How to auto-scale Azure SQL Databases using Runbooks and Alerts



Prepared by Julio Calderón

Databases SQL Ninja Engineering Team ([datasqlninja@microsoft.com](mailto:datasqlninja@microsoft.com))

**Disclaimer**

The High-Level Architecture, Migration Dispositions and guidelines in this document is developed in consultation and collaboration with Microsoft Corporation technical architects.  Because Microsoft must respond to changing market conditions, this document should not be interpreted as an invitation to contract or a commitment on the part of Microsoft.

Microsoft has provided generic high-level guidance in this document with the understanding that MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THE INFORMATION CONTAINED HEREIN.

This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2019 Microsoft. All rights reserved.

**Note**: The detail provided in this document has been harvested as part of a customer engagement sponsored through the [Databases SQL Ninja Engineering](https://aka.ms/dpn-contents).

Table of Contents

[1 Introduction 5](#_Toc11766963)

[2 General Guidelines 6](#_Toc11766964)

[3 Empirical Overview 7](#_Toc11766965)

[4 Configurations 8](#_Toc11766966)

[6 Feedback and suggestions 10](#_Toc11766971)

1. Introduction

Azure SQL Database offers an easy several-clicks solution to scale database instances when more resources are needed. This is one of the strengths of PaaS, you pay for only what you use and if you need more or less, it's easy to make the change. A current limitation, however, is that the scaling operation is a manual one. Some service tiers don’t support auto-scaling as some of us would expect.

Azure SQL offers a tier which auto-scales depending on its usage pattern called [Azure SQL Database serverless](https://learn.microsoft.com/en-us/azure/azure-sql/database/serverless-tier-overview?view=azuresql). This is a great option to take into account if your current workload uses the General Purpose standard architecture or Hyperscale. Having said that, serverless does not offer Business Critical levels of performance or tiers such as Memory Optimized and DC-series.

Using the power of Azure we can set up a workflow that auto-scales an Azure SQL Database instance to the next immediate tier when a specific condition is met. For example: what if you could auto-scale the database as soon as it goes over 85% CPU usage for a sustained period of 5 minutes? This whitepaper walks through an option that uses Automation runbooks and Azure Monitor Alerts to achieve that.

In addition, the script developed here provides an algorithm useful to scale up and down at specific times during the day. The runbook can be scheduled to run and hardcoded values for a specific database can be set to achieve the same results.

1. General Guidelines

**Supported SKUs**: this script supports DTU, General Purpose/Business Critical provisioned Gen 5 compute and Hyperscale Gen 5 standard series compute. Other tiers such as Hyperscale Premium/Memory-optimized series, Fsv2 and DC series are not supported. Keep in mind these are not supported in the script itself, but the logic presented below can scale any tier given the tier’s names are hardcoded in the arrays provided in the script. This is, you can refer to [this article](https://learn.microsoft.com/en-us/azure/azure-sql/database/resource-limits-vcore-single-databases?view=azuresql#hyperscale---premium-series-memory-optimized-part-2-of-3) for your specific SKU and modify the script (i.e. Gen5 array) to suit it to that tier.

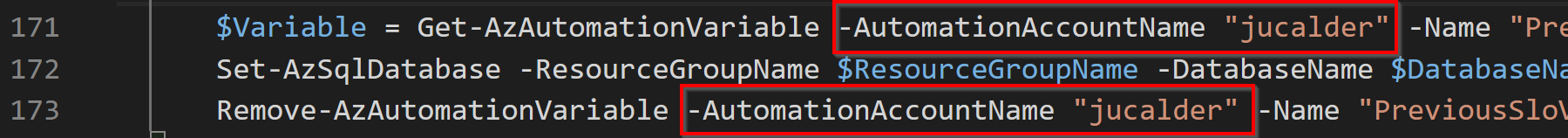
|  |
| --- |
| ***Important****: every time any part of the setup asks if the****Common Alert Schema (CAS)****should be enabled, select****Yes****. The script used in this tutorial assumes the CAS will be used for the alerts triggering it.*    Text  Description automatically generated |

This script also offers some initial modifications, explained below:

* Line 128 is an artificial time delay. When the alert is triggered the first time because the condition has been met, it performs the scale up operation as expected. When the condition is no longer met (imagine CPU is no longer over 50% which was the condition initially), the alert will be automatically marked as resolved and this runs the runbook again, with the difference that this time, it will go into the second part of the script that scales down.

