Automate scaling of Apache Spark pools in Azure Synapse Analytics

Note:

You are not required to complete the processes, tasks, activities, or steps presented in this example. The various samples provided are for illustrative purposes only and it's likely that if you try this out you will encounter issues in your system.

Within an Apache Spark Pool, it is possible to configure a fixed size when you disable autoscaling. When you enable autoscale, you can set a minimum and maximum number of nodes in order to control the scale that you'd like.

Once you have enabled autoscale, Azure Synapse Analytics will monitor the resources of the load, and it will scale the number of nodes up or down. When it comes to the metrics involved in making a decision to scale up or down then there will be continuous monitoring depending on:

- CPU usage
- pending memory
- free CPU
- free memory
- and the used memory per node

It checks these metrics every 30 seconds and makes scaling decisions based on the values. There's no extra charge for this feature.

Below is a table that shows the metrics that autoscale enablement of the Apache Spark Pool within Azure Synapse analytics instance checks and collects:

Metric	Description
Total Pending CPU	The total number of cores required to start execution of all pending nodes.
Total Pending Memory	The total memory (in MB) required to start execution of all pending nodes.
Total Free CPU	The sum of all unused cores on the active nodes.
Total Free Memory	The sum of unused memory (in MB) on the active nodes.
Used Memory per Node	The load on a node. A node on which 10 GB of memory is used is considered to be under more load than a worker with 2GB of used memory

When we look at load-based scale conditions, the autoscale functionality will issue a scale request based on the metrics outlined in the table below:

Scale-up	Scale-down
Total pending CPU is greater than total free CPU for more than 1 minute.	Total pending CPU is less than total free CPU for more than 2 minutes.
Total pending memory is greater than total free memory for more than 1 minute.	Total pending memory is less than total free memory for more than 2 minutes.

When autoscale scales up, it will calculate the number of new nodes that would be needed in order to meet the CPU and memory requirements. Next, it will issue the scale-up requests and add the number of nodes required to do the job.

In case autoscale performs the action of scaling down, the decision is based on the number of executors, the application primaries per node, and the CPU and memory requirements. The autoscale functionality will then issue the request to remove some nodes.

The autoscale functionality will also check which nodes are candidates for removal based on the current job execution. The scale down operation first decommissions the nodes and then removes them from the cluster.

If you would like to get started with the autoscale functionality, then follow these steps:

Create a serverless Apache Spark pool with Autoscaling

To enable the Autoscale feature, complete the following steps as part of the normal Apache Spark pool creation process:

1. On the **Basics** tab, select the **Enable autoscale** checkbox.

2. Enter the desired values for the following properties:

- **Min** number of nodes.
- **Max** number of nodes.

The initial number of nodes will be the minimum. This value defines the initial size of the instance when it's created. The minimum number of nodes can't be fewer than three.

When considering the best practices to use for the autoscale feature, consider latency as part of the scale up or down operations. It could take 1 to 5 minutes in order for the scaling operations (whether that's scaling up or down) to complete.

Also, when you scale down, the nodes will first be put in a decommissioned state such that there won't be new executors launching on the node. The jobs that are still running will continue to run and finish. However, the pending jobs will be in a waiting state to be scheduled as normal but with fewer nodes.