Real-time scoring of R machine learning models on Azure

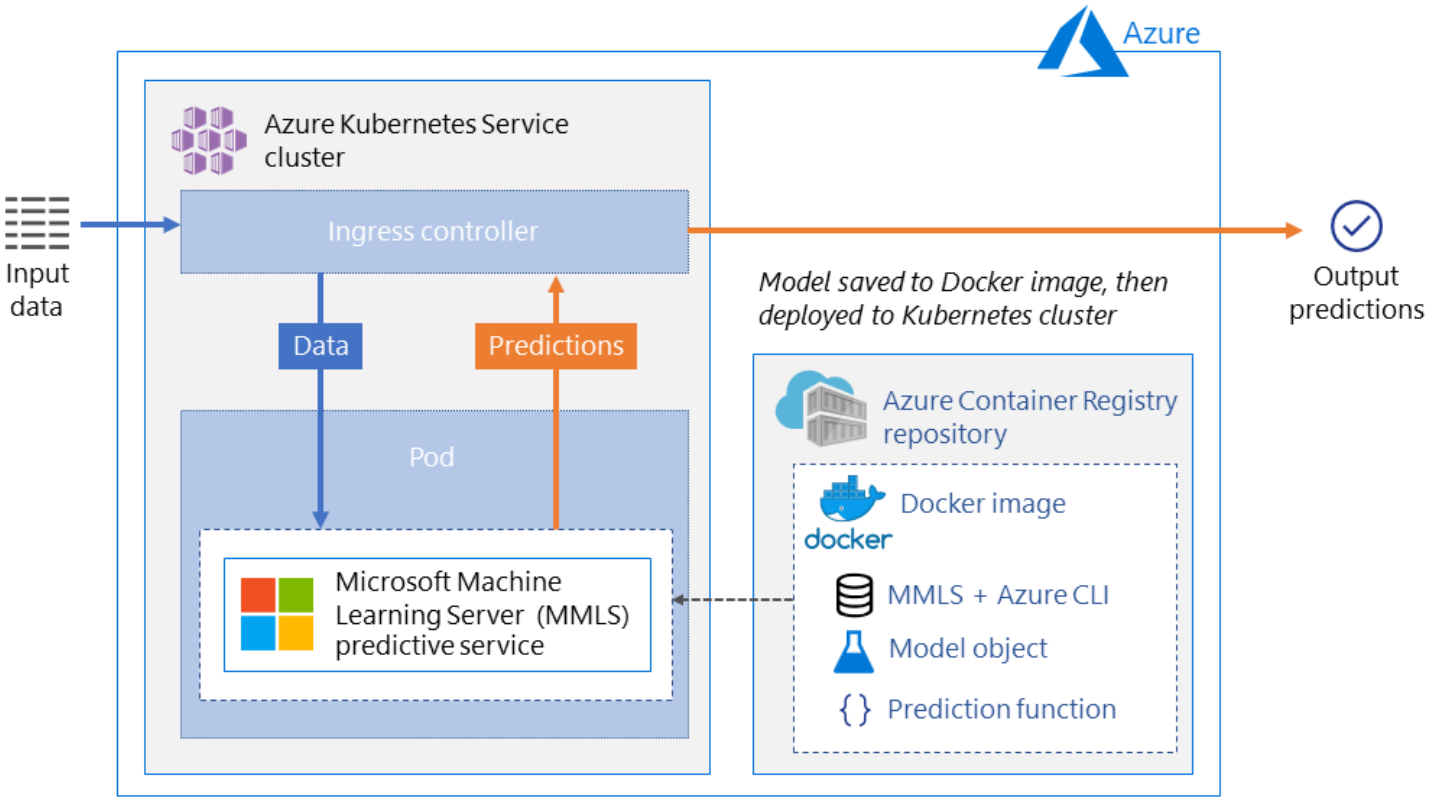
12/12/2018 • 5 minutes to read • Contributors

In this article

This reference architecture shows how to implement a real-time (synchronous) prediction service in R using Microsoft Machine Learning Server running in Azure Kubernetes Service (AKS). This architecture is intended to be generic and suited for any predictive model built in R that you want to deploy as a real-time service. **Deploy this solution**.

Azure Machine Learning alternative: This architecture provides a pure R experience. It doesn't use the Python oriented Azure Machine Learning Services (**AzureML SDK**), which is a mature cloud service for developing AI solutions at scale. AzureML SDK provides an easy path for development and deployment of containerized scoring scripts. We provide an alternative solution that shows how to use Conda to install R and R packages to leverage AzureML SDK via the rpy2 Python package. The value in using this alternative to operationalize R models is that you don't need to know Flask, and you can reuse AzureML SDK expertise, which can be useful for teams that are comfortable using both R and Python languages for their data science projects. The implementation of this alternative architecture is available on GitHub.

