This Session will be Recorded

ORACLE

Oracle Machine Learning Feature Highlight

OML4R REST API for Embedded R Execution with Autonomous Database

OML AskTOM Office Hours

Move the Algorithms; Not the Data!

Sherry LaMonica
Oracle Machine Learning Product Management

Akash Dharamshi
Solution Engineer, Oracle Analytics

Agenda

- OML4R embedded R execution overview
- Benefits and motivation
- Interfaces for embedded R execution
- OML4R REST API overview
- Typical workflow
- Demo
- Q&A

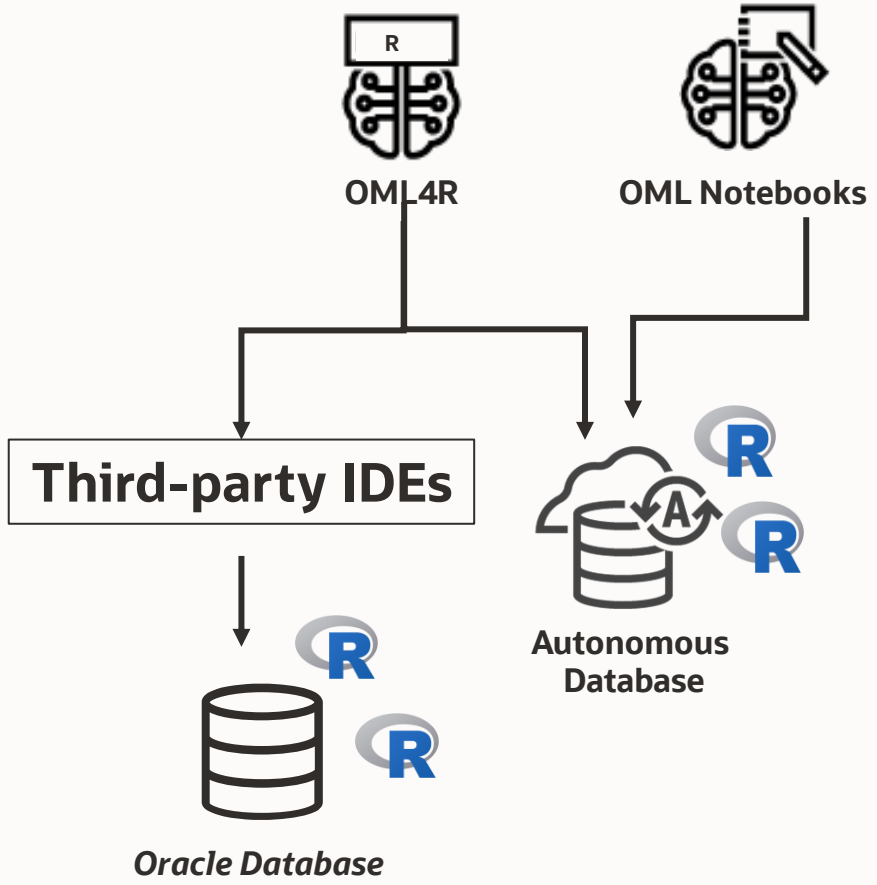
Introduction and Motivation

Oracle Machine Learning for R – Embedded R Execution

Empower data scientists with open source environments

Embedded R Execution runs user-defined R functions on R engines dynamically spawned and managed by the database environment

- Manage user-defined R functions in the database
- Invoke user-defined R functions with automatic data loading
- Data-parallel, task-parallel, and non-parallel execution
- Use open source packages to augment functionality
- R, SQL, and REST interfaces
- Return structured data, JSON, XML, PNG



OML4R Embedded R Execution

Why use embedded R execution?

