

Describe performance monitoring

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/describe-performance-monitoring/1-introduction>

Introduction

Completed

A major part of the job of a database administrator is proper performance monitoring. This task doesn't change when moving to a cloud platform. While Azure offers tools for monitoring, you may lack some specific controls around hardware that you would have in an on-premises environment which makes understanding how to identify and resolve performance bottlenecks while in Azure SQL that is much more critical.

Learning objectives

- Understanding methods to review potential performance issues
- Identify critical Azure metrics
- Learn how to collect metrics for an established baseline
- Use extended events for performance analysis
- Understand database watcher

2. Describe performance monitoring tools

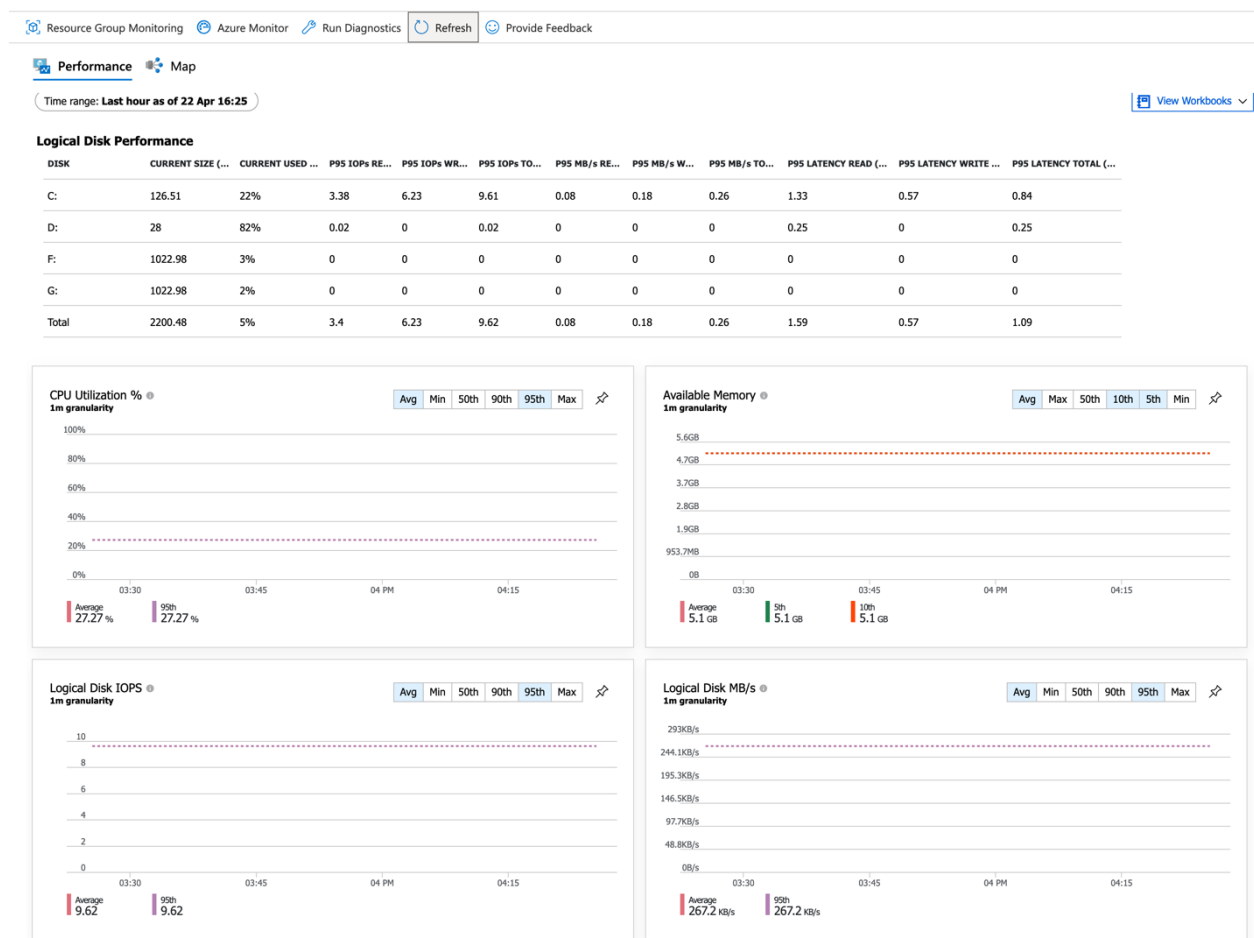
<https://learn.microsoft.com/en-us/training/modules/describe-performance-monitoring/2-describe-performance-monitoring-tools>

