

# Architecture Overview of Machine Learning Services in MI

**Content here is MS Confidential and for internal use only. Please do not share this with Customers**

## Contents

- [Content here is MS Confidential and for internal use only. ...](#)
- [Overview of Advanced Analytics in SQL](#)
  - [More Information](#)
- [Current Architecture on Box](#)
- [Architecture for ML Services on SQL MI](#)
- [Communication between XdbpackageLauncher and SQL](#)
- [Unzip and Mount R and python VHDs with XdbpackageLa...](#)
- [Satellite Process Isolation using AppContainers](#)
- [Code and Data Packages](#)
- [Resource Governance](#)
- [External Library Management for R and Python](#)
- [Deployment in Azure](#)
- [SLOs supported](#)


## Overview of Advanced Analytics in SQL

Advanced analytics (In DB R Services) feature was released in SQL2016. This feature allows application to run R Scripts through TSQL. R Scripts are run in external processes within appcontainers which allows the secure execution of external R scripts.

Integration with SQL server allows keeping the data with SQL Server itself.

Below links have more information on this feature:

<https://msdn.microsoft.com/en-us/library/mt604845.aspx> 

<https://blogs.technet.microsoft.com/dataplatforminsider/2016/03/29/in-database-advanced-analytics-with-r-in-sql-server-2016/> 

## More Information

The ML Service Limited Preview is only for "Managed Instance" and not Azure SQL Database.

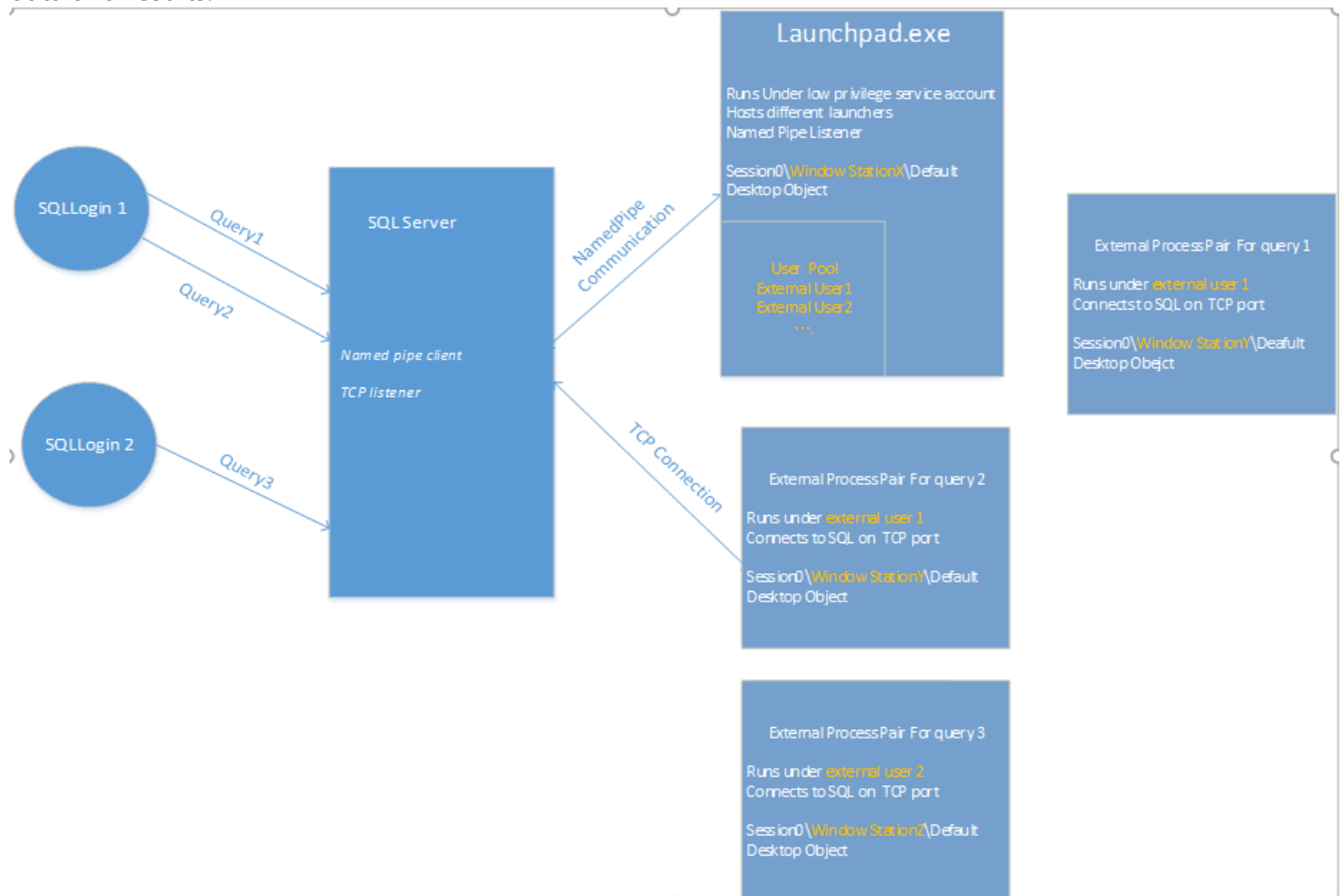
In Azure SQL Database we had this feature in preview earlier which will not be released for GA. "As of 30 June 2020, Azure SQL Database Machine Learning Services with R (preview) - support will be discontinued and the preview will not be released for general availability. R scripts in use after 30 June 2020 will not work" See [here](#) ☐