Application Use

Develop/test R scripts interactively with R interface

Invoke R scripts directly from SQL and REST for production applications

Return structured or image results: table, XML, PNG, JSON

Improved Performance

System-supported data-parallelism and task-parallelism

Score data using third-party R models at scale

Security and Automation

R scripts are stored securely in Oracle Database

Schedule R scripts to run automatically via SQL

Invocations can be handled synchronously or asynchronously

Simplified Deployment

Ability to save and invoke R scripts immediately

Use the database to start, manage, and control invocation of R scripts

Embedded R Execution

Typical use cases

Score data at scale

- Use an R package such as **glm** to build a model on a table and then score chunks of data in parallel
- Score data in batch using the model and multiple R engines for “embarrassingly parallel” scalability
- Access dynamically or store results in the database

Automation

- Automate the building of multiple types machine learning models in parallel
- Store the resulting models in the database

Partitioned data computations

- Build models, compute metrics, or generate plots on data partitions, e.g., per customer or region
- Store the results in the database or access them dynamically in applications or dashboards

Enable simulations

- Run simulations, such as Monte Carlo methods,
- Data may be generated or selected using a random seed supplied to the user-defined function

Interfaces for embedded R execution

Embedded R Execution APIs

Available interfaces – principal uses

R

- Development
- Testing
- Handoff to IT and developers

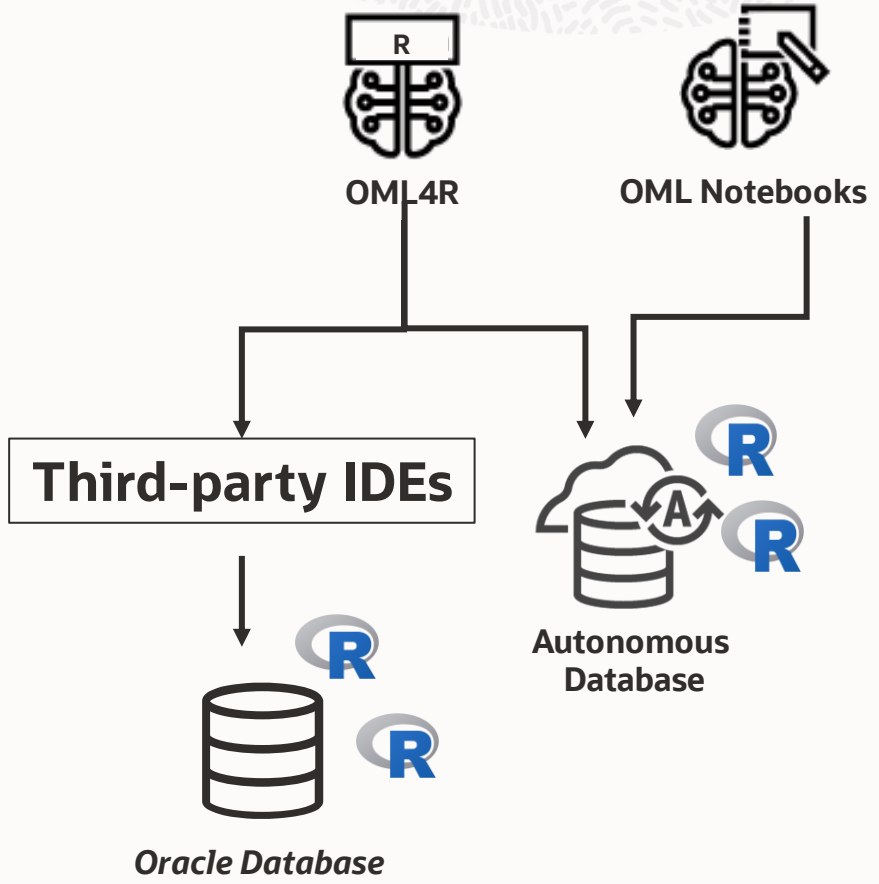
SQL

- Deploying R UDFs in SQL-based applications
- Enable ease of scheduling via DBMS_SCHEDULER

REST (ADB only)

- Deploying R UDFs in REST-based applications
- Lightweight - no dependency on client package