Describe performance monitoring tools

Completed

Azure provides multiple methods to monitor the performance of your resources and create a baseline. Each method can be tailored for a specific metric. The metrics that you can monitor vary depending on the type of Azure resource you're monitoring. For example, Azure SQL Database and SQL Server on an Azure Virtual Machine have different metrics available in the Azure portal.

The following examples are focused on an Azure Virtual Machine. When you deploy an Azure Virtual Machine from Azure Marketplace, an agent is installed in the virtual machine that provides a basic set of operating system metrics that are presented to you in the Azure portal. This agent supplies metrics to a service called Azure Monitor, which is a comprehensive platform monitoring solution that collects and displays a standard set of metrics from Azure resources. If you're using a virtual machine, the default metrics captured include CPU usage, network utilization, and disk read and write operations. You can capture extra metrics beyond what is captured in Azure monitor by enabling Monitoring Insights for your virtual machine as shown in the following image.



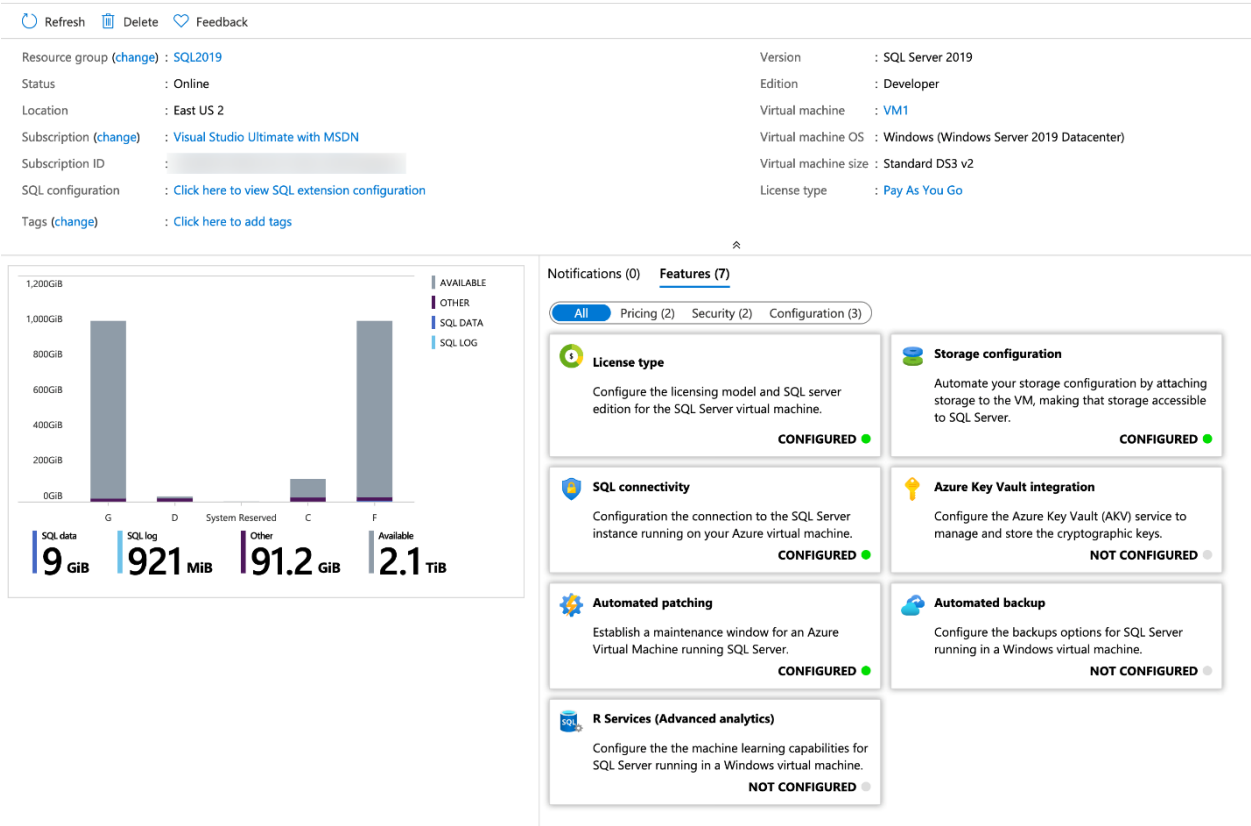
These metrics pertain to the operating system, not SQL Server. Notice that the namespace for each metric is the virtual machine host, not SQL Server.