To continue working with machine learning in Azure SQL, see Machine Learning Services in [Azure SQL Managed Instance \(preview\)](#). ☐

## Current Architecture on Box

There are following major services/components in current Box Architecture:

1. SQL Service running under configured SQL Account.
2. Launchpad service responsible for launching external process, managing isolation between external process and managing resources for external process. Resource governance is defined by SQL and enforced by Launchpad.
3. R runtime\External R Process responsible for executing R Scripts. Also communicates with SQL for input data and results.



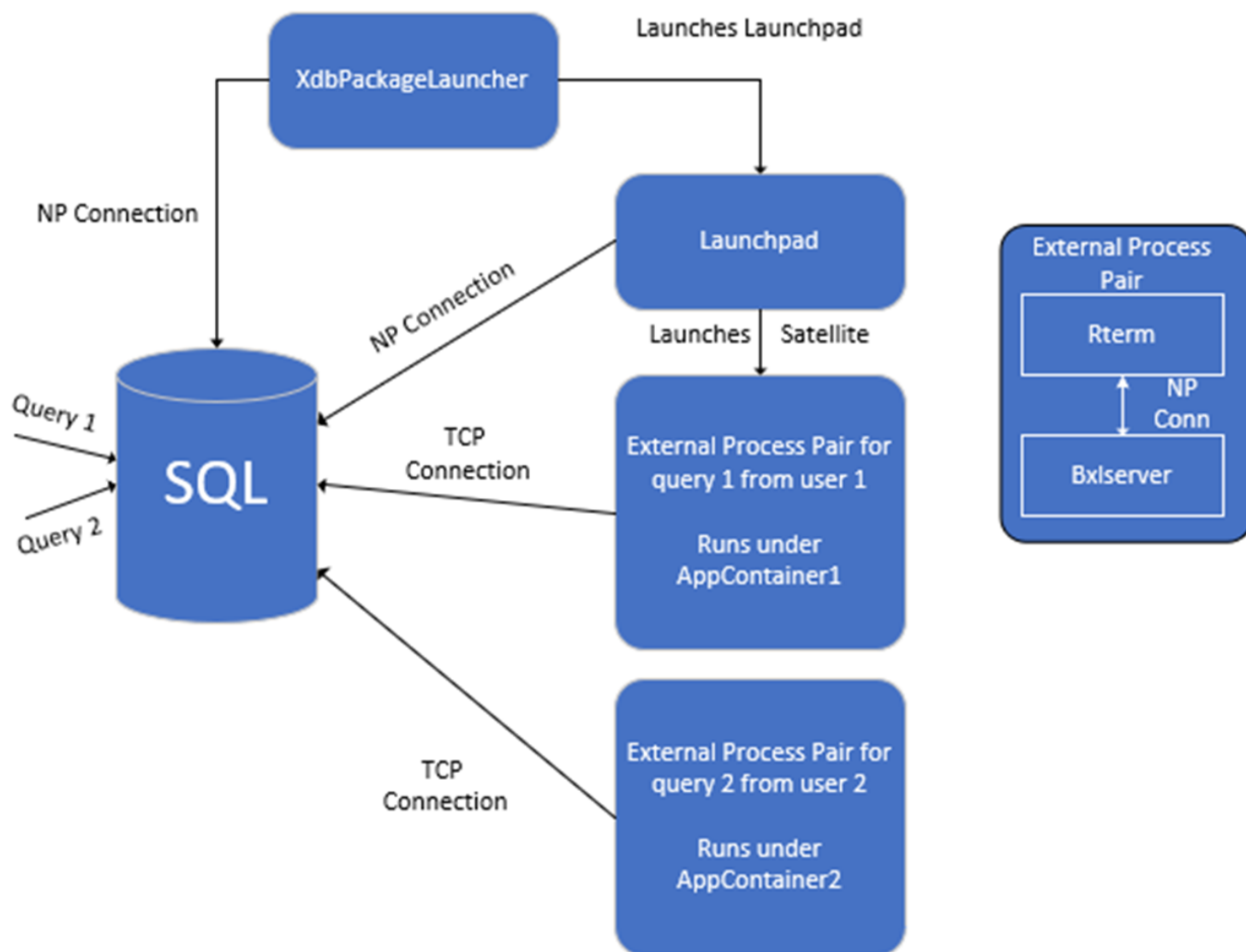
## Architecture for ML Services on SQL MI