It might be desired to *not* scale down as soon as the alert is resolved, since a scale up will reset the resource usage and there’s a risk of a condition not being met just by virtue of the time taken since the scale up ends. In this case, this artificial time limit can be used to choose how much time to wait before the scale down occurs (if at all) after the alert has been marked as resolved.

* There are 3 Boolean flags on lines 8-10 which allow to select which scale down operation to use: only scale up, only scale down or scale both up and down.
* The script contains 2 ways of scaling down.
  + #1 involves inverting the logic of the scale up operation, and is included (and commented by default) in between lines 130 -> 169. This is commented by default since the main objective of the article is to provide an automatic scale up and down mechanism in the simplest way possible. But its acknowledged that this article can also be used to scale up and down on a **schedule** and thus it might be needed to separate the scale up and down logics into separate script. The scale down inverted logic commented in between those lines is provided as additional logic in case it’s needed.
  + #2 is the main way of scaling down, which involves the scale up operation creating a variable during the first run on lines 88 and 116 and then using that variable in lines 171 -> 173.
  + #1 and #2 are interchangeable and will achieve the same result. The only other thing to consider for #2 is that the Automation Account’s name must be hardcoded into the commands on lines 88, 116, 171 and 173 because it can’t be programmatically obtained.



Empirical Overview

Diagram, schematic

Description automatically generated

SQL Database triggers an alert based on a specific condition, then the alert triggers a runbook and passes to it a webhook with data from the SQL Database that fired it, which allows the runbook to connect using its Managed Service Identity to Azure Resource Manager and use the webhook’s data to read the metadata from the SQL Database and decide which tier it should be scaled up or down to.

1. Configurations

**Step #1: deploy Azure Automation account**

The scale operation will be executed by a PowerShell runbook inside of an Azure Automation account. Search *Automation*in the Azure Portal search bar and create a new Automation account. Make sure to create a **Managed Identity**for the automation account while doing this:

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

After the automation account is created, add the system assigned managed identity as contributor on the server, so the scripts get the permissions to scale up and down the databases in a particular server.

Graphical user interface

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Step #2: create scaling runbook**

