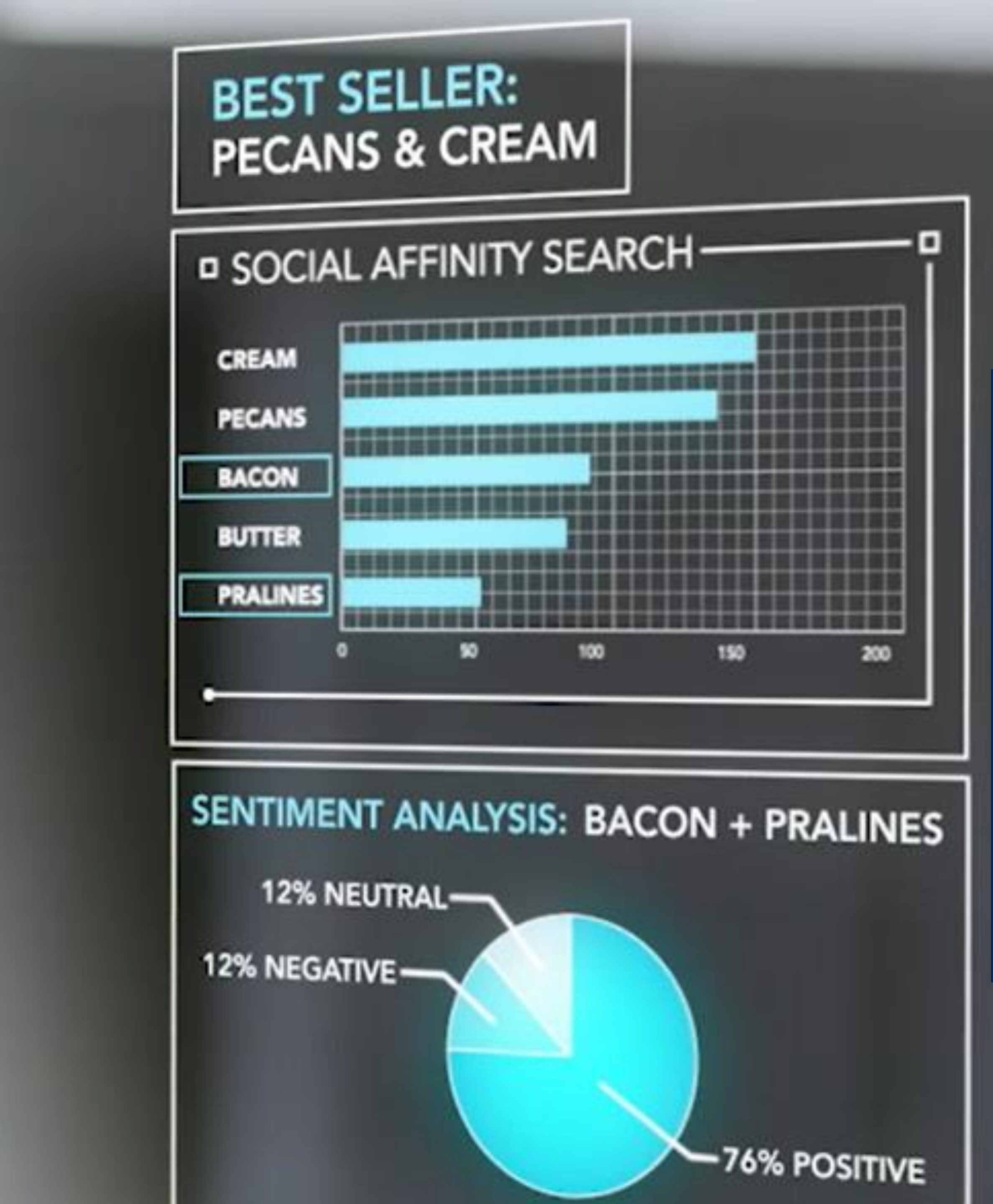
Agenda – Day 2

Who	When	What	How
Instructors	09:00 – 09:30	R Deployment options	Chalk & Talk
You	09:30 – 11:00	Lab 05: Operationalizing R with Azure Machine Learning	Lab
You	11:00 – 12:30	Lab 08: SQL Server R Services	Lab
All	12:30 – 13:30	< LUNCH >	
You	13:30 – 14:00	Microsoft R Server on Hadoop	Presentation
You	14:00 – 16:00	Lab 07 : Getting started with MRS on HDInsight (Spark)	Lab
All	16:00 – 16:30	Wrap up: Questions and Answers	Discussion



R Deployment

(web services)

Microsoft

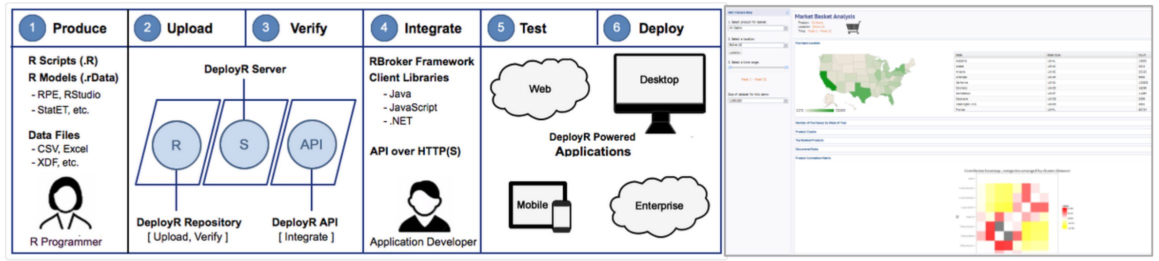
R is a great **modelling** tool, but

How do we operationalize R?

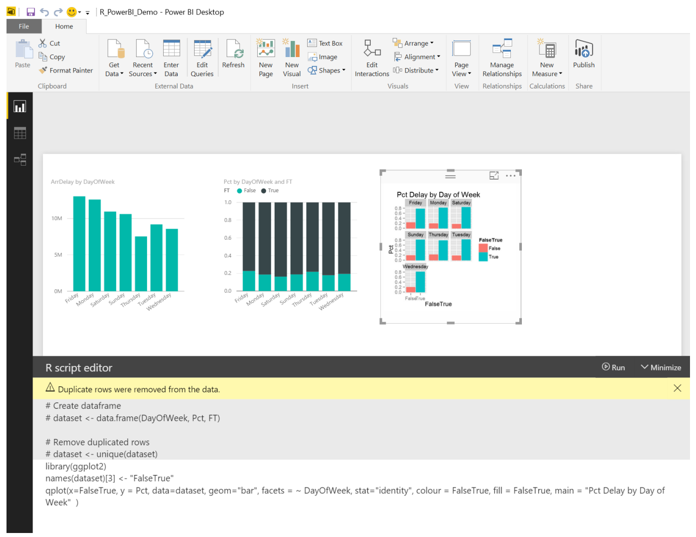
Deployment Acceleration

Microsoft R Server Operationalization

Basic Workflow



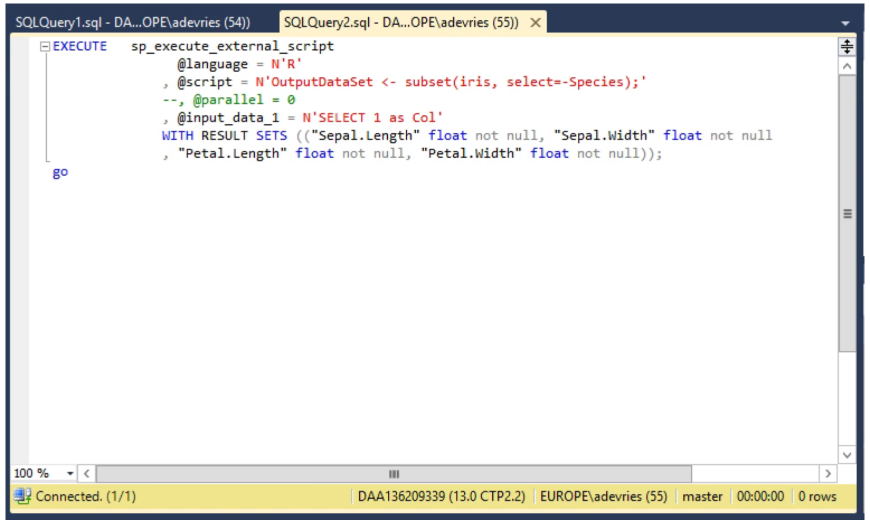
Deploy in PowerBI – R Integration



Duplicate rows were removed from the data.

```
# Create dataframe
# dataset <- data.frame(DayOfWeek, Pct, FT)

# Remove duplicated rows
# dataset <- unique(dataset)
library(ggplot2)
names(dataset)[3] <- "FalseTrue"
qplot(x=FalseTrue, y = Pct, data=dataset, geom="bar", facets = ~ DayOfWeek, stat="identity", colour = FalseTrue, fill = FalseTrue, main = "Pct Delay by Day of
Week"  )
```