OML4R ERE Documentation



Embedded R Execution Functions

OML4R interfaces for embedded R execution

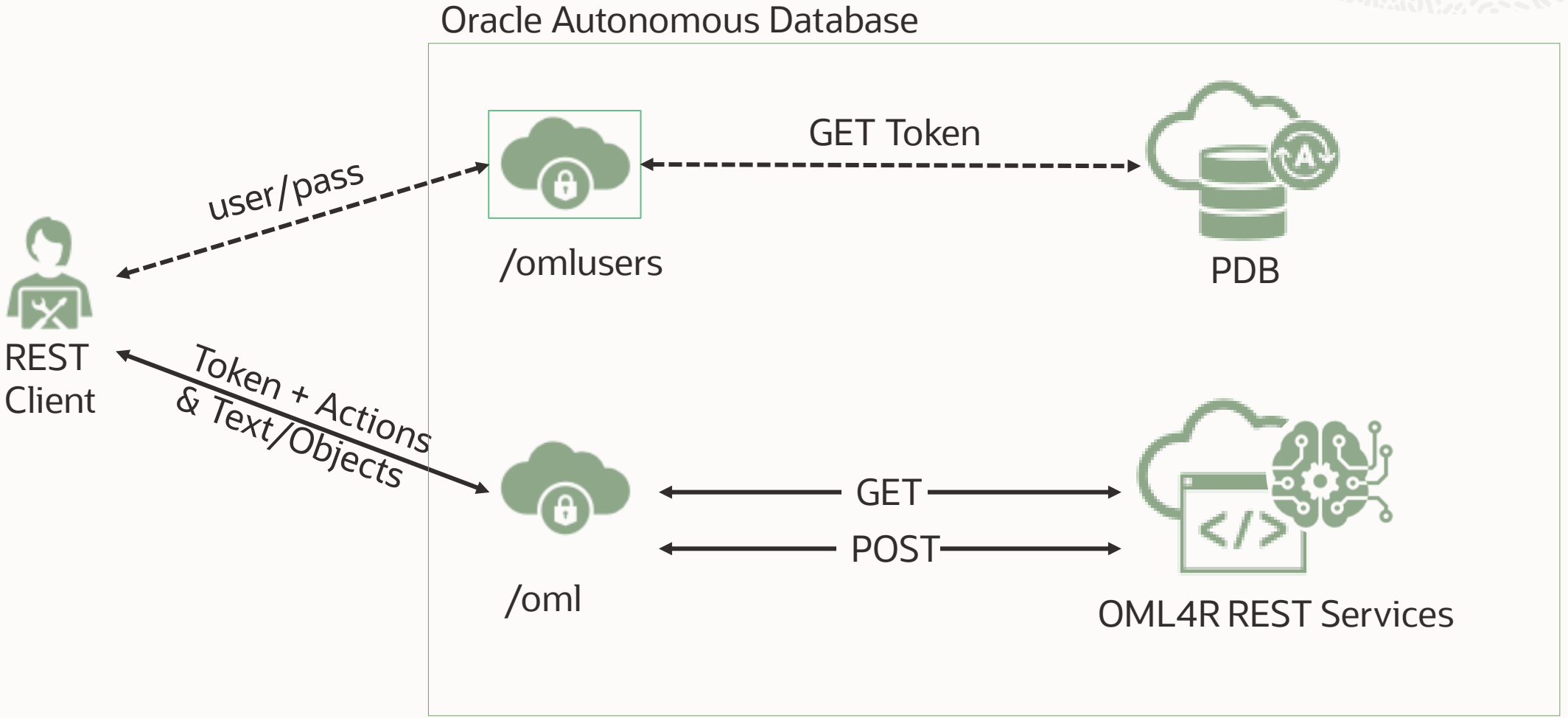
R Interface Function (ADB, ODB)	REST API Endpoint (ADB)	SQL Interface Function (ADB)	SQL Interface Function (ODB)	Purpose
ore.doEval	do-eval	rqEval2	rqEval	Invoke stand-alone R user-defined function (UDF)
ore.tableApply	table-apply	rqTableEval2	rqTableEval	Invoke R UDF with full table input
ore.rowApply	row-apply	rqRowEval2	rqRowEval	Invoke R UDF one row at a time, or multiple rows in “chunks”
ore.groupApply	group-apply	rqGroupEval2	rqGroupEval	Invoke R UDF on data indexed by grouping column
ore.indexApply	index-apply	rqIndexEval2	rqIndexEval	Invoke R UDF N times