Two primary things to consider while deploying an application in Azure are secure execution of application and Resource Governance.

The proposed process layout is as following:

1. SQL server runs as standalone process
2. **XdbPackageLauncher** will be installed as a code package and runs as standalone process.
3. **Launchpad** and **External** Process are launched in appcontainer by XdbPackageLauncher to achieve secure execution and isolation among different external process. Launchpad, R and Python data packages would be used for this purpose.
4. SQL External Resource governance will be disabled. **XdbPackageLauncher** does the resource governance instead of it happening at fabric level.

This means that we will have resource governance at data package level in windows fabric for data packages being launched by XdbPackageLauncher.



## Communication between XdbpackageLauncher and SQL

A package launcher service (**XdbPackageLauncher**) launches launchpad on first query execution. SQL serves as the server end for communication between Xdb package launcher and SQL. Communication is done over a **namedpipe** connection through Namedpipe acing, Only xdb package launcher can connect to SQL over named pipe. XdbPackageLauncher Launches **Sandbox** Process

## Unzip and Mount R and python VHDs with XdbpackageLauncher.

- R data package is around 823MB and contains 8k+ files.
- Python data package is around 2.7GB and contains 35k+ files

Deploying such huge packages would result in increased time in deployments (ring and app creation). This would increase server creation time too.

An optimization is made in this case to use compressed R/Python VHDs. Going ahead with the compressed VHD in the python and R data packages, would reduce the size of Python VHD to 824MB and R VHD to 222 MB. Vhds solve the problem of large number of files by reducing the total number of files to one.

Since on deployment, the data packages have compressed vhds, XdbPackageLauncher code package would unzip the vhds in the data package folder and mount them under C:\WFRoot\ext\ directory as mounted folders.

Example: C:\WFRoot\ext\R\_VERSION A global mutex would be used during unzip and mount so that multiple XdbPackageLauncher processes do not interfere with each other. (in case of GM- models where instances are placed on the same node. So, multiple XdbPackageLauncher processes would be running at the same time).

The vhds mounted do not use drive letters and would be a ReadOnly mount.

## Satellite Process Isolation using AppContainers

Launchpad creates a fixed number of **Appcontainers** at startup phase, and then creates a working directory for each Appcontainer and associates the working directory with corresponding Appcontainer SID.

A process within an AppContainer runs with an Integrity Level of low, which effectively means it has no access to almost everything, until unless granted. External processes from different users would be running under different Appcontainers.

Processes inside Appcontainer would have limited access to resources: it would only be allowing to access the resource which we grant the permission explicitly. For any external query Launchpad decides which Appcontainer the external process should be launched inside, and which working directory it can use. When satellite process tries to establish connection with SQL Server, SQL server needs to check if the satellite process under the same account as launchpad and it is also an appcontainer process by verifying the appcontainer sid.

## Code and Data Packages

To deploy on SQL MI, we will need to have packages for Launchpad, R and Python. Packages we can think of as XCOPY bits, that have everything bundled to instantiate the services. We have the XdbPackageLauncher code package and data packages for R, Python and Launchpad.