You're unable to view SQL Server-specific metrics from within the portal. For detailed SQL Server-specific metrics, you need to gather them from the virtual machine itself.

Azure Monitoring Insights allows you to collect extra data points like storage latency, available memory, and disk capacity. These Azure Monitor Insights can be one way of viewing a baseline of performance for your Azure Virtual Machine including I/O performance, memory, and CPU utilization.

This data is stored in an Azure Log Analytics workspace. Azure Log Analytics is the primary tool in Azure for storing and querying log files of all kinds. Log Analytics is queried by a SQL-like language called Kusto Query Language (KQL).

If you create a virtual machine with one of the preconfigured SQL Server images in Azure Marketplace, you can also get the SQL virtual machine resource provider as shown in the following image.



You can launch this screen in the Azure portal by navigating to the **Settings** section of the main blade for an Azure Virtual Machine, and then selecting the **SQL Server configuration** option.



Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Networking

Connect

Disks

Size

Security

Advisor recommendations

Extensions

Continuous delivery

Availability + scaling

Configuration

Identity

SQL Server configuration

Properties

Locks

Operations

Bastion

Auto-shutdown

SQL management experience on Virtual Machines

The new SQL focused management experience provides a single view of all your [Virtual Machines running SQL Server](#). You can manage your SQL Virtual Machines with features like automated patching, automated backup, licensing and edition flexibility.

Earlier SQL manageability was offered for only SQL Server Azure marketplace images, but you can now register any Azure virtual machine, with SQL Server installed, with the [SQL VM Resource provider](#) and unlock all manageability features.

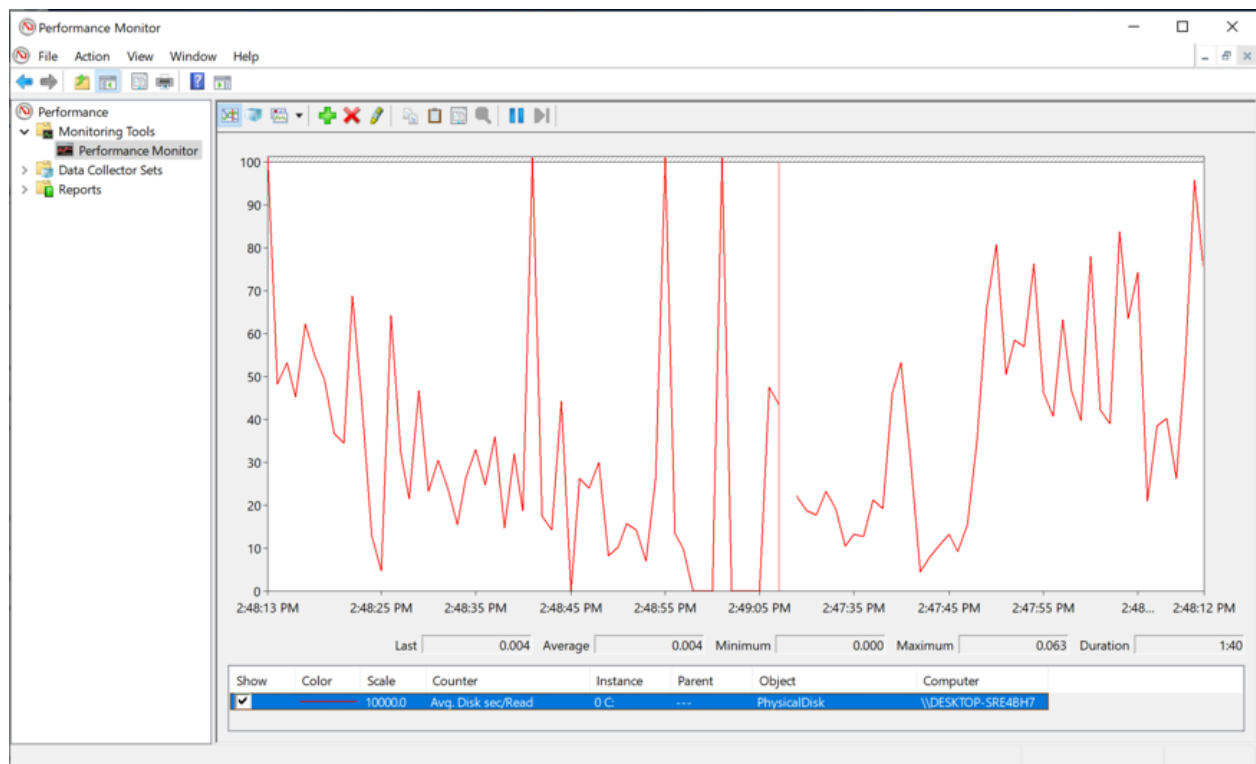
All upcoming manageability features and improvements will only be made available through this new experience.

