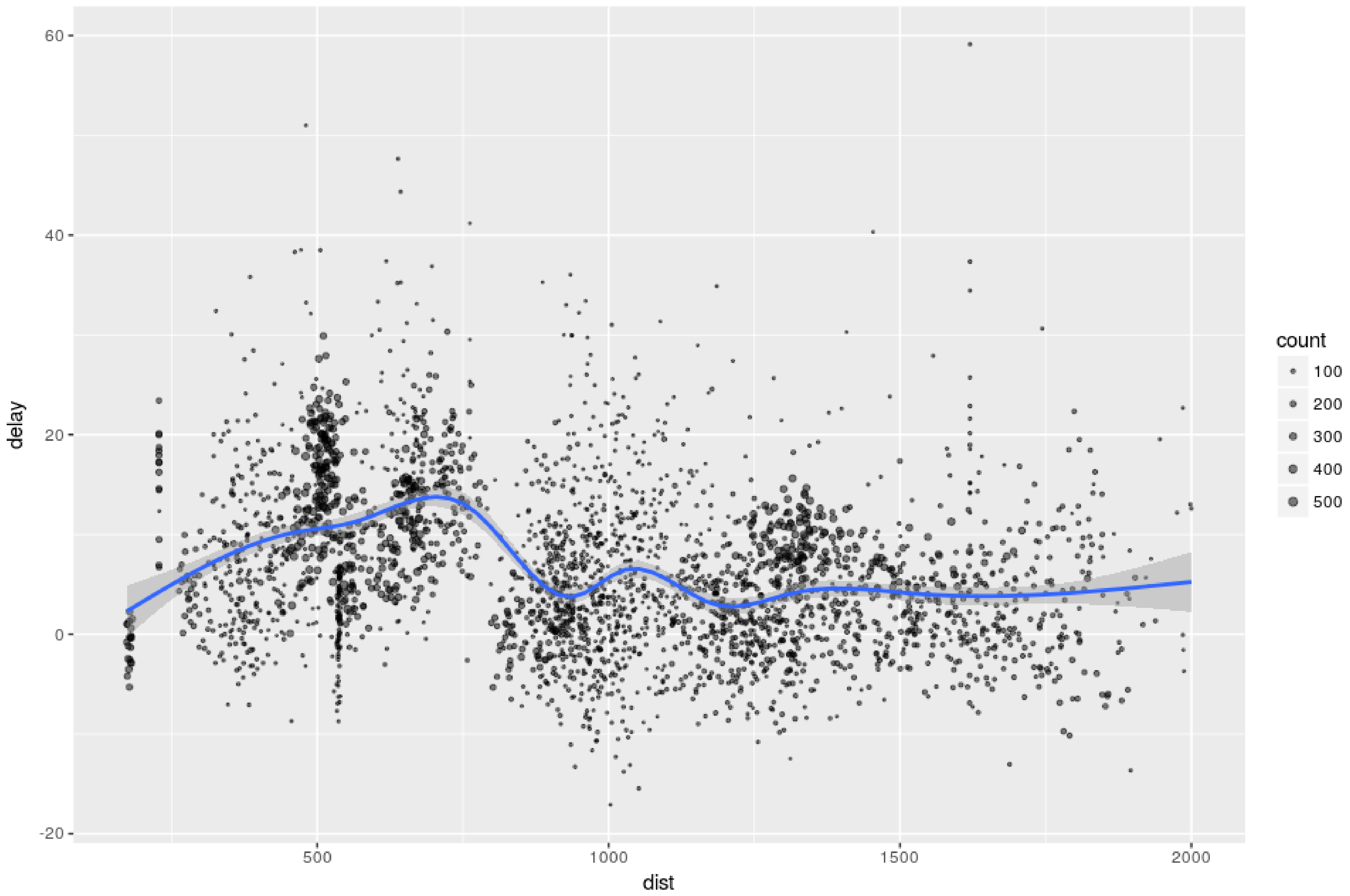
```
library(dplyr)
flights_tbl <- copy_to(sc, nycflights13::flights, "flights")
```

Munge with dplyr:

```
delay <- flights_tbl %>%
  group_by(tailnum) %>%
  summarise(count = n(), dist = mean(distance), delay = mean(arr_delay)) %>%
  filter(count > 20, dist < 2000, !is.na(delay)) %>%
  collect
```

Things to note

- All of the computation take place in the Spark cluster
  - Computations are delayed until you need results
  - Behind the scenes, Spark SQL statements are being written for you
- None of the data comes back to R
  - Until you call `collect`, when it becomes a `tbl`
  - It’s only at this point you have to worry about data size
- This is all ordinary `dplyr` syntax