Deploy in SQL Server Stored Procedure


SQLQuery1.sql - DA...OPE\adevries (54))
SQLQuery2.sql - DA...OPE\adevries (55))
100 %

```
EXECUTE   sp_execute_external_script
            @language = N'R'
          , @script = N'OutputDataSet <- subset(iris, select=-Species);'
          --, @parallel = 0
          , @input_data_1 = N'SELECT 1 as Col'
          WITH RESULT SETS (("Sepal.Length" float not null, "Sepal.Width" float not null
          , "Petal.Length" float not null, "Petal.Width" float not null));
go
```

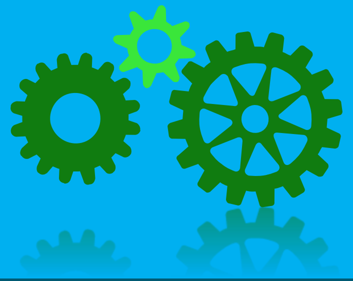
Connected. (1/1) | DAA136209339 (13.0 CTP2.2) | EUROPE\adevries (55) | master | 00:00:00 | 0 rows

Deploy to Azure (Cloud)

AzureML R package

```
api <- publishWebService(
  ws,
  fun = add,
  name = "aalab-silly",
  inputSchema = list(
    x = "numeric",
    y = "numeric"
  ),
  outputSchema = list(
    ans = "numeric"
  )
)

api
```

Microsoft R Server
The Operationalization Engine of your Advanced Analytics

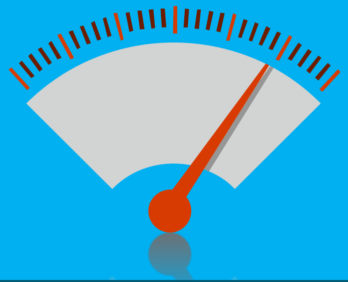


Instant Deployment

- Turn R analytics → Web services in one line of code;
- Swagger-based REST APIs, easy to consume, with any programming languages, including R!

Deploy to Anywhere

- Deploying web service server to any platform: Windows, SQL, Linux/Hadoop
- On-prem or in cloud



Fast and Scalable

- Fast scoring, real time and batch
- Scaling to a grid for powerful computing with load balancing
- Diagnostic and capacity evaluation tools

Secure and Reliable

- Enterprise authentication: AD/LDAP or AAD
- Secure connection: HTTPS with SSL/TLS 1.2
- Enterprise grade high availability

Microsoft R Server
The Operationalization Engine of your Advanced Analytics

Instant Deployment

- Turn R analytics → Web Service in one line of code;
- Swagger-based REST APIs, easy to consume, with any programming languages, including R!

Deploy to Anywhere

- Deploying Web Service server to any platform: Windows / SQL / Linux/Hadoop
- On Prem or in Cloud

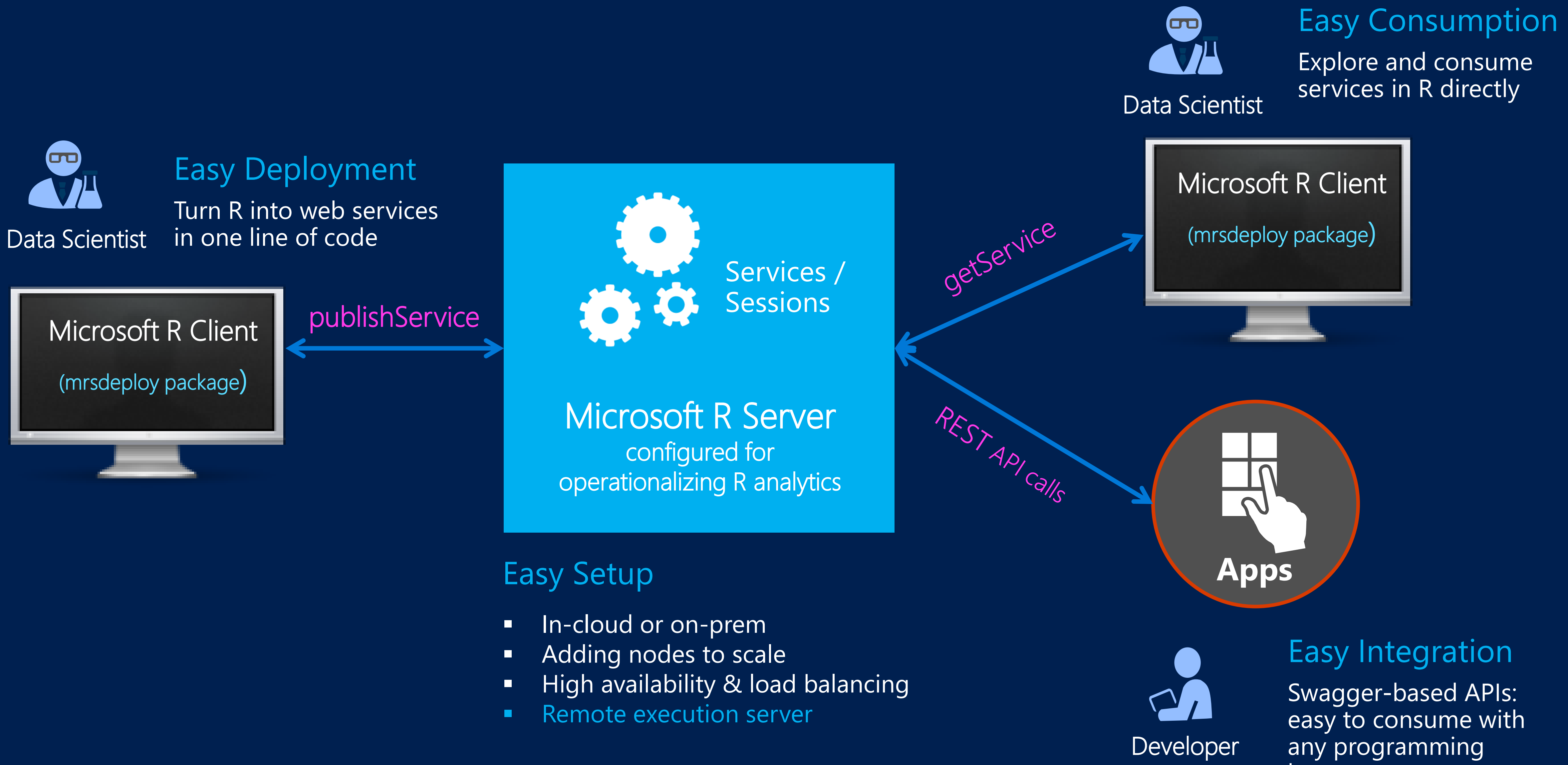
Fast and Scalable

- Fast scoring, real time and batch
- Scaling to a grid for powerful computing with load balancing
- Diagnostic and capacity evaluation tools

Secure and Reliable

- Enterprise authentication: LDAP / AD/ AAD
- Secure connection: HTTPS with SSL.TSL1.2
- Enterprise grade High Availability

Best-in-class Deployment Experience



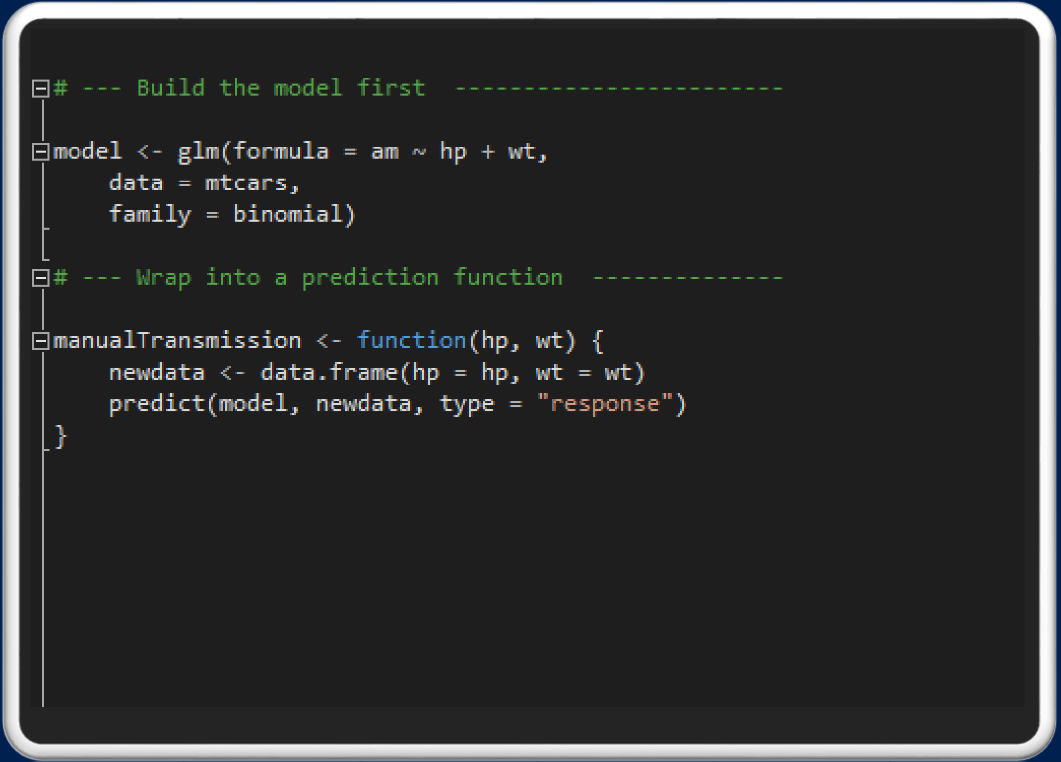
Easy Setup

- In-cloud or on-prem
- Adding nodes to scale
- High availability & load balancing
- Remote execution server