[Manage SQL virtual machine](#)

This dashboard allows you to see how much space your database files and transaction log file are consuming, and allows you to manage the features provided by the resource provider like automated patching and storage configuration. You can manually install the SQL Resource Provider for other installations of SQL Server on Azure Virtual Machine that weren't defined as part of the virtual machine.

Performance Monitor with SQL Server on an Azure Virtual Machine

Whether you're using an on-premises server or on an Azure Virtual Machine, the Windows Server platform has a native tool called Performance Monitor (commonly shortened to *perfmon* after the name of its executable file) that allows you to easily and routinely monitor performance metrics. *Perfmon* operates with counters for both the operating systems and installed programs. When SQL Server is installed on the operating system, the database engine creates its own group of specific counters.



The image shows the reporting interface of Performance Monitor, with a single counter being collected. This screen is reached from launching Performance Monitor in Windows and shows a live tracker of a specific performance counter. In many cases, you capture multiple counters in the same session. *Perfmon* data can be stored locally and analyzed, but in larger environments you can forward performance monitor results into Azure Monitor, where you can have a single view across many servers.

3. Describe critical performance metrics

<https://learn.microsoft.com/en-us/training/modules/describe-performance-monitoring/3-describe-critical-performance-metrics>

Describe critical performance metrics

Completed

Let's learn how to create metrics in Azure Monitor, which allow you to trigger alerts or execute automated error responses.

Review of Azure metrics

The Azure Monitor service includes the ability to track various metrics about the overall health of a given resource. Metrics are gathered at regular intervals and are the gateway for alerting processes that help to resolve issues quickly and efficiently. Azure Monitor Metrics is a powerful subsystem that allows you to not only analyze and visualize your performance data, but to also trigger alerts that notify administrators or automated actions that can trigger an Azure Automation runbook or a webhook. You can also archive your Azure Metrics data to Azure Storage, since active data is only stored for 93 days.

