**Microsoft Machine Learning Server**

It includes a collection of packages, interpreters, and infrastructure for developing and deploying **R** **and Python**-based machine learning and data science solutions on a range of platforms

To me ML Client and ML Server seem to be distributions, i.e. a collection of packages and programs.

**Server Side:**

Available **on premise** (Supported Platforms):

Linux (Ubuntu), Windows (use server-OS), Hadoop

available **in the cloud** as follows:

* [Machine Learning Server as preconfigured Azure virtual machine on Linux](https://docs.microsoft.com/en-us/machine-learning-server/install/machine-learning-server-azure-vm-on-linux)
* [Machine Learning Server as preconfigured Azure virtual machine on Windows](https://docs.microsoft.com/en-us/sql/advanced-analytics/r/provision-the-r-server-only-sql-server-2016-enterprise-vm-on-azure)
* [Machine Learning Server on Azure HDInsight](https://docs.microsoft.com/en-us/machine-learning-server/install/machine-learning-server-on-azure-hdinsight) (Hadoop, Spark)
* [R Server on the Microsoft Data Science Virtual Machine](https://docs.microsoft.com/en-us/machine-learning-server/install/r-server-vm-data-science) (ML Server coming soon)

**Client Side:**

Microsoft R Client is a free data science tool for high performance analytics. R Client is built on top of Microsoft R Open so you can use any open source R packages to build your analytics. Additionally, R Client introduces the [powerful ScaleR technology](https://docs.microsoft.com/en-us/machine-learning-server/r/tutorial-revoscaler-data-import-transform) and its proprietary functions to benefit from parallelization and remote computing.

R Client allows you to work with production data locally using the full set of ScaleR functions, but there are some constraints. On its own, the data to be processed must fit in local memory, and processing is capped at two threads for RevoScaleR functions.

To benefit from disk scalability, performance and speed, push the compute context using rxSetComputeContext() to a production instance of Microsoft R Server (or R Server) such as [SQL Server Machine Learning Services](https://msdn.microsoft.com/en-us/library/mt604845.aspx) and Machine Learning Server for Hadoop.

(You can offload heavy processing to Machine Learning Server or test your analytics during their developmentYou by running your code remotely using [remoteLogin() or remoteLoginAAD()](https://docs.microsoft.com/en-us/machine-learning-server/r/how-to-execute-code-remotely) from the mrsdeploy package.)

The ability to switch a compute context means that you can push execution to an interpreter on another machine. Local is the default. Switching to a remote compute context is typically done to get better performance

**Components of R Server/ R Client:**

* R support includes [Microsoft R Open 3.4.1](https://mran.microsoft.com/open/), which is based on R-3.4.1. Python support is based on Anaconda 4.2 over Python 3.5
* MS R-Packages for deployment: MicrosoftML, mrsdeploy, olapR, RevolQQ, RevoPemaR, RevoScaleR, RevoUtils, sqlrutils

…

What is operationalization in this context:

Operationalization refers to the process of publishing R models and code to Microsoft R Server in the form of web services and the consumption of these services within client applications to affect business results. generate predictions from the pre-trained model on demand.

**Listed Pros/ Concepts**

# *Computing context* (you can push execution to an interpreter on another machine. Local is the default) RevoScaleR package

# *Distributed and parallel computing* (breakdown of a complicated computation into pieces that can be performed independently, while maintaining a framework that allows for the results of those independent computations to be pulled together to create the final result.)

# *Web services:* R and Python code and models can be deployed as web services. models and code can be accessed and consumed in R, Python, programmatically using REST APIs, or using Swagger generated client libraries. Web services can be deployed from one platform and consumed on another. They can be consumed by authenticated users synchronously, in realtime, or in batch mode. You can consume directly in R or Python, using APIs, or in your preferred language via Swagger. by providing them with the name and version of the web service. Requirement! Before you can deploy and work with web services, you must have access to a Machine Learning Server instance [configured to host web services](https://docs.microsoft.com/en-us/machine-learning-server/operationalize/configure-start-for-administrators#configure-server-for-operationalization). two types of web services: standard and realtime.