Easy Deployment

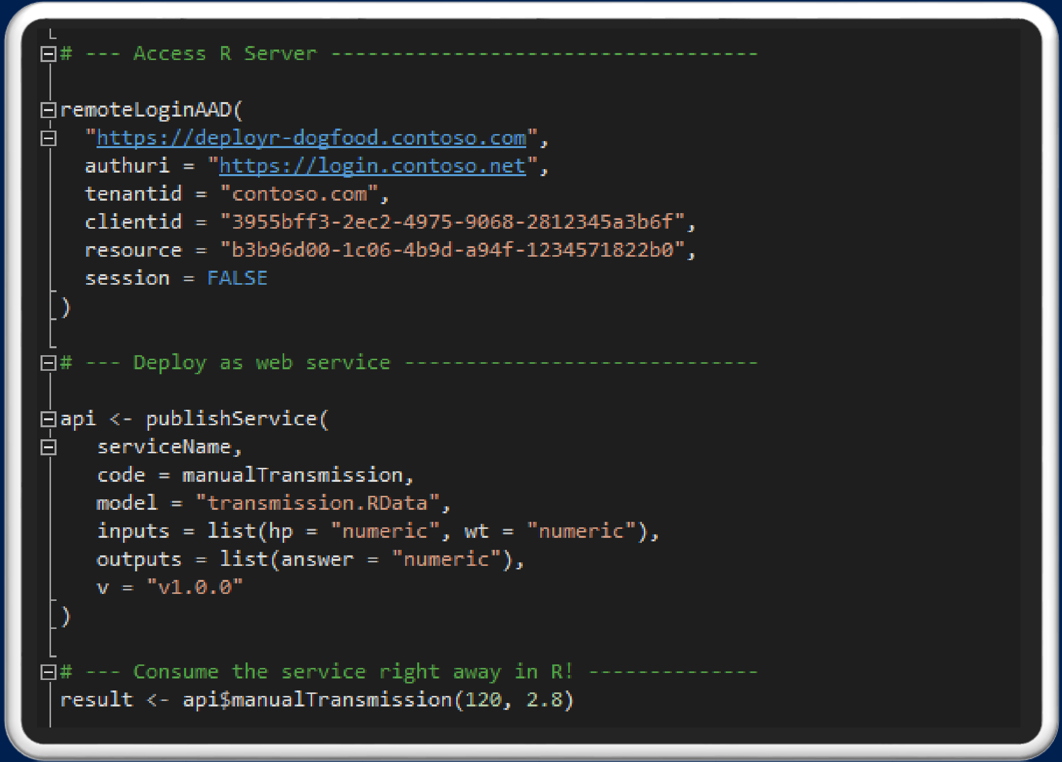
Turn R into Web Services in one line of code in R; and even consume them in R!

Build the model first



```r
# --- Build the model first  ------------------------

model <- glm(formula = am ~ hp + wt,
    data = mtcars,
    family = binomial)

# --- Wrap into a prediction function  --------------

manualTransmission <- function(hp, wt) {
    newdata <- data.frame(hp = hp, wt = wt)
    predict(model, newdata, type = "response")
}
```

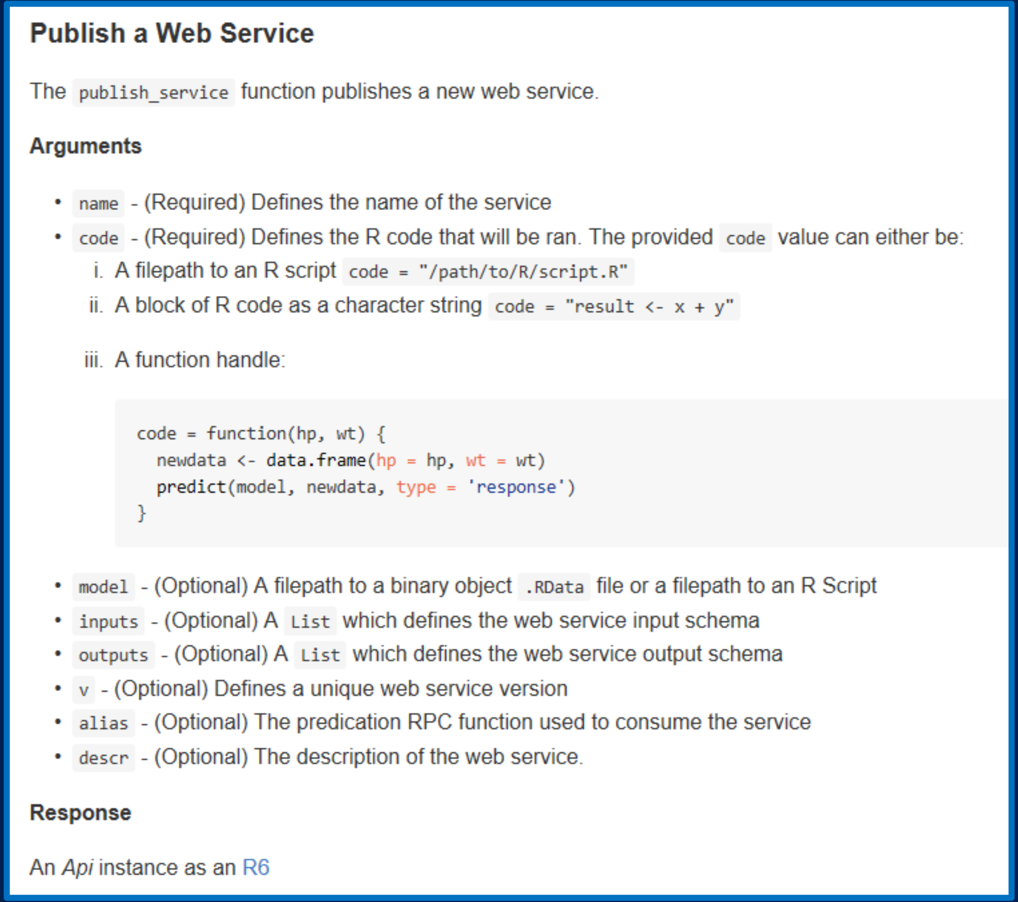
Deploy as a web service instantly



```r
# --- Access R Server ---------------------------------

remoteLoginAAD(
  "https://deployr-dogfood.contoso.com",
  authuri = "https://login.contoso.net",
  tenantid = "contoso.com",
  clientid = "3955bff3-2ec2-4975-9068-2812345a3b6f",
  resource = "b3b96d00-1c06-4b9d-a94f-1234571822b0",
  session = FALSE
)

# --- Deploy as web service ----------------------------

api <- publishService(
   serviceName,
   code = manualTransmission,
   model = "transmission.RData",
   inputs = list(hp = "numeric", wt = "numeric"),
   outputs = list(answer = "numeric"),
   v = "v1.0.0"
)

# --- Consume the service right away in R! -------------
result <- api$manualTransmission(120, 2.8)
```

Web Service Functions Cheat Sheet

Function	Description
publishService	Publish a predictive function as a Web Service
deleteService	Delete a Web Service
getService	Get a Web Service
ListServices	List the different published web services
serviceOption	Retrieve, set, and list the different service options
updateService	Updates a Web Service



Publish a Web Service

The `publish_service` function publishes a new web service.

Arguments

- `name` - (Required) Defines the name of the service
- `code` - (Required) Defines the R code that will be ran. The provided `code` value can either be:
  i. A filepath to an R script `code = "/path/to/R/script.R"`
  ii. A block of R code as a character string `code = "result <- x + y"`
  iii. A function handle:

```
code = function(hp, wt) {
  newdata <- data.frame(hp = hp, wt = wt)
  predict(model, newdata, type = 'response')
}
```

- `model` - (Optional) A filepath to a binary object `.RData` file or a filepath to an R Script
- `inputs` - (Optional) A `List` which defines the web service input schema
- `outputs` - (Optional) A `List` which defines the web service output schema
- `v` - (Optional) Defines a unique web service version
- `alias` - (Optional) The predication RPC function used to consume the service
- `descr` - (Optional) The description of the web service.