60
40
20
0
-20
delay
500
1000
1500
2000
dist
count
100
200
300
400
500

Machine Learning with SparklyR

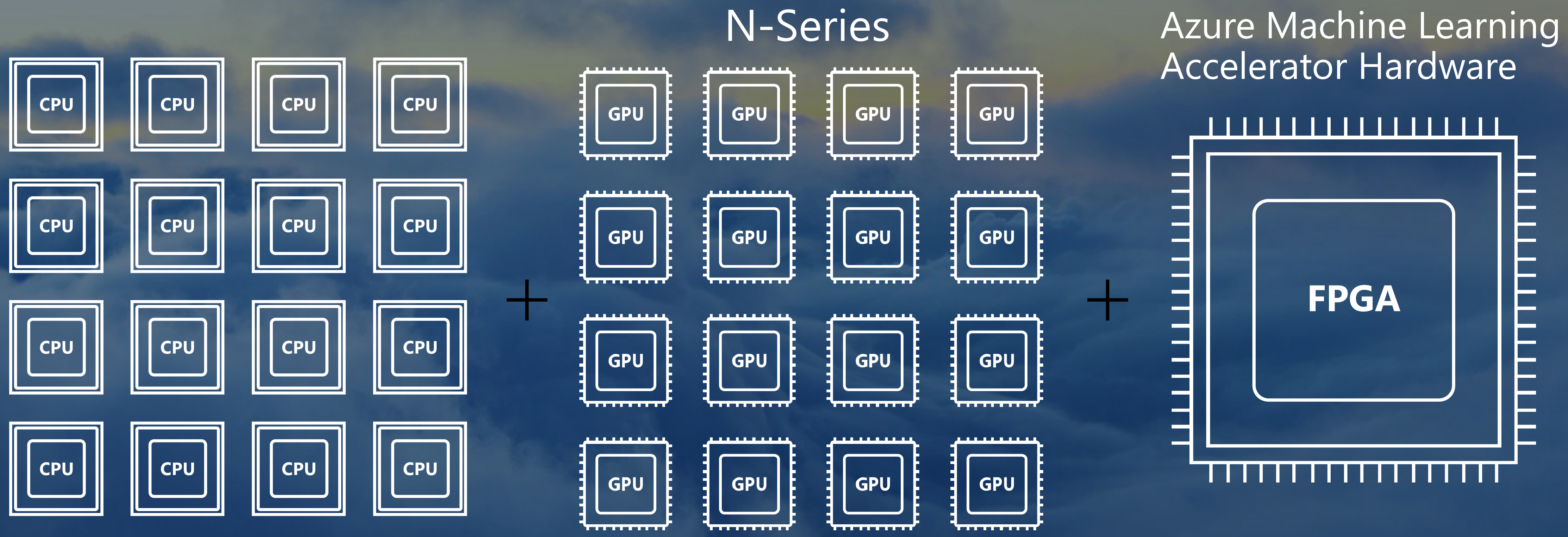
SparklyR provides R interfaces to Spark's distributed machine learning algorithms (MLlib)

Computations happing in the Spark cluster, not in R

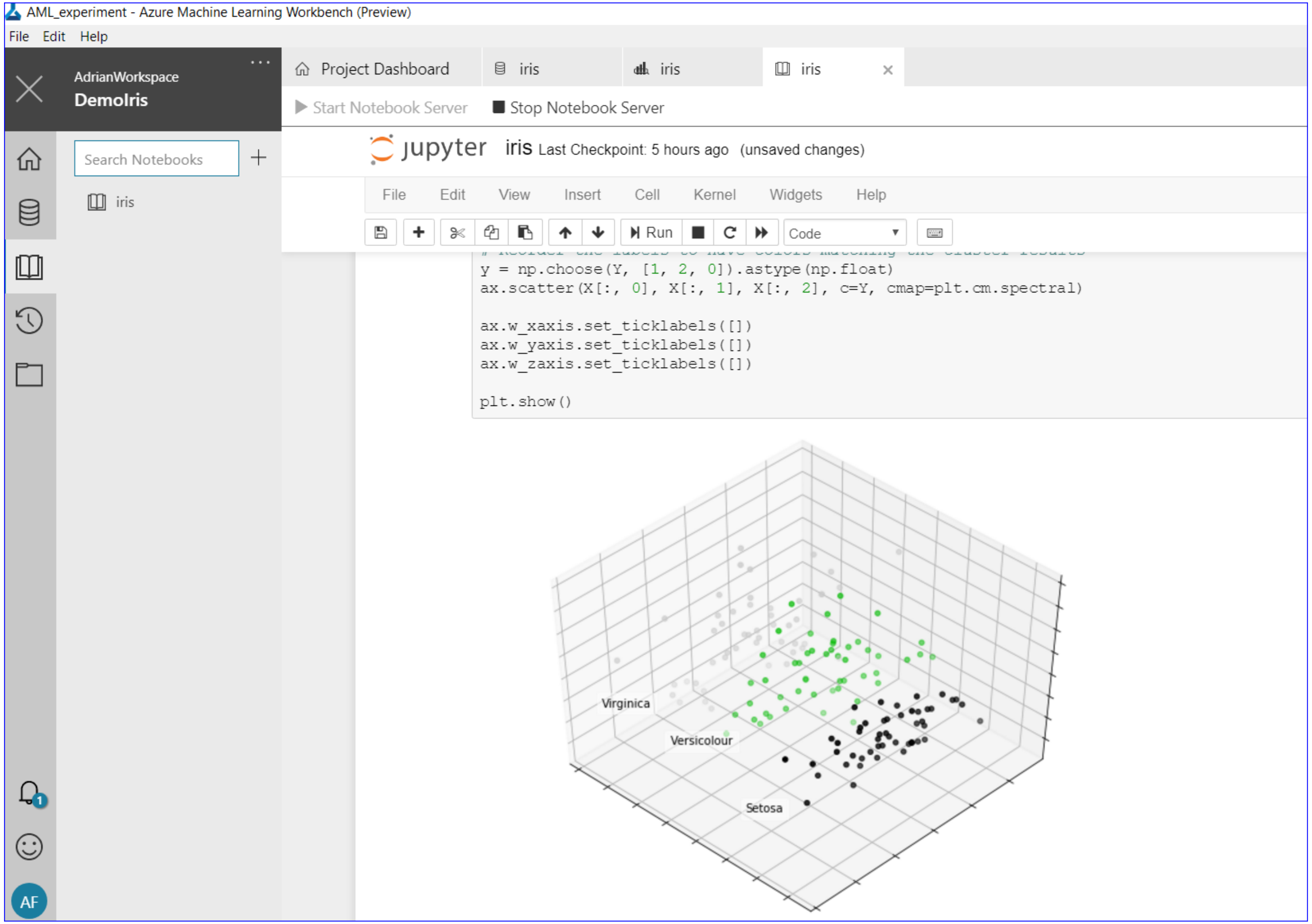
```
> m <- ml_linear_regression(delay ~ dist, data=delay_near)
* No rows dropped by 'na.omit' call
> summary(m)
Call: ml_linear_regression(delay ~ dist, data = delay_near)

Deviance Residuals::
     Min       1Q   Median       3Q      Max
-19.9499  -5.8752  -0.7035   5.1867  40.8973

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.6904319  1.0199146   0.677   0.4986
dist        0.0195910  0.0019252  10.176   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-Squared: 0.09619
Root Mean Squared Error: 8.075
>
```