ADB = Oracle Autonomous Database
ODB = Oracle Database

REST API for embedded R execution

OML4R REST Architecture

Connectivity and use from client



Oracle Machine Learning RESTful URLs

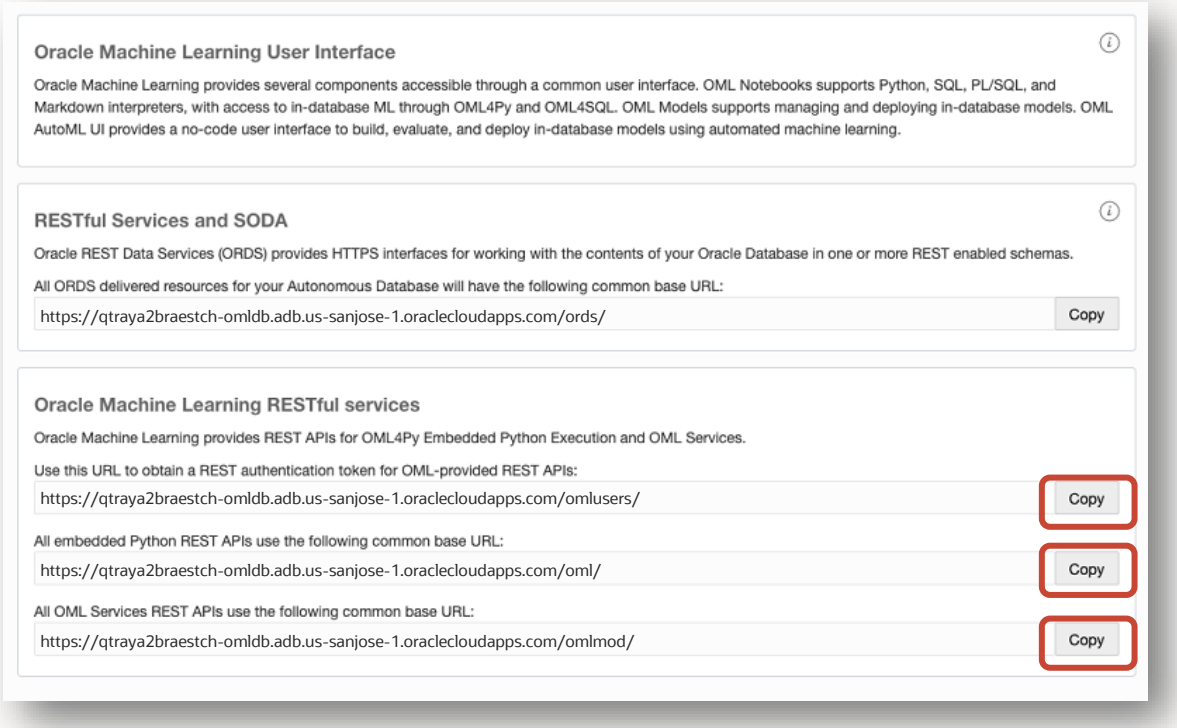
Where can I find the URLs that correspond to my tenancy?

Location of REST URLs

From your Oracle Autonomous Database instance:

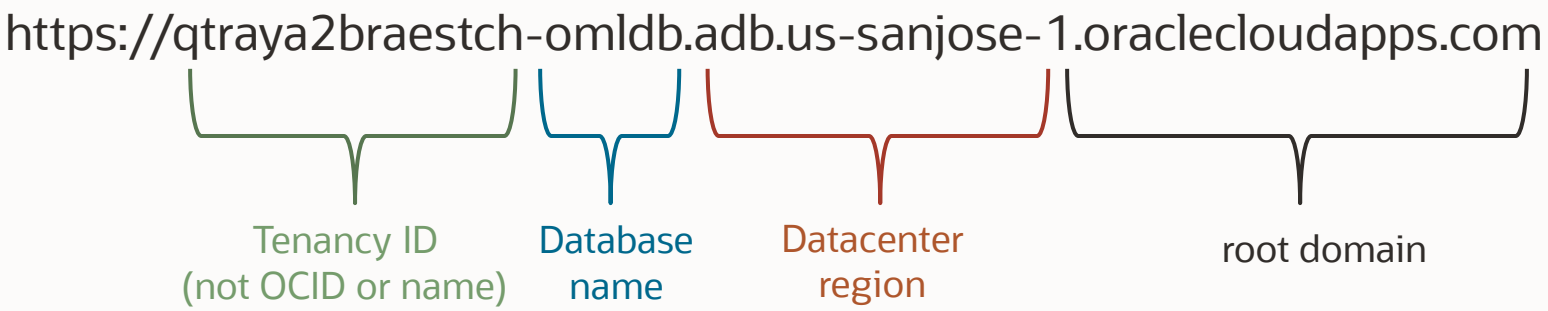
1. Click **Database Actions**

2.Scroll down to **Oracle Machine Learning RESTful Services** under **Related Services** and copy the URL