Response

An *Api* instance as an R6

Integration with Apps

Swagger based APIs, easy to consume, with any programming language

Data Scientist

Developer

Developer

Generate Swagger Docs for Web Services → **Run Swagger tools to generate code** → **Write a few code to consume the service**

```
# Run the following code in R

swagger <- api$swagger()

cat(swagger, file = "swagger.json",
append = FALSE)
```

Popular Swagger Tools:
AutoRest or Code Generator

```
AutoRest.exe -CodeGenerator
CSharp -Modeler Swagger -
Input swagger.json -
Namespace Mynamespace
```

```
using System;
using MyNamespace;
using MyNamespace.Models;

namespace TransmissionApiExample
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var api = new Transmission(new Uri("https://rservertest.com"));
            var accessToken = "{{YOUR_JWT_TOEKN}}";

            var headers = client.HttpClient.DefaultRequestHeaders;
            headers.Remove("Authorization");
            headers.Add("Authorization", $"Bearer {accessToken}");

            InputParameters inputs = new InputParameters() { hp = 120, wt = 2.8 };
            var serviceResult = api.Manual.TransmissionAsync(inputs).Result;

            Console.Out.WriteLine(serviceResult.OutputParameters);
        }
    }
}
```

Easy Consumption of web services in R

Enabling exciting new scenarios for data scientists

Enable **Model Management** capabilities

- A Predictive Web Service = “Model” + “Prediction Script”
- R Server hosts all those services → Central Repo of Models
- Each service has a version tag → Model Version Control
- All versions are active → Model Roll Back (to any version)
- A service can be accessed by any authorized users →
    - Model reuse
    - Model validation and monitoring by QA team

After service is published, I can test if the service works as expected right away

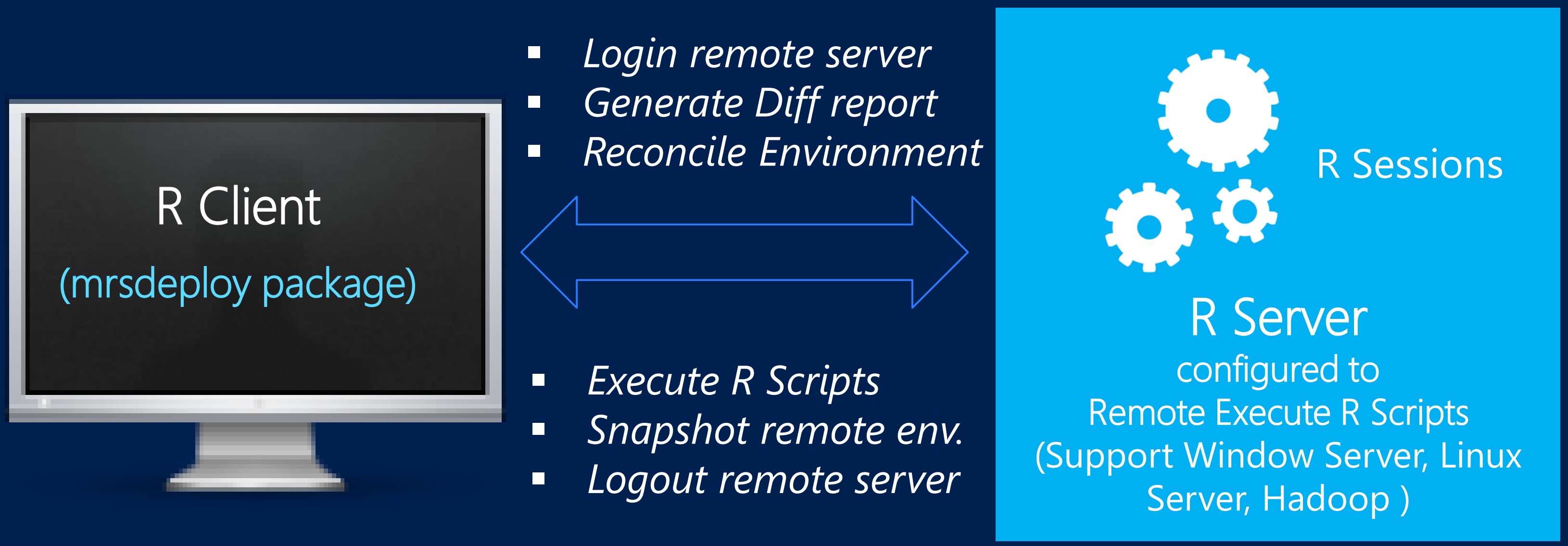
Share / Reuse R code / functions

- Not just models, a data scientist can share any functional code as a service.
- Other data scientists can explore in the repository to re-use those functions.

Remote Execute R scripts

Configure R Server to host remote R sessions

- Built-in remote execute functions in R Client/R Server
- Generate Diff report to reconcile local and remote
- Execute .R script or interactive R commands
- Results come back to local
- Generate working snapshots for resume and reuse
- IDE agnostic



"I can offload the function execution for heavy processing to a chunky server"

"I can validate my scripts against production environment before deployment"

Remote Execution Cheat Sheet

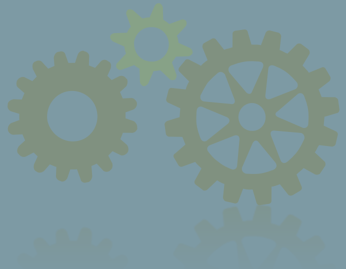
Remote Connection	
remoteLogin	Remote login to the R Server with AD or admin credentials
remoteLoginAAD	Remote login to R Server server using Azure AD
remoteLogout	Logout of the remote session on the DeployR Server.

Remote Execution	
remoteExecute	Remote execution of either R code or an R script
remoteScript	Wrapper function for remote script execution
diffLocalRemote	Generate a 'diff' report between local and remote
pause	Pause remote connection and back to local
resume	Return the user to the 'REMOTE >' command prompt

Snapshot Functions	
createSnapshot	Create a snapshot of the remote session (workspace and working directory)
loadSnapshot	Load a snapshot from the server into the remote session (workspace and working directory)
listSnapshots	Get a list of snapshots for the current user
downloadSnapshot	Download a snapshot from the server
deleteSnapshot	Delete a snapshot from the server

Remote Objects Management	
listRemoteFiles	Get a list of files in the working directory of the remote session
deleteRemoteFile	Delete a file from the working directory of the remote R session
getRemoteFile	Copy a file from the working directory of the remote R session
putLocalFile	Copy a file from the local machine to the working directory of the remote R session
getRemoteObject	Get an object from the remote R session
putLocalObject	Put an object from the local R session and load it into the remote R session
getRemoteWorkspace	Take all objects from the remote R session and load them into the local R session
putLocalWorkspace	Take all objects from the local R session and load them into the remote R session

Microsoft R Server
The Operationalization Engine of your Advanced Analytics



Instant Deployment

- Turn R analytics → Web Service in one line of code;
- Swagger-based REST APIs, easy to consume, with any programming languages, including R!

Deploy to Anywhere

- Deploying Web Service server to any platform: Windows / SQL / Linux/Hadoop
- On Prem or in Cloud