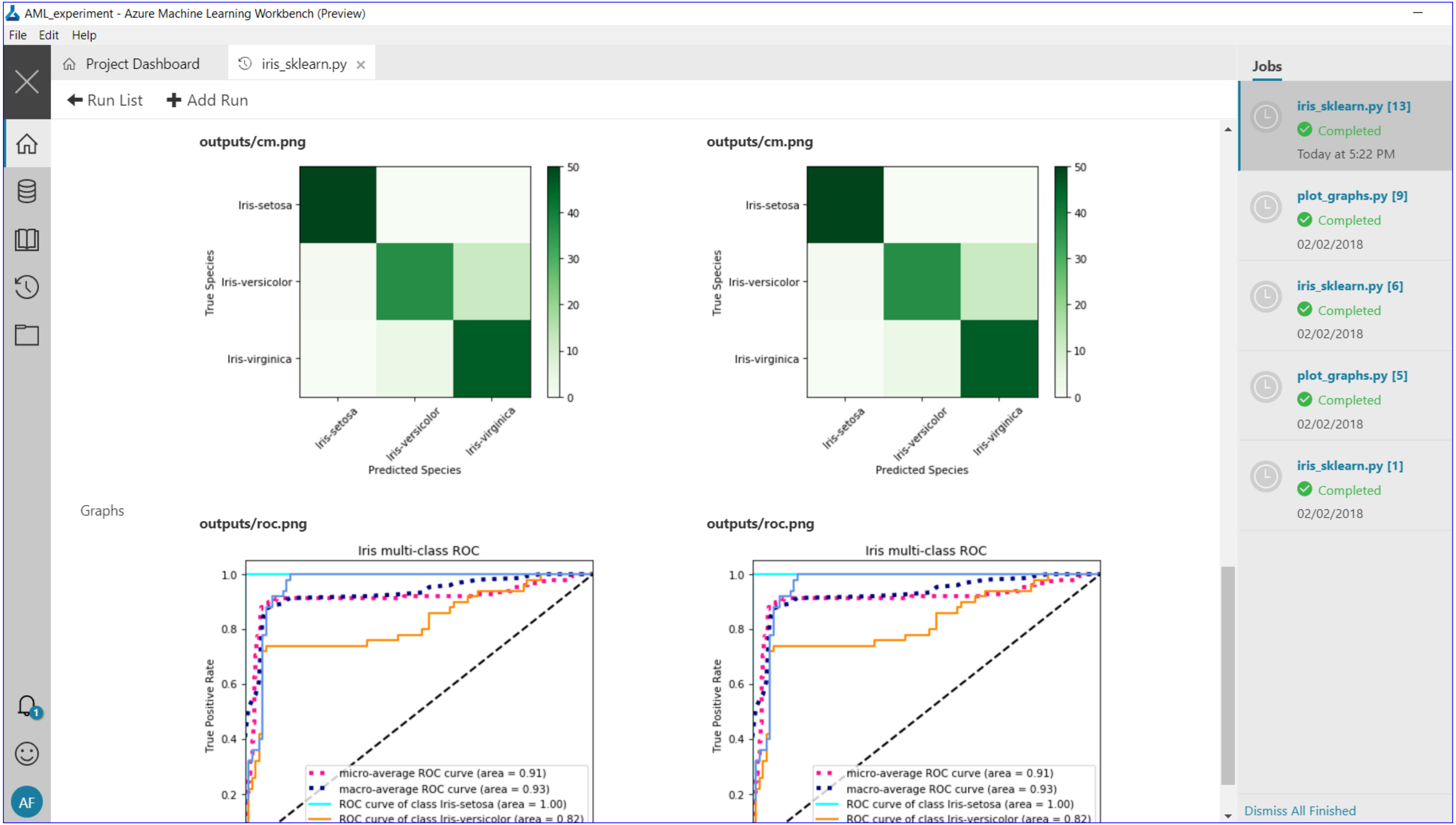
Azure

AI Supercomputer

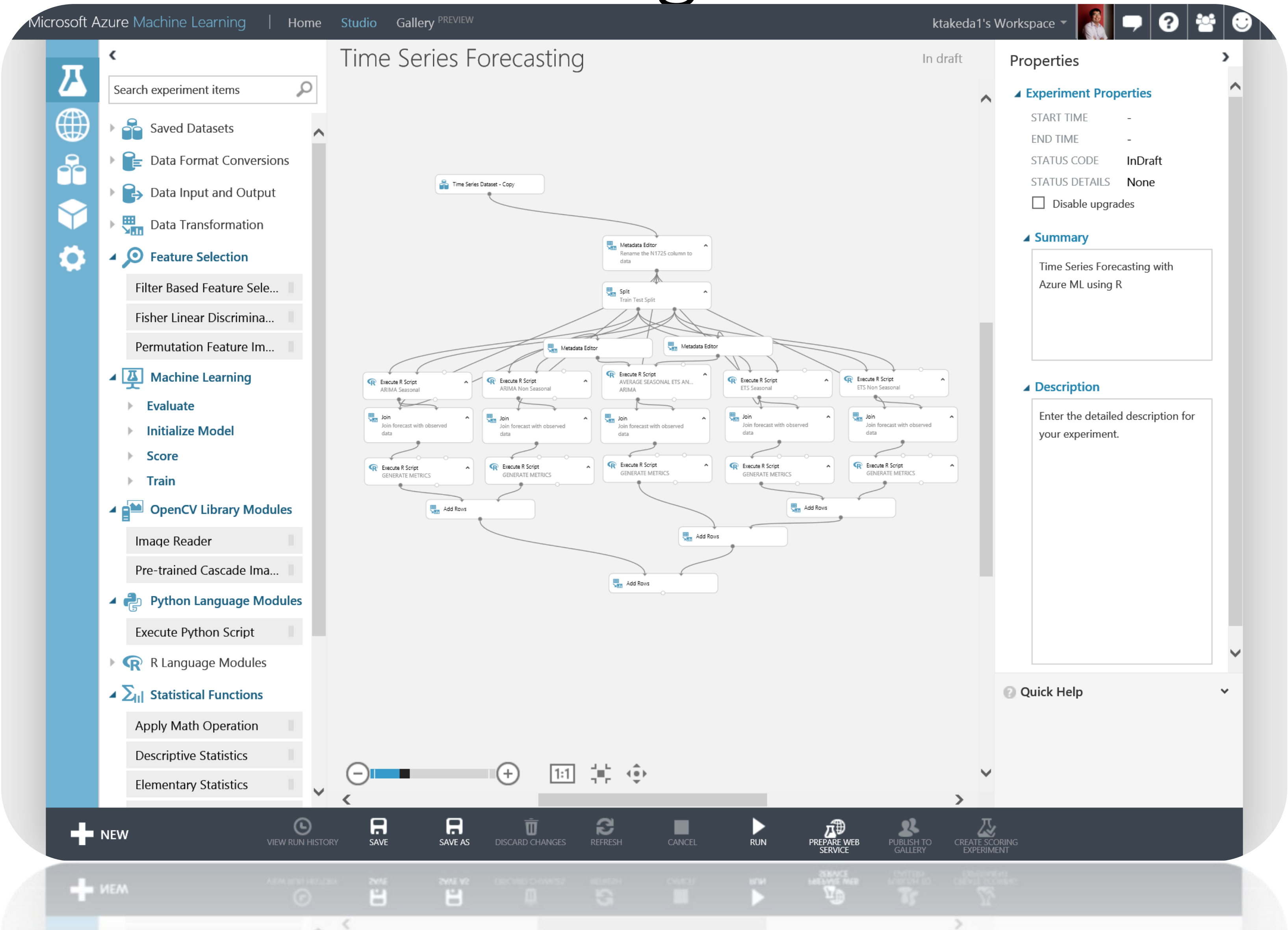


Azure Machine Learning Workbench

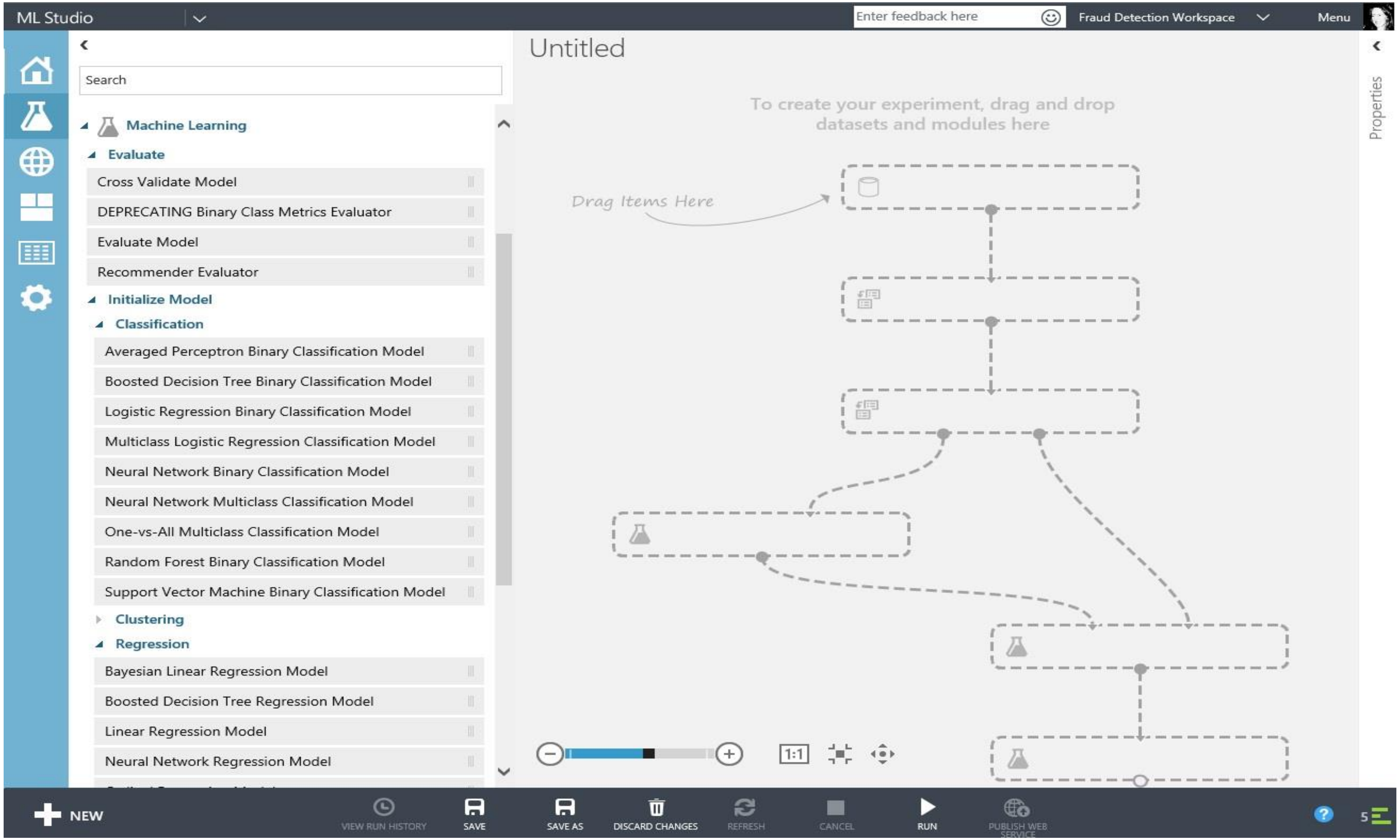



AML_experiment - Azure Machine Learning Workbench (Preview)
File
Edit
Help
Project Dashboard
iris_sklearn.py
Run List
Add Run
outputs/cm.png
Iris-setosa
Iris-versicolor
Iris-virginica
True Species
Predicted Species
outputs/cm.png
Graphs
outputs/roc.png
Iris multi-class ROC
True Positive Rate
micro-average ROC curve (area = 0.91)
macro-average ROC curve (area = 0.93)
ROC curve of class Iris-setosa (area = 1.00)
ROC curve of class Iris-versicolor (area = 0.82)
outputs/roc.png
Jobs
iris_sklearn.py [13]
Completed
Today at 5:22 PM
plot_graphs.py [9]
Completed
02/02/2018
iris_sklearn.py [6]
Completed
02/02/2018
plot_graphs.py [5]
Completed
02/02/2018
iris_sklearn.py [1]
Completed
02/02/2018
Dismiss All Finished
AF