## Resource Governance

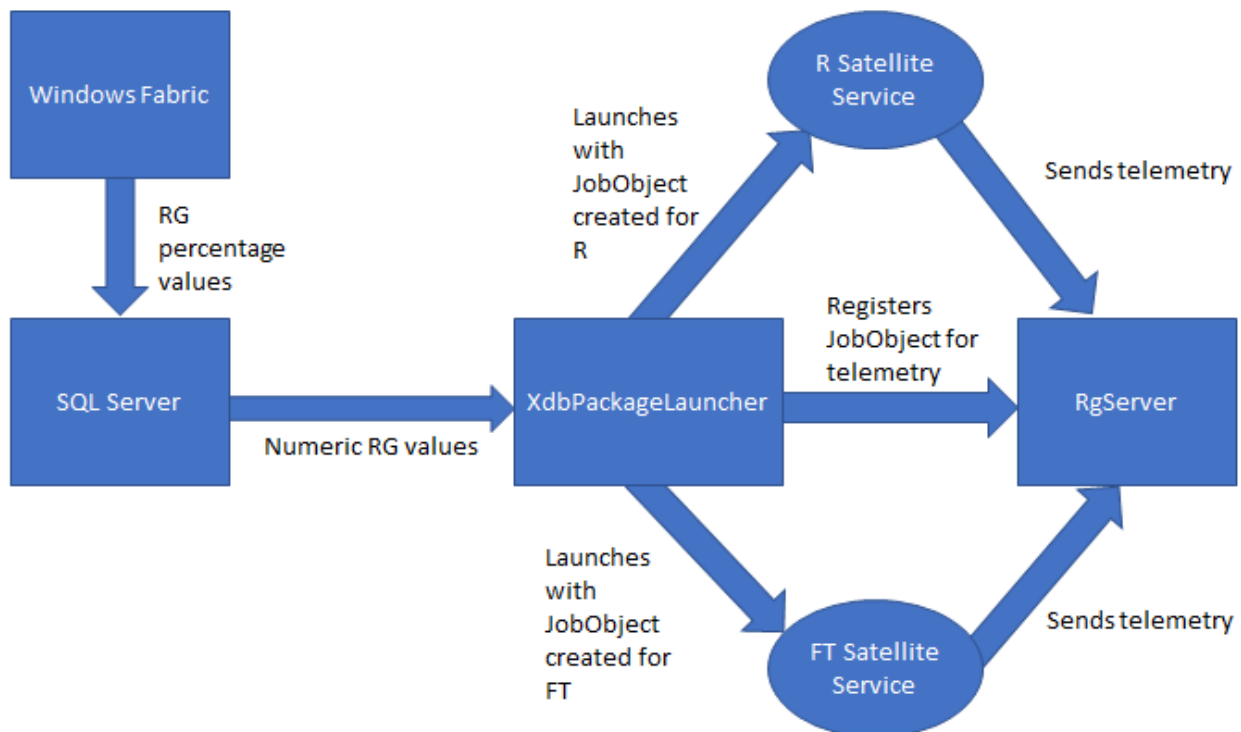
Currently, RG implementation for SQL DB would be reused for SQL MI.

- **Description**

Currently the resource governance for XdbPackageLauncher is defined as a set value at the code package level. The code package is given a percentage of SQL's memory and all the child processes that it creates inherit this resource governance property. This implementation of resource governance has some limitations that we are trying to overcome. There are three major issues that are being addressed by this design. The first issue is that we are overcommitting memory when we just give XdbPackageLauncher a percentage of SQL's memory. Second, XdbPackageLauncher won't be able to do anything if SQL is using all of its memory. And finally, there is no way to dynamically change the amount of memory used across different data packages/SKUs. The design tackles these problems and creates a new extensible implementation of resource governance at a data package level.

- **Design Details**

The main design change is that XdbPackageLauncher will be doing the resource governance instead of it happening at fabric level. This means that we will have resource governance at data package level in windows fabric for data packages being launched by XdbPackageLauncher. This helps us take care of the overcommitting problem because XdbPackageLauncher has access to JobObjects. XdbPackageLauncher will create one JobObject for each data package type that it launches. The resource governance values will be passed into XdbPackageLauncher from SQL as part of the LaunchDataPackage message.