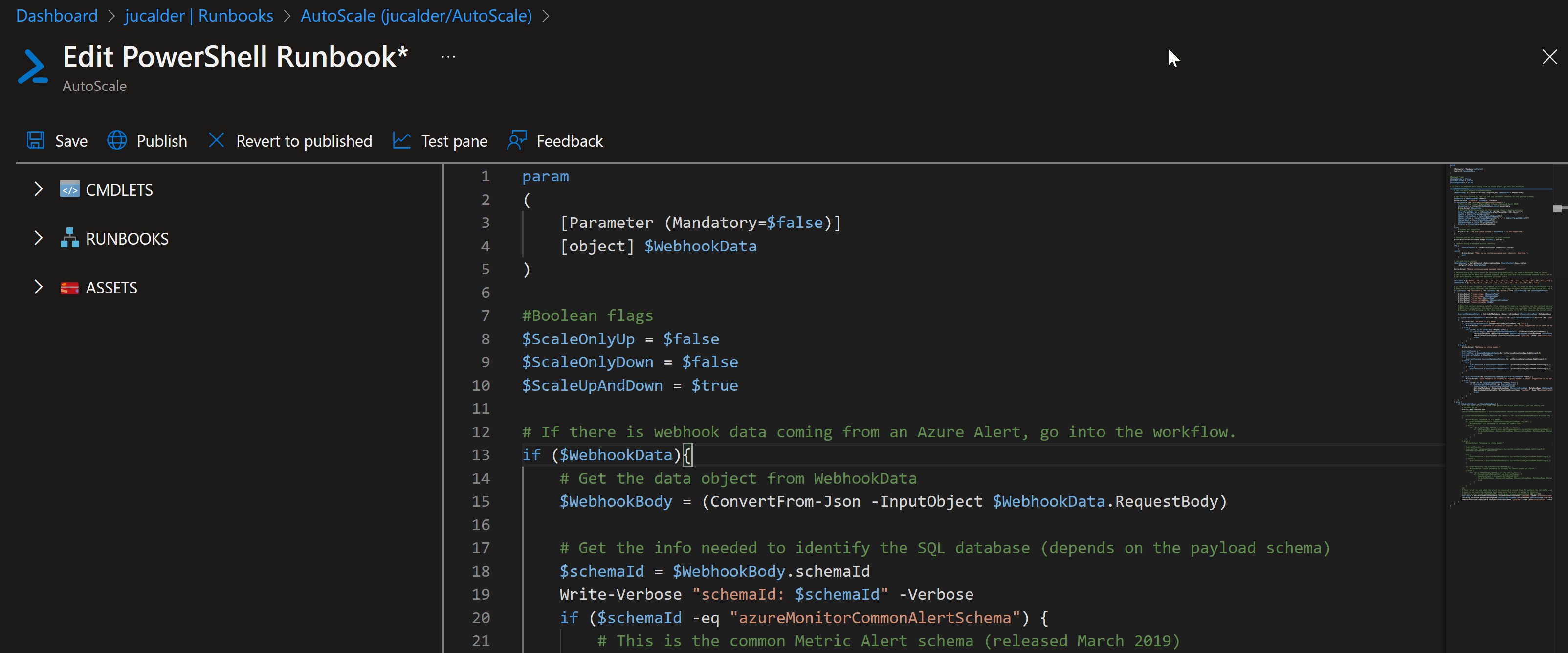
With our Automation account deployed and with rights to our Azure SQL Server, we are now ready to create the script. Create a new runbook and copy the code below:

Graphical user interface, application

Description automatically generated

The script attached uses Webhook data passed from the alert. This data contains useful information about the resource the alert gets triggered from, which means the script can auto-scale any database and no parameters are needed; it only needs to be called from an alert using the Common Alert Schema on an Azure SQL database.

Copy the contents of the script file attached and paste it into the new runbook, save it and click **Publish**.



****

**Step #3: create Azure Monitor Alert to trigger the Automation runbook**

On your Azure SQL Database, create a new alert rule:

Graphical user interface, text, application, chat or text message

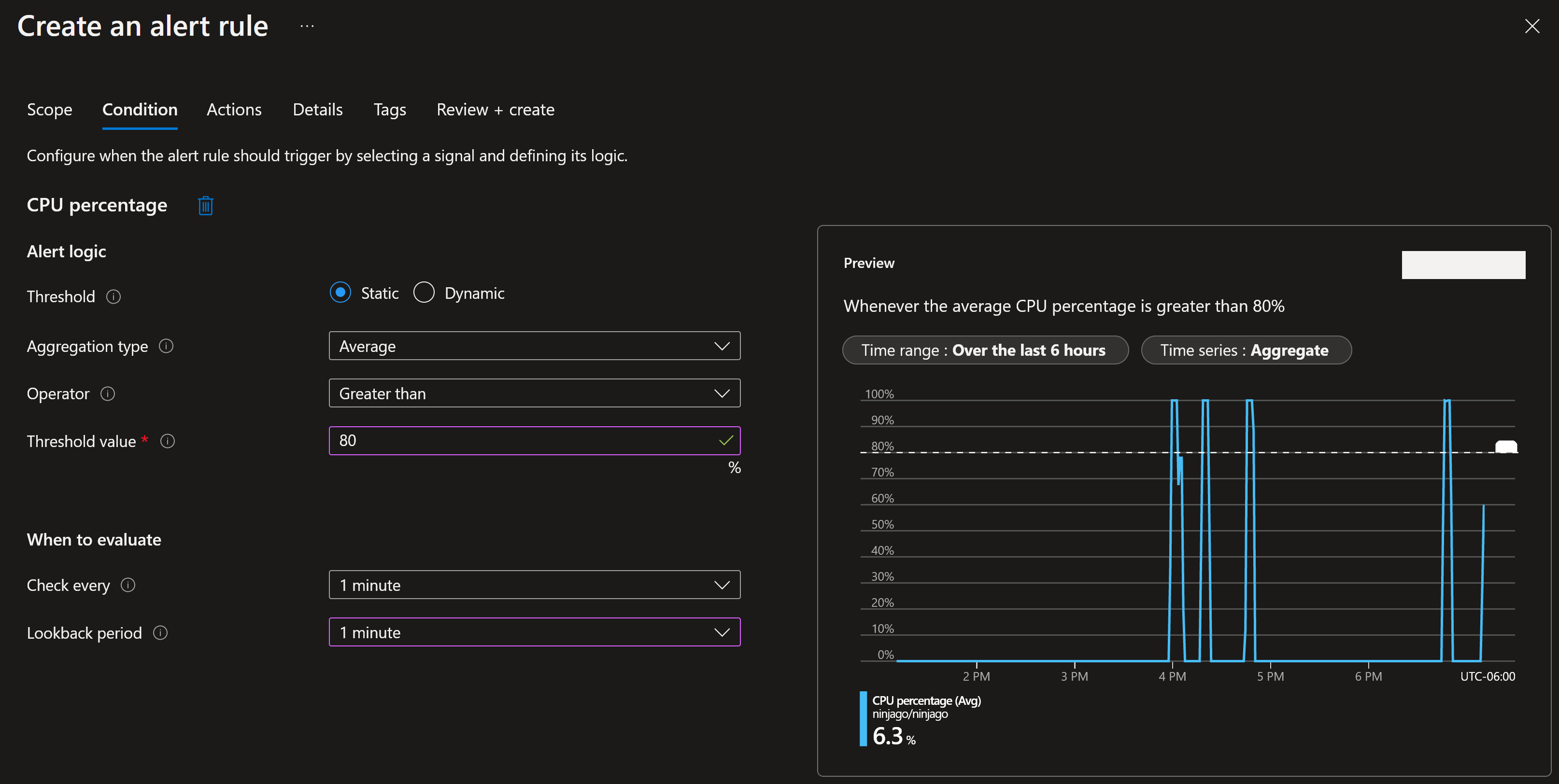
Description automatically generated

The next blade will require several different setups:

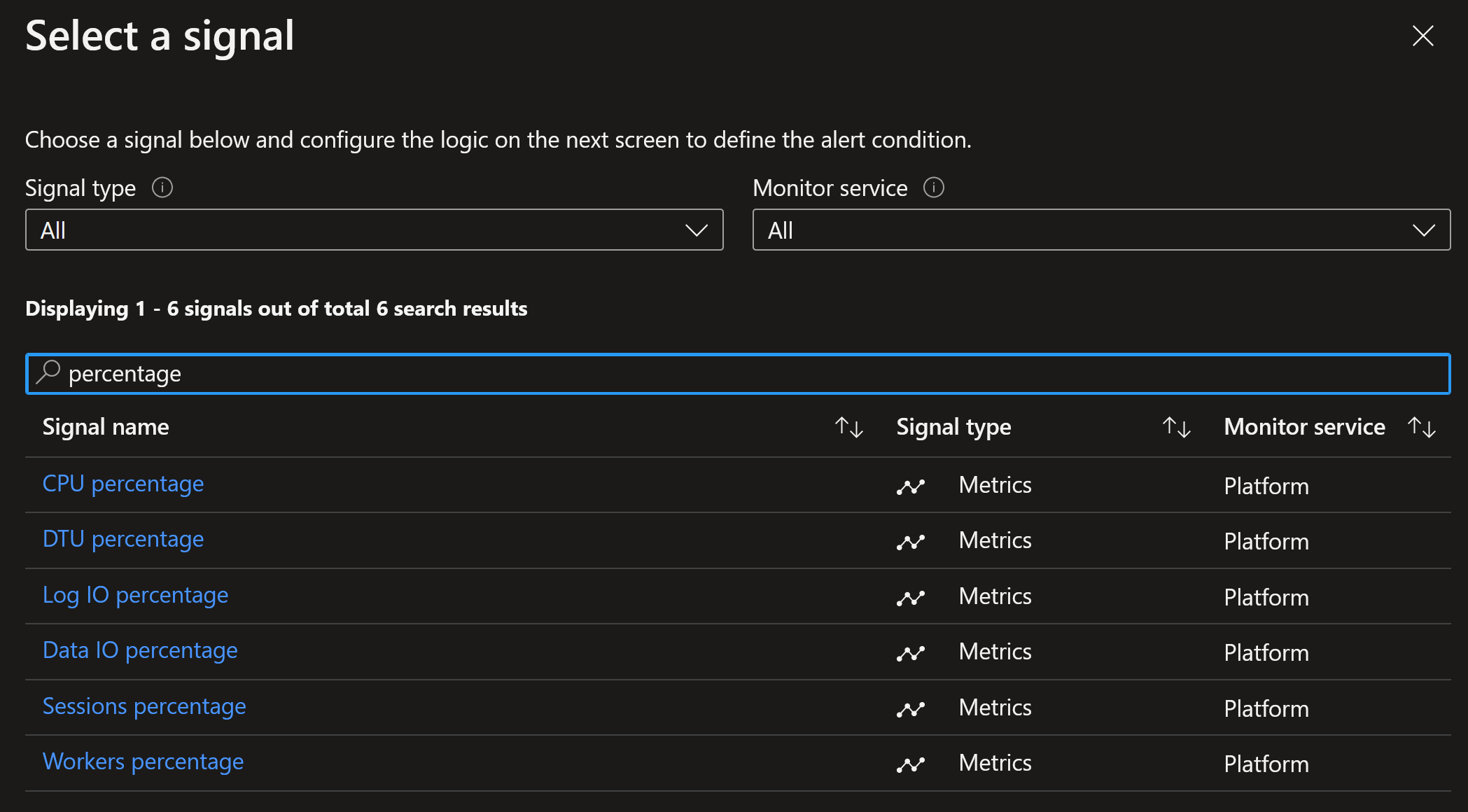
1. **Scope of the alert**: this will be auto populated if **+New Alert Rule**was clicked from within the database itself.
2. **Condition**: when should the alert get triggered by selecting a signal and defining its logic.
3. **Actions**: when the alert gets triggered, what will happen?

**Condition**

For this example, the alert will monitor the CPU consumption every 1 minute. When the average goes over 85%, the alert will be triggered:



Other conditions can also be used, like for example Log IO or workers percentage, that can help monitor parallelism impact:



**Actions**

Text

Description automatically generated

After the signal logic is created, we need to tell the alert what to do when it gets fired. We will do this with an **action group**. When creating a new action group, two tabs will help us configure sending an email and triggering the runbook:

***Notifications***

A screenshot of a computer

Description automatically generated with medium confidence

***Actions***

Select your subscription, automation account and runbook previously created:

A screenshot of a computer

Description automatically generated with medium confidence

After saving the action group, add the remaining details to the alert.

That's it! The alert is now enabled and will auto-scale the database when fired. The runbook will be executed twice per alert: once when fired and another when resolved, and depending on the Boolean flag chosen will scale up, down or both.

1. Feedback and suggestions

If you have feedback or suggestions for improving this data migration asset, please contact the Databases SQL Ninja Engineering Team ([datasqlninja@microsoft.com](mailto:datasqlninja@microsoft.com)). Thanks for your support!

Note: For additional information about migrating various source databases to Azure, see the [Azure Database Migration Guide](https://datamigration.microsoft.com/).