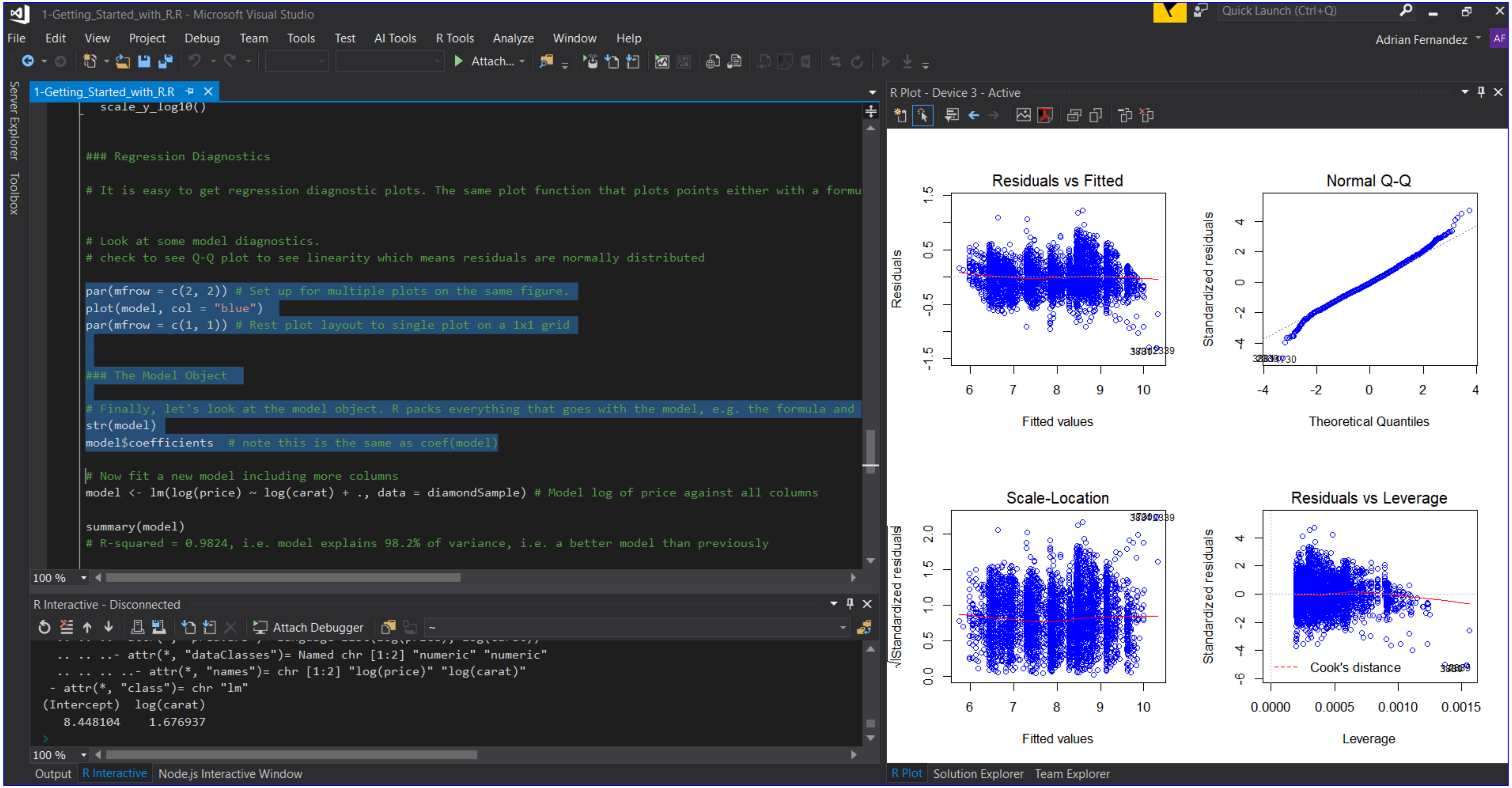
Azure Machine Learning Studio

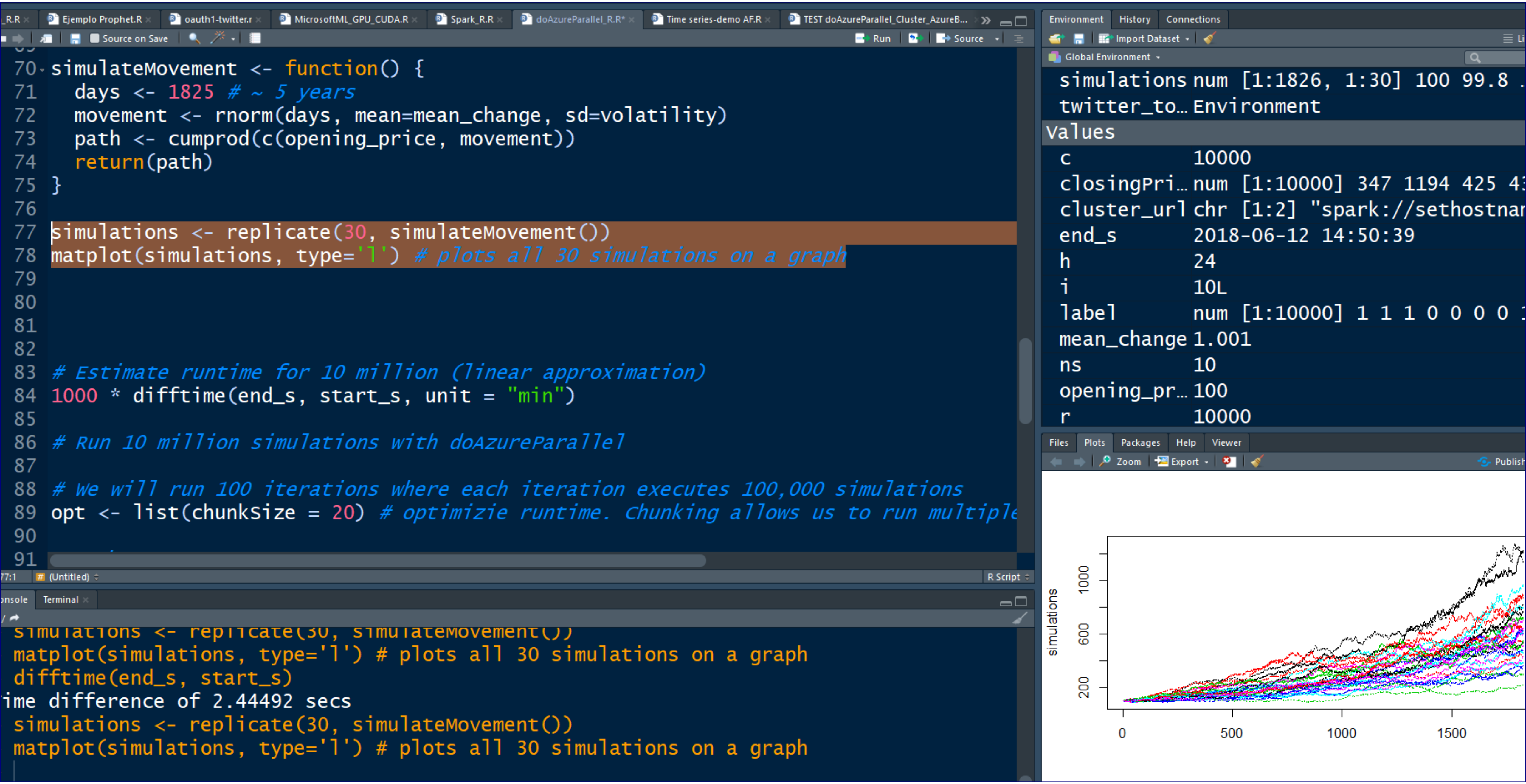


Drag & Drop + Los mejores Algoritmos ML

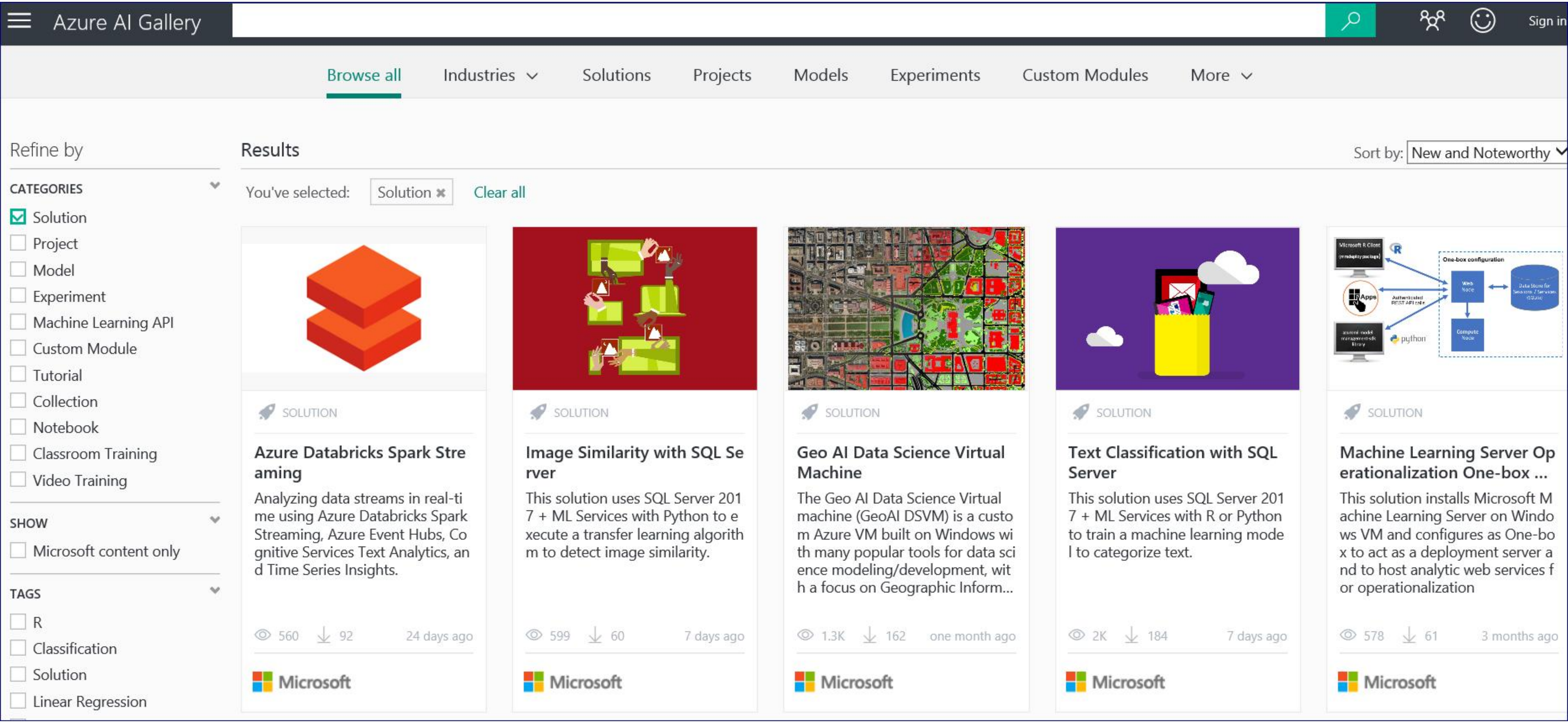


Visual Studio 2017 con R Tools

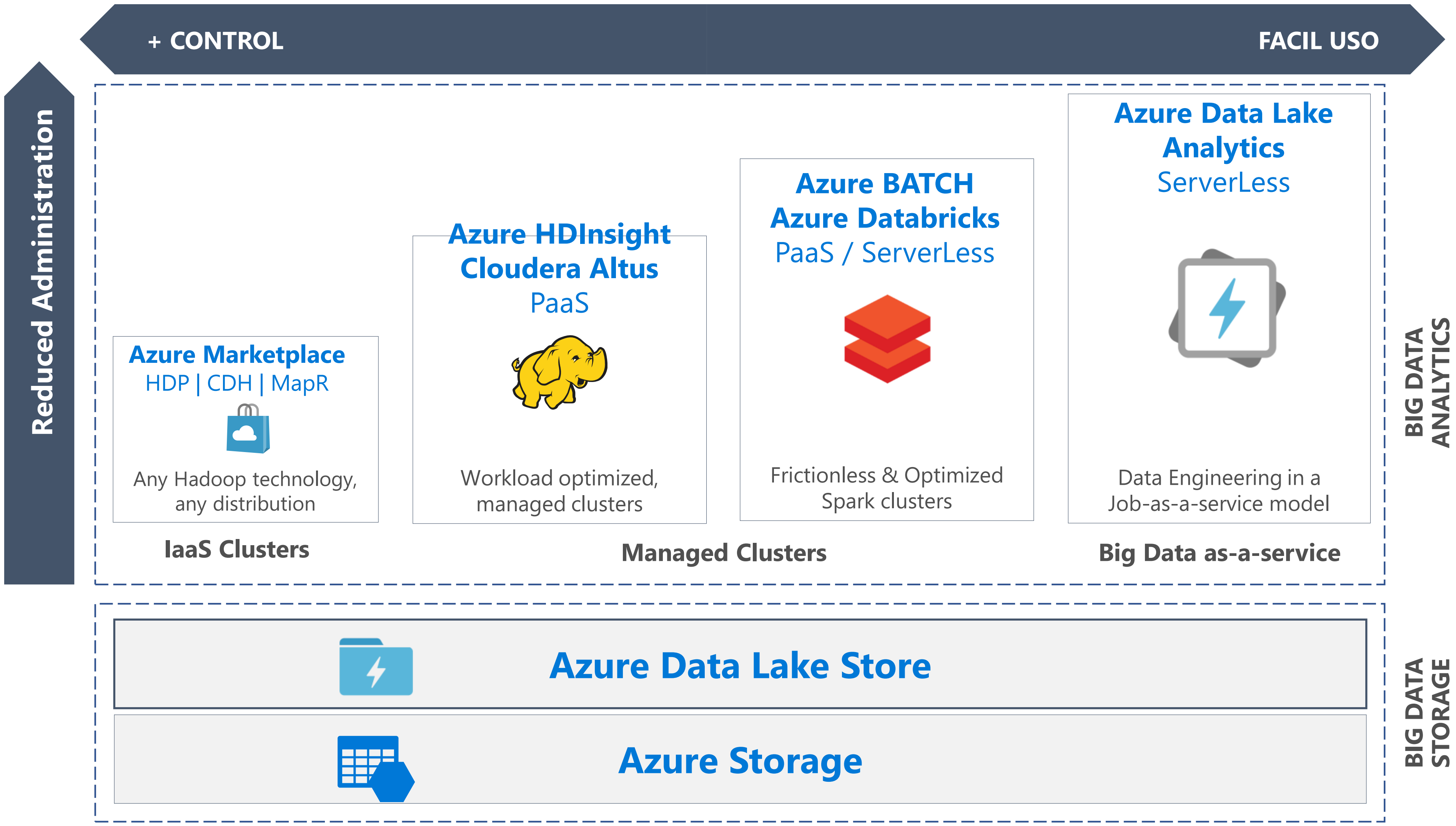




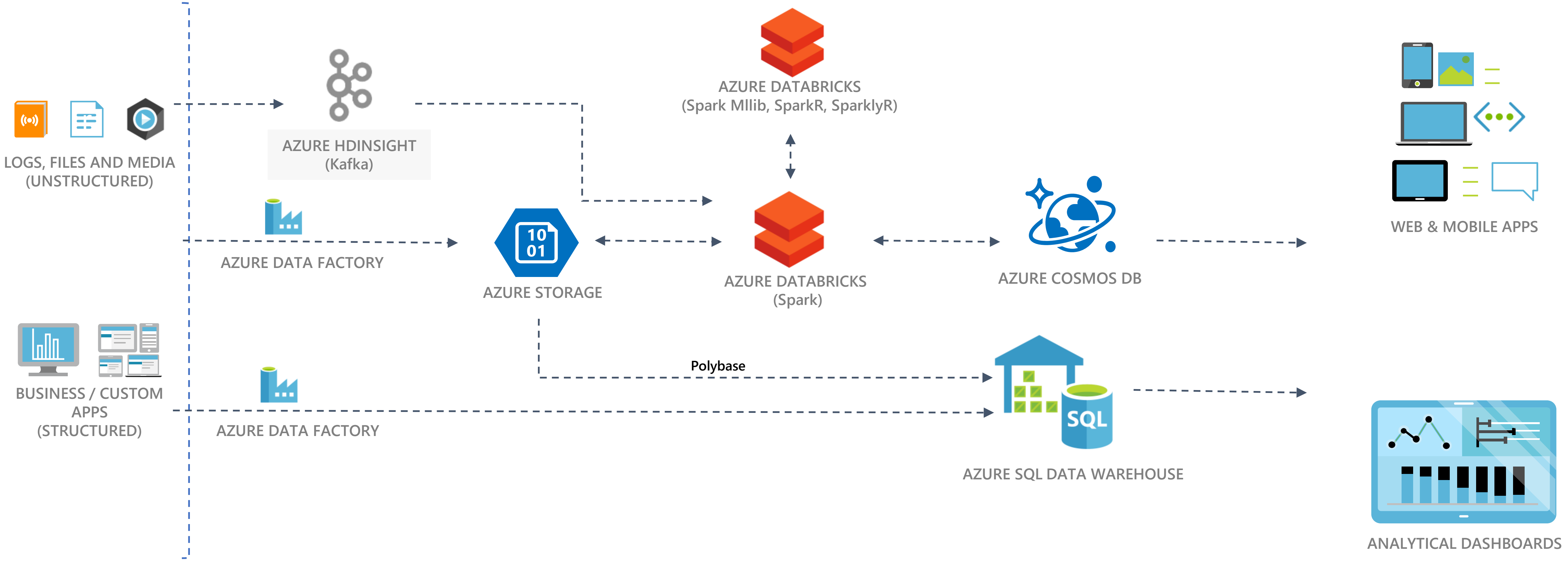
.R.R
Ejemplo Prophet.R
oauth1-twitter.r
MicrosoftML_GPU_CUDA.R
Spark_R.R
doAzureParallel_R.R*
Time series-demo AF.R
TEST doAzureParallel_Cluster_AzureB...
Source on Save
Run
Source
simulateMovement <- function() {
  days <- 1825 # ~ 5 years
  movement <- rnorm(days, mean=mean_change, sd=volatility)
  path <- cumprod(c(opening_price, movement))
  return(path)
}
simulations <- replicate(30, simulateMovement())
matplot(simulations, type='l') # plots all 30 simulations on a graph
Estimate runtime for 10 million (linear approximation)
1000 * difftime(end_s, start_s, unit = "min")