OML REST URL

Standard URL for all OML REST endpoints



Same for Token acquisition

OML4R REST API - Methods

GET and POST methods

Admin

POST

Token using OML user and password

Generic

GET

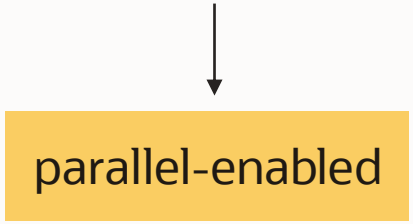
- Metadata for all Versions: Version 1 Metadata
- Open API Specification

Invoke Scripts

POST

Invoke R UDF:

- do-eval
- table-apply
- group-apply
- row-apply
- index-apply



Asynchronous mode available

List Scripts

GET

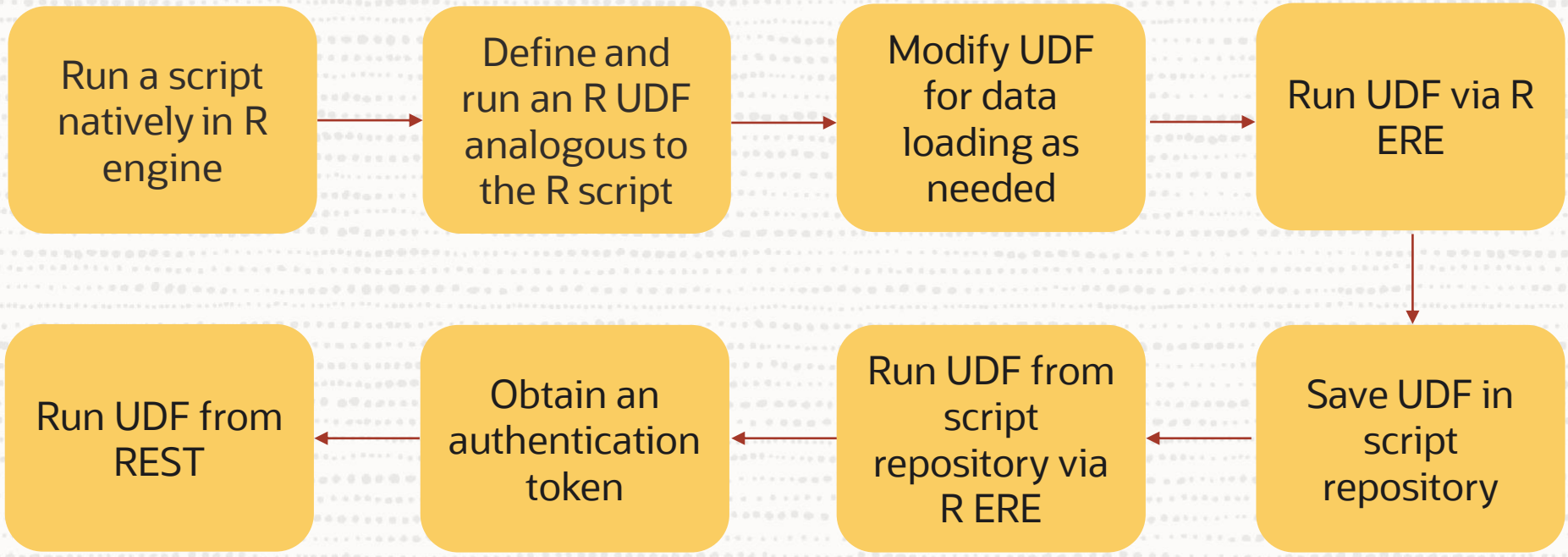
List available scripts in script repository