Create metric alerts

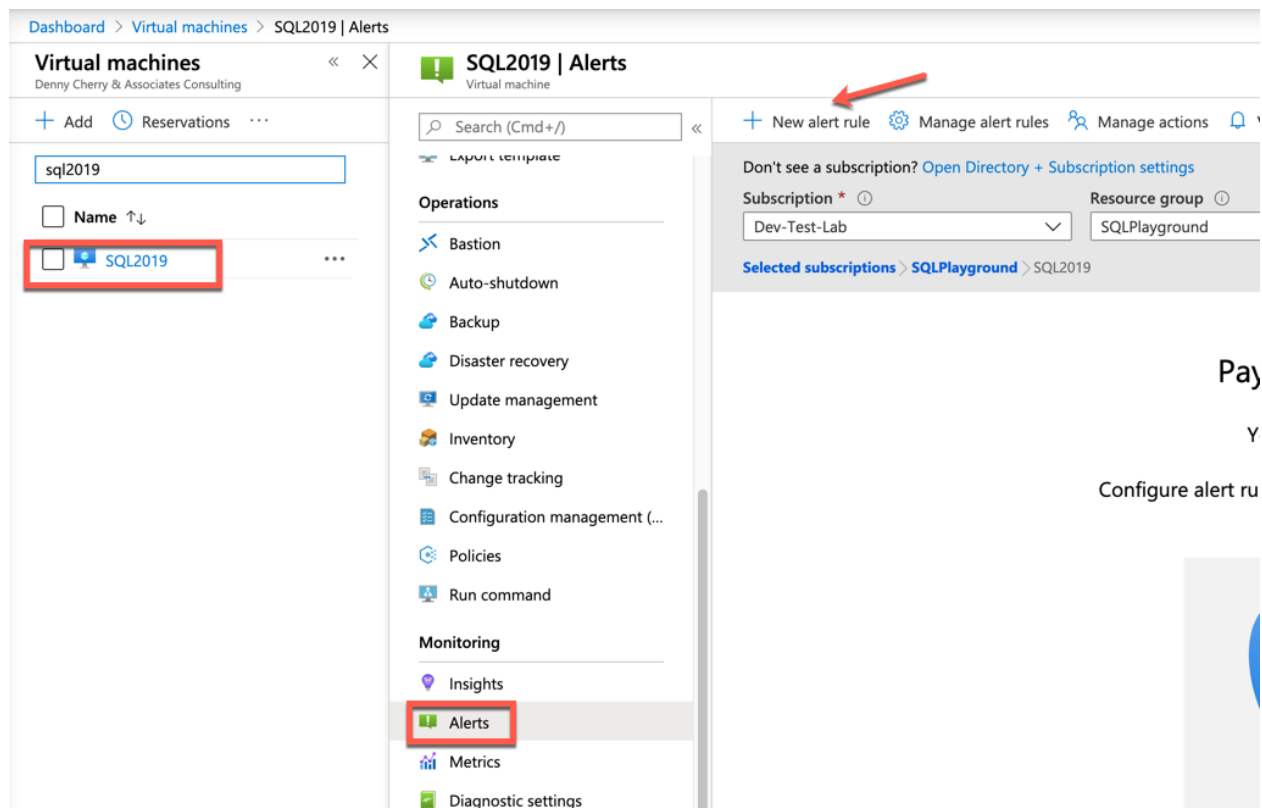
Utilizing the Azure portal, you can create alert rules, based on defined metrics, in the overview section of the Azure Monitor blade. Azure Monitor Alerts can be scoped in three ways. For example, using Azure Virtual Machines as an example you can specify the scope as:

- A list of virtual machines in one Azure region within a subscription
- All virtual machines (in one Azure region) in one or more resource groups in a subscription
- All virtual machines (in one Azure region) in one subscription

In this manner, you can create an alert rule based on resources contained within resource groups as shown.

The screenshot shows the Azure portal interface for the 'Monitor | Alerts' section. The left-hand navigation pane has the 'Monitor' tab highlighted. The main area shows the 'Monitor | Alerts' blade with a search bar and several action links: '+ New alert rule', 'Manage alert rules', 'Manage actions', 'View classic alerts', and 'Refresh'. A red arrow points to the '+ New alert rule' button. Below these links, there are dropdown menus for 'Subscription' (currently set to 'Dev-Test-Lab') and 'Resource group' (currently set to '33 selected'). A message at the bottom of the blade states 'All is good! You have 1 alert rule.' and a link 'Manage alert rules (2)' is visible.

The following example demonstrates creating an alert for a virtual machine named *SQL2019*, focusing on the scope of the individual virtual machine.



Regardless the scope of the alert, the creation process is the same.

From the alerts screen, select **New Alert Rule**. If an alert is created from within the scope of a resource, the resource values should be populated for you. You can see that the resource is the *SQL2019* virtual machine, the subscription is *Dev-Test-Lab* and the resource group in which it resides is *SQLPlayground*.

Under the Condition section, select **Add**:

Dashboard > Virtual machines > SQL2019 | Alerts > Create rule

Create rule

Rules management

Configure signal logic

Choose a signal below and configure the logic on the next screen to define the alert condition.