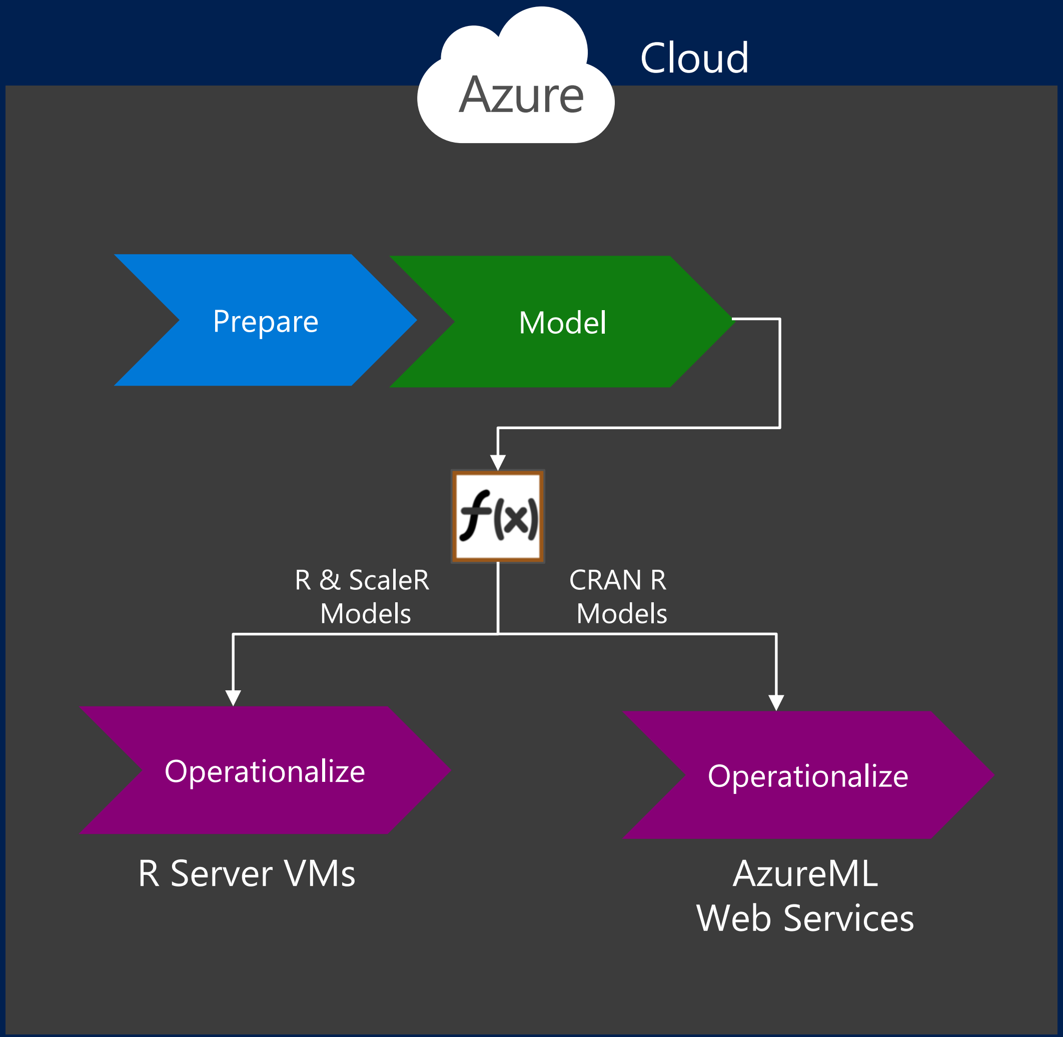
Fast and Scalable

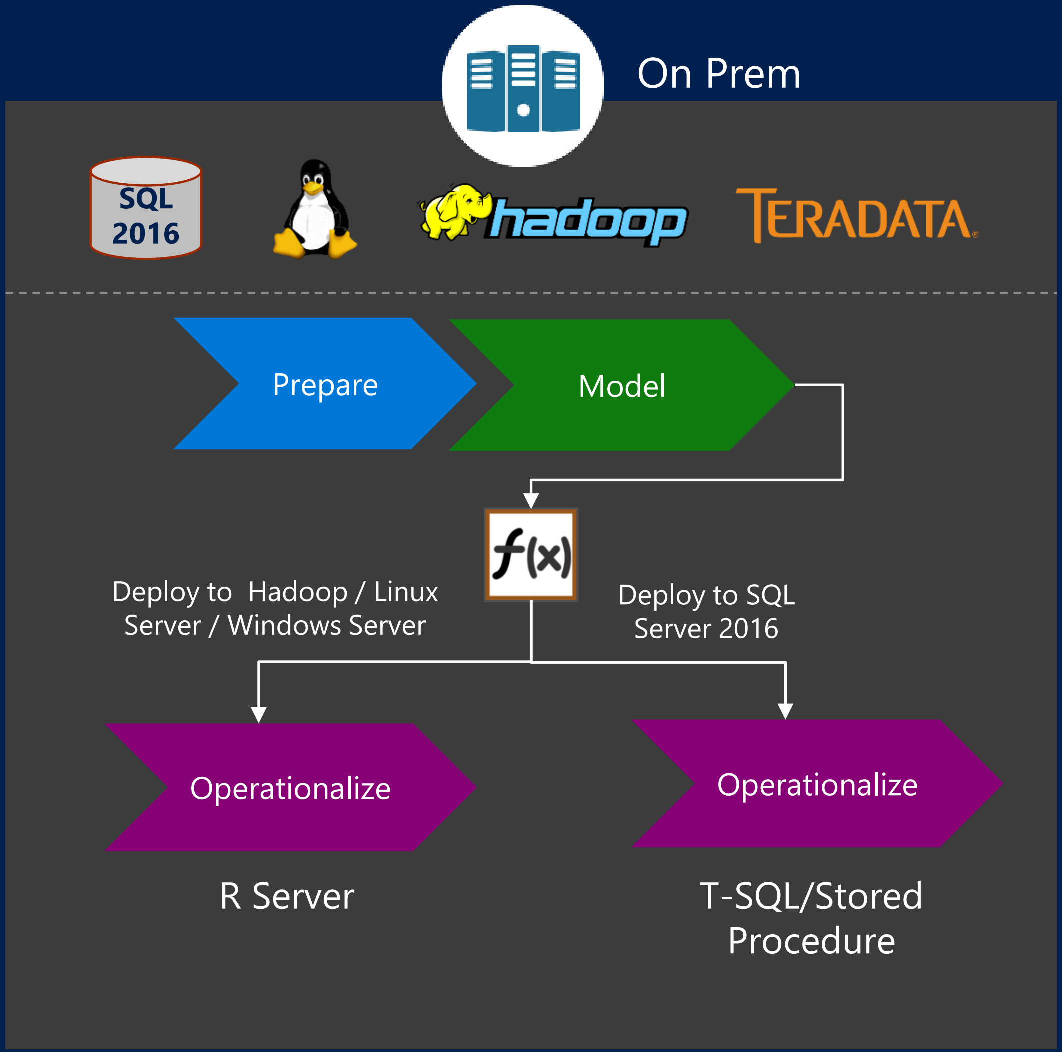
- Fast scoring, real time and batch
- Scaling to a grid for powerful computing with load balancing
- Diagnostic and capacity evaluation tools

Secure and Reliable

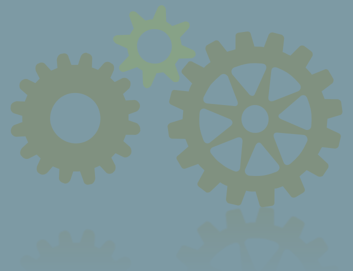
- Enterprise authentication: LDAP / AD/ AAD
- Secure connection: HTTPS with SSL.TSL1.2
- Enterprise grade High Availability

Deploy to Anywhere: On Prem or In Cloud





Microsoft R Server
The Operationalization Engine of your Advanced Analytics

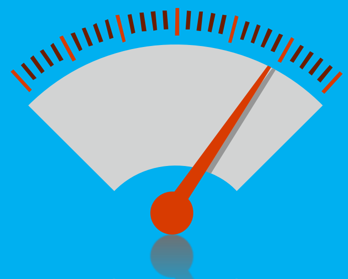


Instant Deployment

- Turn R analytics → Web Service in one line of code;
- Swagger-based REST APIs, easy to consume, with any programming languages, including R!

Deploy to Anywhere

- Deploying Web Service server to any platform: Windows / SQL / Linux/Hadoop
- On Prem or in Cloud



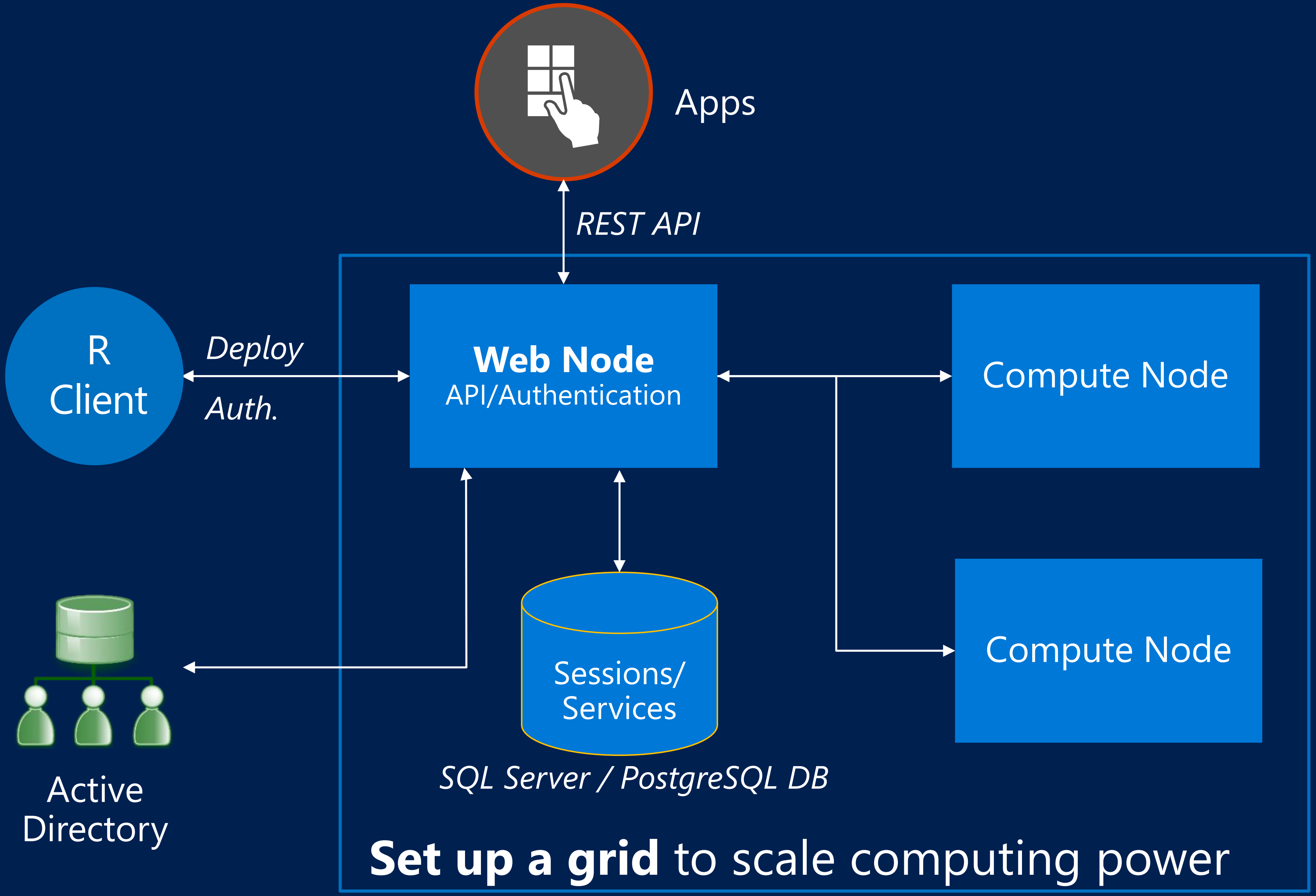
Fast and Scalable

- Fast scoring, real time and batch
- Scaling to a grid for powerful computing with load balancing
- Diagnostic and capacity evaluation tools

Secure and Reliable

- Enterprise authentication: LDAP / AD/ AAD
- Secure connection: HTTPS with SSL.TSL1.2
- Enterprise grade High Availability

Scale up for more powerful computing



- Easily scale up a single server to a grid to handle more concurrent requests
- Load balancing cross compute nodes
- A shared pool of warmed up R shells to improve scoring performance.