Architecture



This reference architecture takes a container-based approach. A Docker image is built containing R, as well as the various artifacts needed to score new data. These include the model object itself and a scoring script. This image is pushed to a Docker registry hosted in Azure, and then deployed to a Kubernetes cluster, also in Azure.

The architecture of this workflow includes the following components.

- **Azure Container Registry** is used to store the images for this workflow. Registries created with Container Registry can be managed via the standard Docker Registry V2 API and client.
- **Azure Kubernetes Service** is used to host the deployment and service. Clusters created with AKS can be managed using the standard Kubernetes API and client (kubectl).
- **Microsoft Machine Learning Server** is used to define the REST API for the service and includes Model Operationalization. This service-oriented web server process listens for requests, which are then handed off to other background processes that run the actual R code to generate the results. All these processes run on a single node in this configuration, which is wrapped in a container. For details about using this service outside a dev or test environment, contact your Microsoft representative.

Performance considerations

Machine learning workloads tend to be compute-intensive, both when training and when scoring new data. As a rule of thumb, try not to run more than one scoring process per core. Machine Learning Server lets you define the number of R processes running in each container. The default is five processes. When creating a relatively simple model, such as a linear regression with a small number of variables, or a small decision tree, you can increase the number of processes. Monitor the CPU load on your cluster nodes to determine the appropriate limit on the number of containers.

A GPU-enabled cluster can speed up some types of workloads, and deep learning models in particular. Not all workloads can take advantage of GPUs — only those that make heavy use of matrix algebra. For example, tree-based models, including random forests and boosting models, generally derive no advantage from GPUs.

Some model types such as random forests are massively parallelizable on CPUs. In these cases, speed up the scoring of a single request by distributing the workload across multiple cores. However, doing so reduces your capacity to handle multiple scoring requests given a fixed cluster size.

In general, open-source R models store all their data in memory, so ensure that your nodes have enough memory to accommodate the processes you plan to run concurrently. If you are using Machine Learning Server to fit your models, use the libraries that can process data on disk, rather than reading it all into memory. This can help reduce memory requirements significantly. Regardless of whether you use Machine Learning Server or open-source R, monitor your nodes to ensure that your scoring processes are not memory-starved.

Security considerations

Network encryption

In this reference architecture, HTTPS is enabled for communication with the cluster, and a staging certificate from Let's Encrypt is used. For production purposes, substitute your own certificate from an appropriate signing authority.

Authentication and authorization

Machine Learning Server Model Operationalization requires scoring requests to be authenticated. In this deployment, a username and password are used. In an enterprise setting, you can enable authentication using Azure Active Directory or create a separate front end using Azure API Management.

For Model Operationalization to work correctly with Machine Learning Server on containers, you must install a JSON Web Token (JWT) certificate. This deployment uses a certificate supplied by Microsoft. In a production setting, supply

your own.

For traffic between Container Registry and AKS, consider enabling role-based access control (RBAC) to limit access privileges to only those needed.

Separate storage

This reference architecture bundles the application (R) and the data (model object and scoring script) into a single image. In some cases, it may be beneficial to separate these. You can place the model data and code into Azure blob or file storage, and retrieve them at container initialization. In this case, ensure that the storage account is set to allow authenticated access only and require HTTPS.

Monitoring and logging considerations

Use the Kubernetes dashboard to monitor the overall status of your AKS cluster. See the cluster's overview blade in Azure portal for more details. The GitHub resources also show how to bring up the dashboard from R.

Although the dashboard gives you a view of the overall health of your cluster, it's also important to track the status of individual containers. To do this, enable Azure Monitor Insights from the cluster overview blade in Azure portal, or see Azure Monitor for containers (in preview).

Cost considerations

Machine Learning Server is licensed on a per-core basis, and all the cores in the cluster that will run Machine Learning Server count towards this. If you are an enterprise Machine Learning Server or Microsoft SQL Server customer, contact your Microsoft representative for pricing details.

An open-source alternative to Machine Learning Server is Plumber, an R package that turns your code into a REST API. Plumber is less fully featured than Machine Learning Server. For example, by default it doesn't include any features that provide request authentication. If you use Plumber, it's recommended that you enable Azure API Management to handle authentication details.

Besides licensing, the main cost consideration is the Kubernetes cluster's compute resources. The cluster must be large enough to handle the expected request volume at peak times, but this approach leaves resources idle at other times. To limit the impact of idle resources, enable the horizontal autoscaler for the cluster using the kubectl tool. Or use the AKS cluster autoscaler.

Deploy the solution

The reference implementation of this architecture is available on GitHub. Follow the steps described there to deploy a simple predictive model as a service.