# *Standard*: (fast execution and scoring of arbitrary Python or R code and models and model assets. should also define the required inputs and any output

# *Realtime:* only accept models (not arbitrary code) created with the supported functions from packages installed with the product. Even lower latency. model is only loaded once in the compute node and can be scored multiple times. Don’t need to specify inputs and outputs (dataframes are assumed) nor code (only serialized models are supported) Supported R functions: only (model-)functions of the RevoScaleR and MicrosoftML packages, see <https://docs.microsoft.com/en-us/machine-learning-server/operationalize/concept-what-are-web-services>

# How to deploy R-models on MS ML Server

# <https://docs.microsoft.com/en-us/machine-learning-server/operationalize/concept-operationalize-deploy-consume>

# Data scientists work locally with [Microsoft R Client](https://docs.microsoft.com/en-us/machine-learning-server/r-client-get-started) in their preferred R IDE and favorite version control tools to build scripts and models. Using the mrsdeploy package that ships with Microsoft R Client and R Server, the data scientist can develop, test, and ultimately deploy these R analytics as web services in your production environment. An R Server web service is an R code execution on the [operationalization compute node](https://docs.microsoft.com/en-us/machine-learning-server/operationalize/configure-start-for-administrators#configure-server-for-operationalization) . Each web service is uniquely defined by a name and version.

# The mrsdeploy package provides functions for publishing and managing a web service that is backed by the R code block or script you provided. Anyone who wishes to consume the service must have [authenticated access](https://docs.microsoft.com/en-us/machine-learning-server/operationalize/configure-authentication) to an instance of Microsoft R Server with its [operationalization feature configured](https://docs.microsoft.com/en-us/machine-learning-server/operationalize/configure-start-for-administrators#configure-server-for-operationalization) .

# <https://docs.microsoft.com/en-us/machine-learning-server/operationalize/quickstart-publish-r-web-service>

# How to consume web services

# Consume the service in R:

# By publisher:

# When you publish, update, or get a web service, an API instance is returned as an [R6](https://cran.r-project.org/web/packages/R6/index.html) class. This instance is a client stub you can use to consume that service and view its service holdings.

# <https://docs.microsoft.com/en-us/machine-learning-server/operationalize/quickstart-publish-r-web-service>

# <https://docs.microsoft.com/en-us/machine-learning-server/operationalize/how-to-consume-web-service-interact-in-r>

# Users can get the Swagger file they need to consume the service directly in R or via the API. Users can consume the service directly using a single consumption call or do a asynchronous BATCH call

# By another data scientist (directly in R):

# You can share the name and version of a web service with fellow data scientists so they can call that service in R using the functions in the mrsdeploy package. After authenticating, data scientists can use the getService() function in R to call the service. Then, they can get details about the service and start consuming it.

# Consume the service from REST API

# Application developers can call and integrate a web service into their applications using the service-specific Swagger-based JSON file and by providing any required inputs to that service. request the file as an authenticated user with an [active bearer token](https://docs.microsoft.com/en-us/machine-learning-server/operationalize/how-to-build-api-clients-from-swagger-for-app-integration#authentication) in the request header (since all API calls must be authenticated). The URL is formed as follows:

GET /api/<service-name>/<service-version>/swagger.json

# Exkurs: What is Swagger?

The goal of Swagger™ is to define a standard, language-agnostic interface to REST APIs which allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection. When properly defined via Swagger, a consumer can understand and interact with the remote service with a minimal amount of implementation logic. Similar to what interfaces have done for lower-level programming, Swagger removes the guesswork in calling the service.

**Download prior R Server releases**

The following table provides links for downloading older versions of Microsoft R Server.

| **Site for R Server** | **Edition** | **Details** |
| --- | --- | --- |
| [Visual Studio Dev Essentials](http://go.microsoft.com/fwlink/?LinkId=717968&clcid=0x409) | Developer (free) | This option provides a zipped file, free when you sign up for Visual Studio Dev Essentials. Developer edition has the same features as Enterprise, except it is licensed for development scenarios.   1. Click **Join or Access Now** and enter your account information. 2. Make sure you're in the right place: my.visualstudio.com. 3. Click **Downloads**, and then search for Microsoft R. |
| [Volume Licensing Service Center (VLSC)](http://go.microsoft.com/fwlink/?LinkId=717966&clcid=0x409) | Enterprise | Sign in, search for R Server. Choose the right version for your OS. |
| [MSDN subscription downloads](https://msdn.microsoft.com/subscriptions/downloads/hh442898.aspx) | Developer or Enterprise | Subscribers can download software at given subscription levels. Depending on your subscription, you can get either edition. |

# And draw picture of architecture for Joel