- **Getting resource governance values SQL side**

The memory and cpu resource governance is a percentage of SQL's memory and SKUs cpu. To accomplish this, we created a new Naming Service property called "RgIsolationSettings." The property contains the resource governance information in xml format. SQL will get this value, parse the xml, convert to numeric values, and pass in the appropriate information to XdbPackageLauncher. SQL has to read the property from Windows Fabric so that it can convert the percentages to numeric values and pass them to XdbPackageLauncher. Having SQL read the resource governance allows us to shrink SQL if needed in order to run satellite services. The design for this still needs to be finalized but we are set up to solve the problem of satellite services not being able to run because SQL is using all the memory. Creating a property also allows us to use the existing infrastructure to have different values based on server edition and tier. The xml is constructed in a way so that in the future when other services are added, a tag can just be created with the resource governance for that service. The structure of the xml also allows us to have different resource governance per satellite process. Example of the xml is below.

```
<Size>P1</Size>    <R>    <memory>10</memory>    <cpu>20</cpu>    </R>
</RgIsolationSettings>
```

**- Using resource governance in XdbPackageLauncher** XdbPackageLauncher gets the numeric resource governance as part of the start message that it gets from SQL. The only thing that needs to be done is using those values with the JobObject of the specified satellite service and registering with RgServer. This means that the start message protocol needs to be edited so that it can include the new resource governance values in the message. Once that is done, XdbPackageLauncher just has to use these values to call two APIs and then launch the external process like it already does.

### Note:

For limited public preview, Extensibility would be disabled by default on all the MI instances (i.e., SQL gets 100% of memory and CPU resources). Customers interested in trying out extensibility would contact us, and Engineering Team will do the below steps using CAS:

- Enable Extensibility feature switch for the server.
- Run RgIsolationSettings CAS command to update the limits to 80 (SQL)- 20 (R/Python external processes). If the customer needs more for Python/R, we can set that as well.
- Do a restart

For the end to end work part of GA, We plan to update RgIsolationSettings from SQL and build a quorum safe restart workflow, which would be triggered from SQL.

## External Library Management for R and Python

Extensibility on SQL MI customers might often need to use multiple external libraries to work with. This could be more than the default set of external libraries provided with the runtime [R/Python runtime]. This would bring the External Library Management feature, which is already present in box, to Azure environment. Library Management feature becomes more prominent on Azure, as there is no other way for the customer to install the required external libraries. The external library management is the same as the existing model in box.

We use MSDB tables to store the local state of the library installs. But MSDB is replicated for Managed Instance (MI). To support external library management on MI instances, we would need to migrate the MSDB tablemsdb.dbo.external\_libraries\_installed to physical master. Physical master isn't replicated for BOX, SQL DB OR MI scenarios. So, this common solution would work for all the scenarios which are currently supported for external library management on Azure. This approach has been tried and verified for the library setup errors table and the same needs to be done for the table to track library installations. Since we already have

customers on Azure SQL DB using ML services, we would need to add an Upgrade Step to move the entries away from msdb into physical master

## Deployment in Azure

Packages for different components binaries

Apart from SQL server binaries the execution of R in Azure will require below three components and each of them will be represented by a package in azure terminology. In theory all the binaries can be stuffed in just one package but having separate packages for different component simplifies the servicing of components.

- a. Launchpad
- b. R related binaries
- c. Python related binaries.

DiskCleanupWatchdog to cleanup Vhds which are no longer in use on the node

When R/Python vhd version are updated, older vhds which are attached and older mounted folders would remain on the node, wasting disk space.

XdbPackageLauncher acquires a Global Mutex `Global\Vhd {R. PRODUCT_VERION_BUILD_VERSION}`

VHD hardlink creation and mounting is done inside this mutex by XdbPackageLauncher. To clean up the hard linked vhds under `c:\WFROOT\ext\` folder, We plan add a watchdog to go and delete the vhds from the folder. The watchdog would acquire the same mutex. The ReImageOS task which runs monthly would unmount the vhds from the nodes. Watchdog would be able to delete the vhds if they are mounted. XdbPackageLauncher would take the same mutex before creating a hardlink and mounting the vhds. This way all cases of race between Watchdog and XdbPackageLauncher are handled.

After re-imageOS and restart, the older vhds would be unmounted and only the vhd mounted folders would remain. If watchdog runs before XdbPackageLauncher, we will have the vhds deleted and XdbPackageLauncher would re-create the hardlinks and re-create the vhds for the current version of R and Python. If XdbPackageLauncher runs before watchdog, we would have the required vhds mounted, and watchdog would be able to clean up the unused vhds.

## SLOs supported

For the preview - This functionality is only supported in vCore-based purchasing models in Business Critical and General Purpose tiers. Existing databases should be migrated to the vCore based model before they can be enabled with Machine Learning Services. These databases can be migrated back out of the preview with no impact to your database. Only Gen 5 hardware will support Machine Learning Services with R and Python. These limits will be removed eventually

## How good have you found this content?