Asynchronous Mode

GET

- Job Status
- Job Result

Workflow



Build an ML model on the iris data set using OML4R ERE from R

```
build.glm.mod <- function(dat,dsname) {
  dat$Species <- as.factor(dat$Species)
  mod <- glm(Species~.,family=binomial(link='logit'),dat)
  ore.save(mod, name=dsname)
  TRUE
}

ore.scriptCreate('build.glm.mod', build.glm.mod)

ore.sync(table='IRIS')  # get ore.frame proxy object

ore.tableApply(IRIS, FUN.NAME='build.glm.mod',
               dsname= 'GLM-iris-species',
               ore.connect=TRUE)
```

Save resulting model in database datastore

Store the UDF in the script repository

Provide proxy object IRIS
Invoke UDF by name
Specify the datastore name

Store UDF in script repository using the R and SQL APIs

```
score.glm.mod <- function(dat, dsname) {
      ore.load(dsname)
      dat$Prediction <- predict(mod, newdata = dat)
      dat[,c("Species","Prediction")]
    }
ore.scriptCreate(name = 'score.glm.mod',
                 FUN = score.glm.mod, overwrite = TRUE)
```

SQL

```
BEGIN
 sys.rqScriptCreate('score.glm.mod',
   'function(dat, dsname) {
      ore.load(dsname)
      dat$Prediction <- predict(mod, newdata = dat)
      dat[,c("Species","Prediction")]
    }',FALSE, TRUE); -- not sharing function and enable overwrite
END;
```

POST

Request a token

Initial call to get a token to access all other OML REST endpoints

```
$ curl -X POST \
  --header 'Content-Type: application/json' \
  --header 'Accept: application/json' \
  -d '{"grant_type":"password", "username": "YourOMLuser", "password":"YourOMLpass"}'\
       "ADB_URL/omlusers/api/oauth2/v1/token"
```


GET

List Available Scripts

Call to get the scripts available to the OML user

A list of functions saved as scripts by or shared with the current user is returned.
The functions were previously saved to the script repository.

For the following REST call, consider:
`ADB_URL/oml = OML_URL, and remember to provide the full Token after "Bearer“`

```
$ curl –X GET 'OML_URL/api/r-scripts/v1/scripts’ \
 --header 'Accept: application/json' \
 --header 'Authorization: Bearer eyJhbGciOiJSUzI1NiJ9.....=='
```

This is the token

GET

Open API description

Call to get the Open API description for the current OML4R REST API

To review the Open API specification for the OML Services REST endpoints, pass a valid token.

For the following REST call, consider:

`ADB_URL/oml = OML_URL, and remember to provide the full Token after "Bearer“`

```
$ curl -X GET 'OML_URL/api/r-scripts/v1' \
 --header 'Accept: application/json' \
 --header 'Authorization: Bearer eyJhbGciOiJSUzI1NiJ9.....=='
```

This is the token

POST

Run the scoring function from the REST API – Autonomous Database

Example of parallel partitioned data flow using third party package – REST API

For the following REST call, consider:

ADB_URL/oml = OML_URL, and remember to provide the full Token after "Bearer“

```
$ curl –i –X POST 'OML_URL/api/r-scripts/v1/scripts/row-apply/score.glm.mod’ \
       --header 'Content-Type: application/json’ \
       --header 'Accept: application/json’ \
       -d ‘{“input":"IRIS", dsname=“ds1”, "rows":50,
            "parallelFlag":true, "service":"MEDIUM"}’ \
       --header 'Authorization: Bearer eyJhbGciOiJSUzI1NiJ9.....+='
```

This is the token

Parallelism - REST API for Embedded R Execution

Oracle Autonomous Database

- Extends parallelism by enabling different service levels to **manage the load** on the system
  - LOW - maximum 2 degrees of parallelism
  - MEDIUM - maximum of 4 degrees of parallelism
  - HIGH - maximum of 8 degrees of parallelism

- Parallelism for service levels **LOW, MEDIUM, HIGH**
  - *parallelFlag=True* results in parallelism corresponding to the service level above

- Parallelism is applicable to *row-apply*, *group-apply* and *index-apply*.

Demo

For more information...

OML Webpage
https://oracle.com/machine-learning

Machine Learning Blog
https://bit.ly/omlblogs

GitHub Repository
https://bit.ly/omlgithub

OML Office Hours
https://bit.ly/omlofficehours

OML4R Documentation
https://docs.oracle.com/en/database/oracle/machine-learning/oml4r

Thank you

sherry.lamonica@oracle.com

akash.dharamshi@oracle.com