# Run 10 million simulations with doAzureParallel
# We will run 100 iterations where each iteration executes 100,000 simulations
opt <- list(chunkSize = 20) # optimizie runtime. Chunking allows us to run multiple
77:1
(Untitled)
R Script
Console
Terminal
simulations <- replicate(30, simulateMovement())
matplot(simulations, type='l') # plots all 30 simulations on a graph
difftime(end_s, start_s)
Time difference of 2.44492 secs
simulations <- replicate(30, simulateMovement())
matplot(simulations, type='l') # plots all 30 simulations on a graph
Environment
History
Connections
Import Dataset
Global Environment
simulations num [1:1826, 1:30] 100 99.8
twitter_to... Environment
Values
c 10000
closingPri... num [1:10000] 347 1194 425
cluster_url chr [1:2] "spark://sethostna
end_s 2018-06-12 14:50:39
h 24
i 10L
label num [1:10000] 1 1 1 0 0 0 0
mean_change 1.001
ns 10
opening_pr... 100
r 10000
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
simulations
200
600
1000
0
500
1000
1500


Azure AI Gallery
Sign in
Browse all
Industries
Solutions
Projects
Models
Experiments
Custom Modules
More
Refine by
CATEGORIES
Solution
Project
Model
Experiment
Machine Learning API
Custom Module
Tutorial
Collection
Notebook
Classroom Training
Video Training
SHOW
Microsoft content only
TAGS
R
Classification
Solution
Linear Regression
Results
Sort by: New and Noteworthy
You've selected: Solution
Clear all
SOLUTION
Azure Databricks Spark Streaming
Analyzing data streams in real-time using Azure Databricks Spark Streaming, Azure Event Hubs, Cognitive Services Text Analytics, and Time Series Insights.
560 92 24 days ago
Microsoft
SOLUTION
Image Similarity with SQL Server
This solution uses SQL Server 2017 + ML Services with Python to execute a transfer learning algorithm to detect image similarity.
599 60 7 days ago
Microsoft
SOLUTION
Geo AI Data Science Virtual Machine
The Geo AI Data Science Virtual machine (GeoAI DSVM) is a custom Azure VM built on Windows with many popular tools for data science modeling/development, with a focus on Geographic Inform...
1.3K 162 one month ago
Microsoft
SOLUTION
Text Classification with SQL Server
This solution uses SQL Server 2017 + ML Services with R or Python to train a machine learning model to categorize text.
2K 184 7 days ago
Microsoft
SOLUTION
Machine Learning Server Operationalization One-box ...
This solution installs Microsoft Machine Learning Server on Windows VM and configures as One-box to act as a deployment server and to host analytic web services for operationalization
578 61 3 months ago
Microsoft

TODAS LAS SOLUCIONES BIG DATA EN AZURE



Big Data Lambda Architecture



In summary

- Embarrassingly parallel (small): foreach + local backend
- Embarrassingly parallel (big): foreach + cluster/multicore backend
- distributed data, not embarrassingly parallel: Spark + sparklyr
- Databricks, cost effective Spark
- GPU + FPGA


Adrián J. Fernández
Technology Solutions Professional
Advanced Analytics & Artificial Intelligence
Adrian.Fernandez@microsoft.com
Microsoft Chile
Av. Vitacura 6844, Vitacura, Santiago, Chile
Microsoft