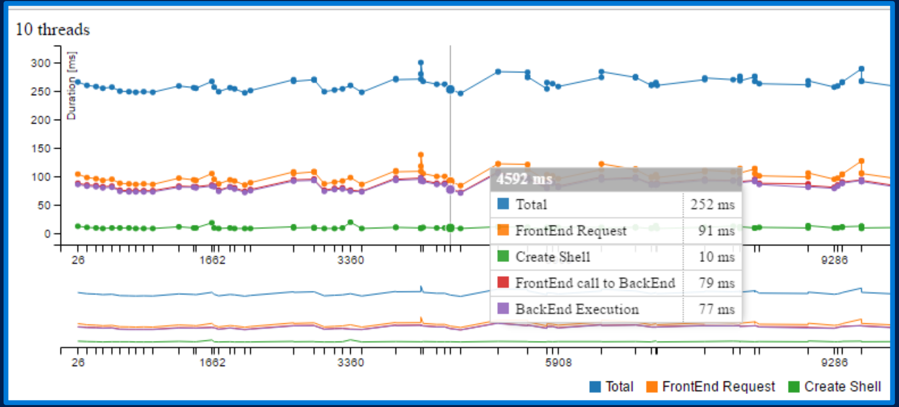
Diagnostic and Evaluation Tools

Diagnostic Tool

- Health check node configuration
- Get system status
- Trace R code execution
- Trace service execution

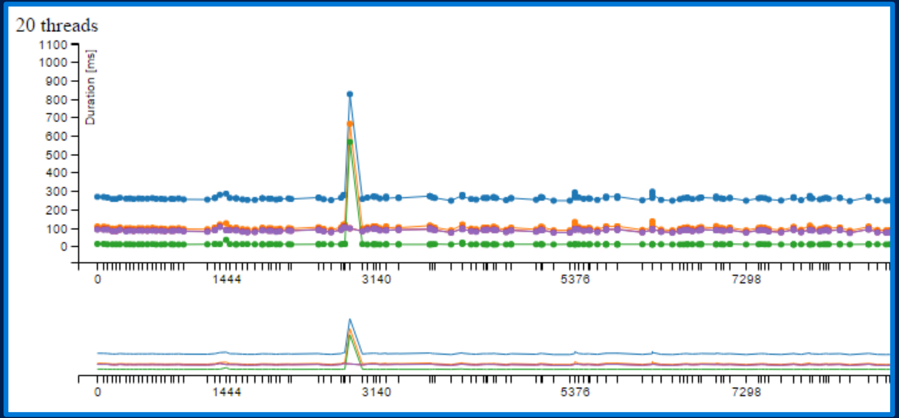
Evaluation Tool

- Evaluate grid capacity
- Simulate traffic per service
- Configure with # of concurrent threads or latency thresholds

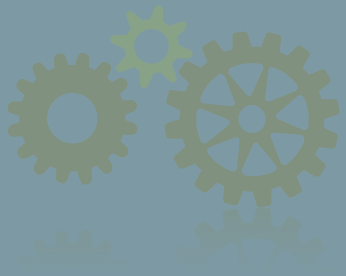


4592 ms

🟦 Total	252 ms
🟧 FrontEnd Request	91 ms
🟩 Create Shell	10 ms
🟥 FrontEnd call to BackEnd	79 ms
🟪 BackEnd Execution	77 ms



Microsoft R Server
The Operationalization Engine of your Advanced Analytics



Instant Deployment

- Turn R analytics → Web Service in one line of code;
- Swagger-based REST APIs, easy to consume, with any programming languages, including R!

Deploy to Anywhere

- Deploying Web Service server to any platform: Windows / SQL / Linux/Hadoop
- On Prem or in Cloud

Fast and Scalable

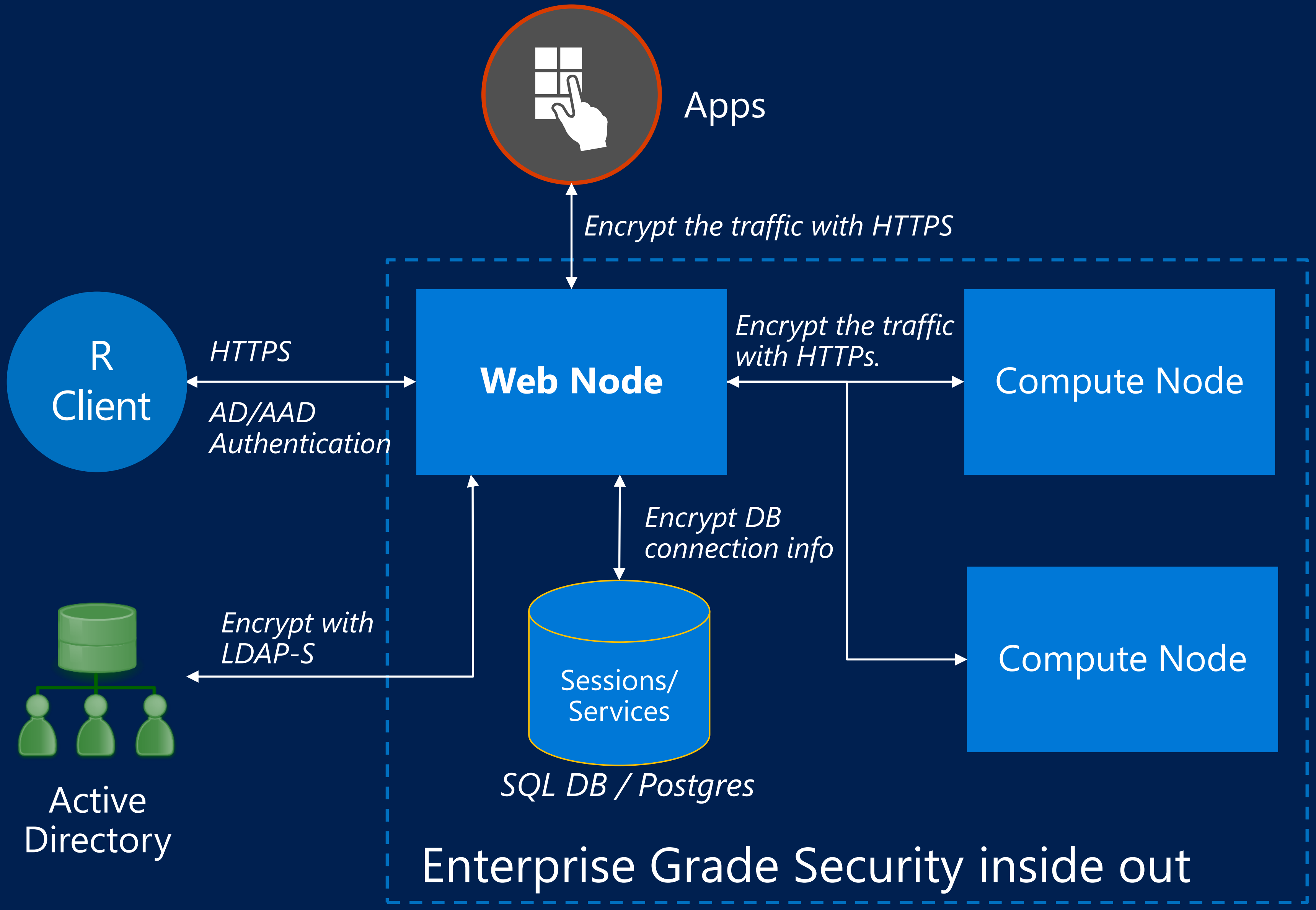
- Fast scoring, real time and batch
- Scaling to a grid for powerful computing with load balancing
- Diagnostic and capacity evaluation tools

Secure and Reliable

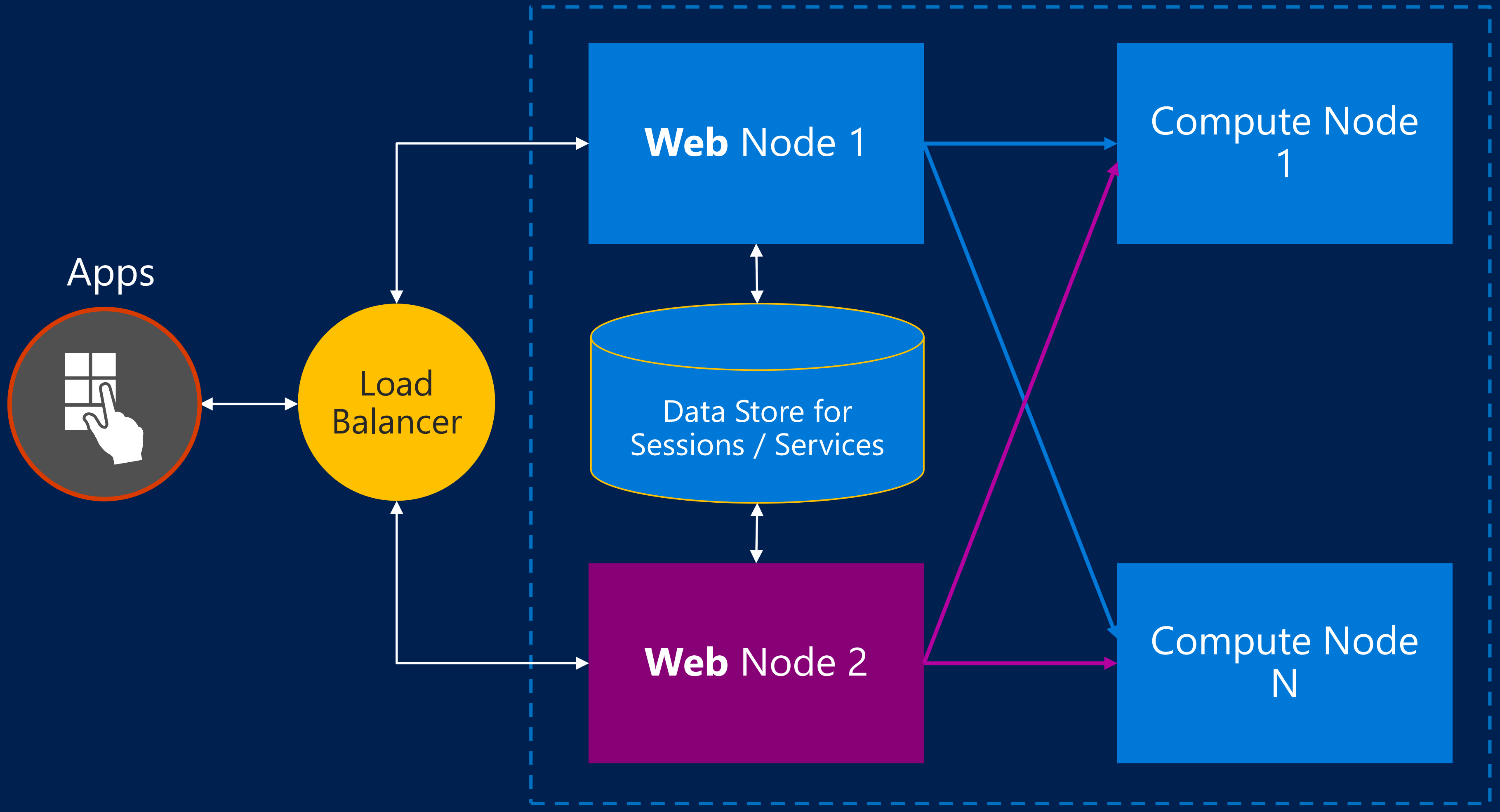
- Enterprise authentication: LDAP / AD/ AAD
- Secure connection: HTTPS with SSL.TSL1.2
- Enterprise grade High Availability

Enterprise Grade Security

- Seamless integration with authentication solution: LDAP/AD/AAD
- Secure connection: HTTPS encrypted by TLS 1.2/SSL
- Compliance with Microsoft Security Development Lifecycle



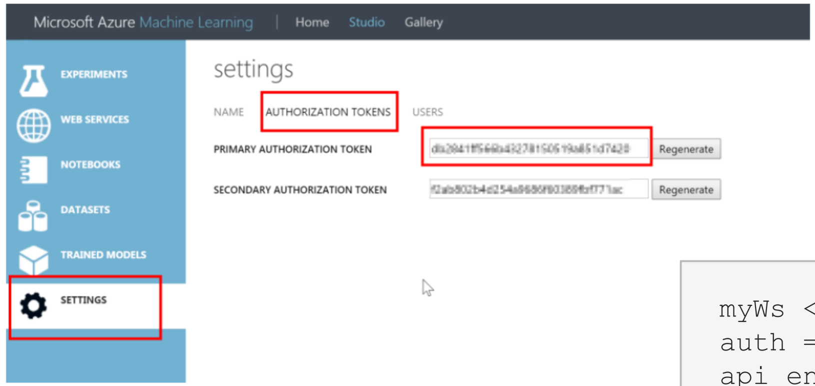
High Availability (disaster recovery)



- Server level HA: Introduce multiple Web Nodes for Active-Active backup / recovery, via load balancer
- Data Store HA: leverage Enterprise grade DB, SQL Server and Postgres' HA capabilities

AzureML R Package - Interact & Publish R to AzureML

- Capture workspace & authorisation token
- Create workspace object in R
- Define and publish an R function to AzureML
- Consume web-service e.g. C#, R, Excel etc

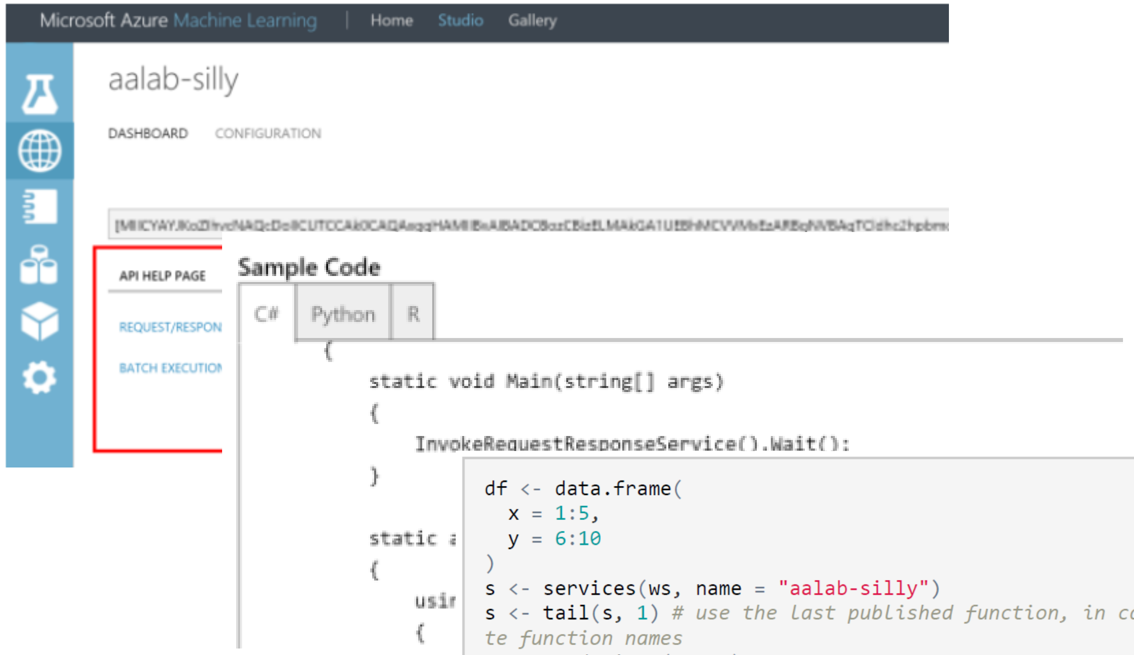


PRIMARY AUTHORIZATION TOKEN [illegible]

SECONDARY AUTHORIZATION TOKEN [illegible]

```r
myWs <- workspace(id = "WORKSPACE ID",
auth = "AUTH KEY",
api_endpoint =
"https://europewest.studio.azureml.net",
management_endpoint =
"https://europewest.management.azureml.net")
```

```r
api <- publishWebService(
  ws,
  fun = add,
  name = "aalab-silly",
  inputSchema = list(
    x = "numeric",
    y = "numeric"
  ),
  outputSchema = list(
    ans = "numeric"
  )
)

api
```



[MIICYAYJKoZIhvcNAQcDoIICUTCCAk0CAQAxggHAMIIBvAIBADCBozCBizELMAkGA1UEBhMCVVMxEzARBgNVBAgTClh2hpbm

```
{
    static void Main(string[] args)
    {
        InvokeRequestResponseService().Wait();
    }

    static a
    {
        usir
        {
```

```r
df <- data.frame(
  x = 1:5,
  y = 6:10
)
s <- services(ws, name = "aalab-silly")
s <- tail(s, 1) # use the last published function, in case of duplica
te function names
ep <- endpoints(ws, s)
consume(ep, df)
```

Demo: R To AzureML

SQL Server 2016 Recap

Run R script

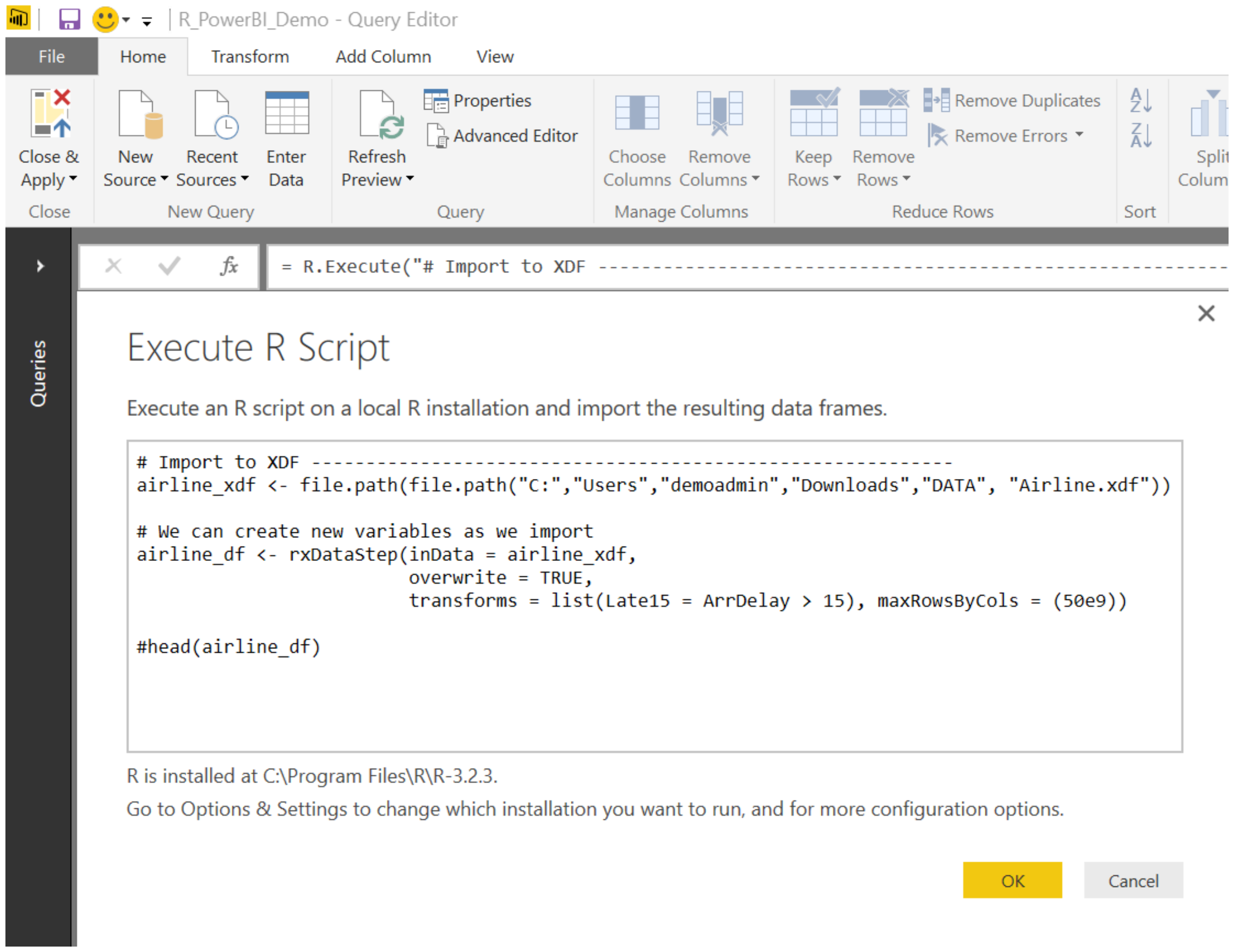
- Use your preferred R IDE
- Set compute context to SQL Server
- Use RevoScaleR rx functions
- Wrap open-source R functions within rxExec for execution on SQL Server

Create SQL query

- Create stored procedure
- Embedded R Language support
- Execute directly in SSMS query

PowerBI - R Integration

Execute R Scripts to create PowerBI data-sources



```
= R.Execute("# Import to XDF ---------------------------------------------
```

Execute R Script

Execute an R script on a local R installation and import the resulting data frames.

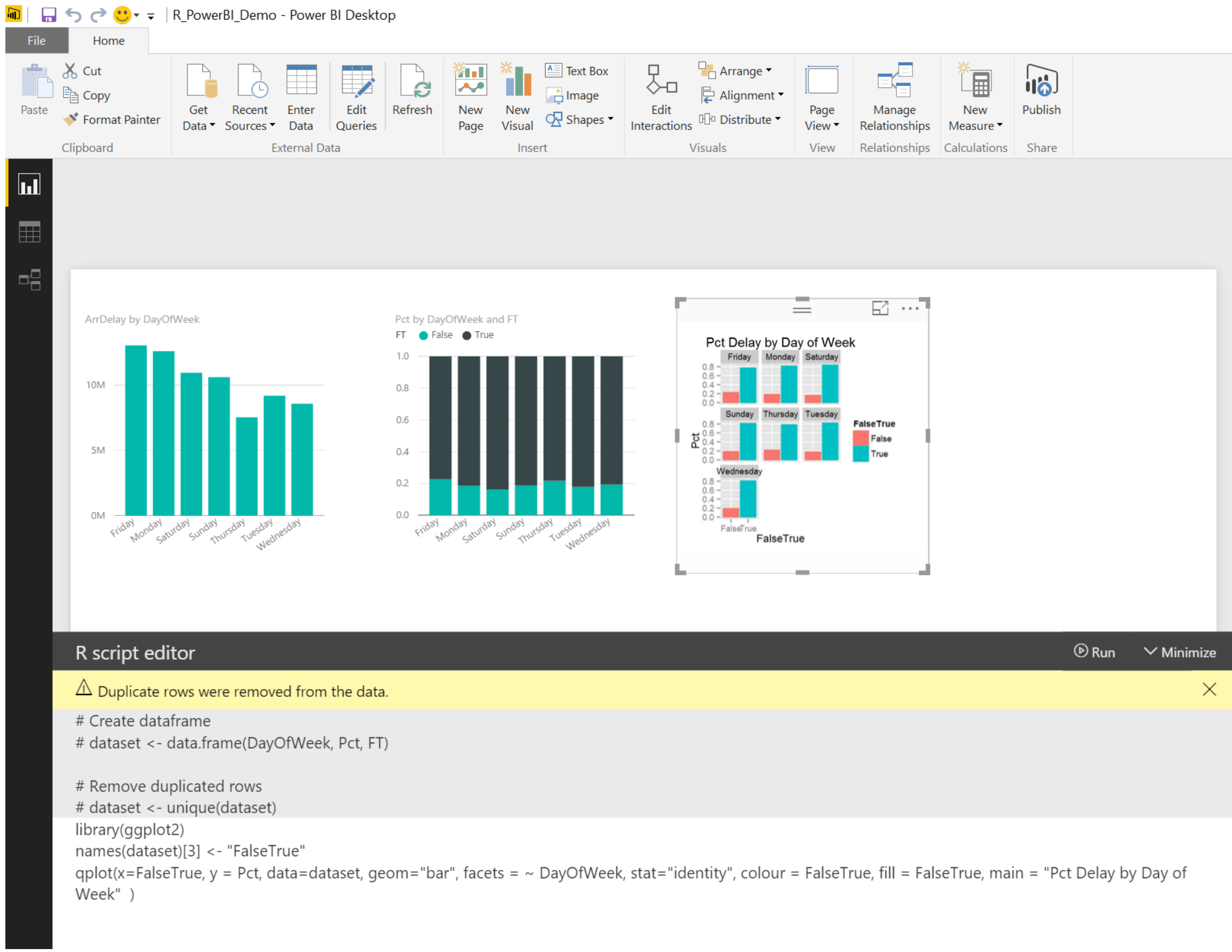
```
# Import to XDF ------------------------------------------------------------
airline_xdf <- file.path(file.path("C:","Users","demoadmin","Downloads","DATA", "Airline.xdf"))

# We can create new variables as we import
airline_df <- rxDataStep(inData = airline_xdf,
                         overwrite = TRUE,
                         transforms = list(Late15 = ArrDelay > 15), maxRowsByCols = (50e9))

#head(airline_df)
```

R is installed at C:\Program Files\R\R-3.2.3.

Go to Options & Settings to change which installation you want to run, and for more configuration options.

OK　Cancel

Use R Visualisations directly in PowerBI



⚠ Duplicate rows were removed from the data.

```
# Create dataframe
# dataset <- data.frame(DayOfWeek, Pct, FT)

# Remove duplicated rows
# dataset <- unique(dataset)
library(ggplot2)
names(dataset)[3] <- "FalseTrue"
qplot(x=FalseTrue, y = Pct, data=dataset, geom="bar", facets = ~ DayOfWeek, stat="identity", colour = FalseTrue, fill = FalseTrue, main = "Pct Delay by Day of
Week"  )
```

Microsoft