Signal type ⓘ

All

Monitor service ⓘ

All

Displaying 1 - 20 signals out of total 50 signals

Search by signal name

Signal name	↑↓	Signal type	↑↓	Monitor service	↑↓
Percentage CPU		 Metric		Platform	
Network In Billable (Deprecated)		 Metric		Platform	
Network Out Billable (Deprecated)		 Metric		Platform	
Disk Read Bytes		 Metric		Platform	
Disk Write Bytes		 Metric		Platform	
Disk Read Operations/Sec		 Metric		Platform	
Disk Write Operations/Sec		 Metric		Platform	

The alerts can be configured in a static manner (for example, raise an alert when CPU goes over 95%) or in a dynamic fashion using Dynamic Thresholds. Dynamic Thresholds learn the historical behavior of the metric and raise an alert when the resources are operating in an abnormal manner. These Dynamic Thresholds can detect seasonality in your workloads and adjust the alerting accordingly.

If Static alerts are used, you must provide a threshold for the selected metric. In this example, 80 percent was specified. This threshold means that if the CPU utilization exceeds 80 percentage over a given period, an alert is fired and reacts as specified.

Both types of alerts offer Booleans operators such as the 'greater than' or 'less than' operators. Along with Boolean operators, there are aggregate measurements to select from such as average, minimum, maximum, count, average, and total. With these options available, it's easy to construct a flexible alert that will suit just about any enterprise level alerting.

Configure signal logic

Percentage CPU (Avg)
sql2019

Alert logic

Threshold ⓘ

Static Dynamic

Operator ⓘ

Greater than

Aggregation type * ⓘ

Average

Threshold value * ⓘ

80 ✓

%

Condition preview

Whenever the percentage cpu is greaterthan 80 %

Evaluated based on

Aggregation granularity (Period) * ⓘ

5 minutes

Frequency of evaluation ⓘ

Every 1 Minute

Done

After you create the alert, in order to notify administrators or launch an automation process, an action group needs to be configured.

Defining an action group is optional, and if one isn't configured the alert will just log the notification to storage with no further action. You can create a new action group from the metrics screen, by selecting **Add** next to Action Groups.

Select an action group to attach to this alert rule

The action group selected will attach to this alert rule

+ Create action group

Subscription Create action group

Contoso Ltd

Once you select **Create Action Group**, you see the following screen. You name the action group and define an alert and the response. In this example, the administrator receives an email notification if

the alert condition is triggered.

Add action group

×

Action group name * ⓘ

CPU Alert for SQL1

✓

Short name * ⓘ

CPUSQL1

✓

Subscription * ⓘ

Contoso Ltd

▼

Resource group * ⓘ

Default-ActivityLogAlerts (to be created)

▼

Actions

Action name *	Action Type *	Status	Configure	Actions
EmailOperator ✓	Email/SMS message/P... ▼		Edit details	×
Unique name for the action	Select an action type ▼			

[Azure Privacy Statement](#)

[Azure Alerts Pricing](#)

ⓘ

Have a consistent format in emails, notifications and other endpoints irrespective of monitoring source. You can enable per action by editing details. Click on the banner to learn more [↗](#)

You can configure the email or SMS details by selecting **Edit Details** under **Configure**, or by adding a new action, which will also bring up the configuration screen.

Email/SMS message/Push/Voice



Add or edit an Email/SMS/Push/Voice action

☒ Email

Email *

☐ SMS (Carrier charges may apply)

Country code

Phone number

☐ Azure app Push Notifications

Azure account email

☐ Voice

Country code

Phone number

Enable the common alert schema. [Learn more](#)

Yes

No

OK

With an action group, there are several ways in which you can respond to the alert. The following options are available for defining the action to take:

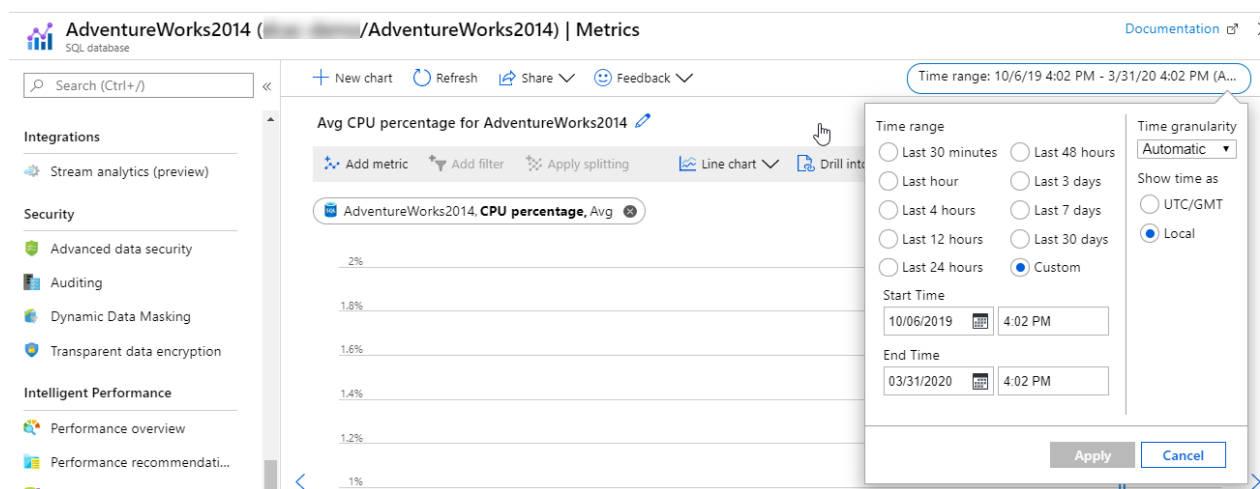
- Automation Runbook
- Azure Function
- Email Azure Resource Manager Role
- Email/SMS/Push/Voice

- ITSM
- Azure Logic App
- Secure Webhook
- Webhook

There are two categories to these actions—notification, which means to notify an administrator or group of administrators of an event, and automation, which is taking a defined action to respond to a performance condition.

Review past performance data

One of the benefits of utilizing the Azure Monitor is the ability to easily and quickly review past metrics that were gathered. If you examine a resource, you note a datetime picker in the upper right-hand corner. Azure Monitor Metrics will be retained for 93 days, after which they're purged, however you do have the option to archive them to Azure Storage.



You're also able to select a smaller window of time such as the last 30 minutes, last hour, last 4 hours, or last 12 hours as an example. The flexibility of Azure monitor allows administrators to quickly identify issues and to potentially diagnose past issues.

SQL Server metrics that matter

Microsoft SQL Server is a well instrumented piece of software that collects a great deal of performance metadata. The database engine has metrics that can be monitored to help identify and improve performance-related issues. Some operating system metrics are only viewable from within performance monitor while others can be accessed through T-SQL queries, in particular, by selecting from the dynamic management views (DMVs). There are some metrics that are exposed in both locations so knowing where to identify specific metrics is important. One example of data that can only be captured from DMVs is data and transaction log file read/write latency as exposed in `sys.dm_os_volume_stats`. On the other hand, an example of an OS metric that isn't available directly through SQL Server is the seconds per disk read and write for the disk volume. Combining

these two metrics can help you gain better understand if a performance issue is related to database structure or a physical storage bottleneck.

4. Establish baseline metrics

<https://learn.microsoft.com/en-us/training/modules/describe-performance-monitoring/4-establish-baseline-metrics>

Establish baseline metrics

Completed

A baseline is a collection of data measurements that helps you understand the normal state of your application or server's performance. Having the data collected over time allows you to identify changes from the normal state. Baselines can be as simple as a chart of CPU utilization over time, or complex aggregations of metrics to offer granular level performance data from specific application calls. The granularity of your baseline depends on the criticality of performance of your database and application.

With any type of application workload, it's imperative to establish a working baseline. A baseline helps you identify if an ongoing issue should be considered within normal parameters or has exceeded given thresholds. Without a baseline, every issue encountered could be considered normal and therefore not require any extra intervention.

Correlating SQL Server and operating system performance

When deploying SQL Server on an Azure virtual machine, it's critical to correlate the performance of SQL Server with the performance of the underlying operating system. If you're using Linux as the operating system, you need to install *InfluxDB*, *Collectd*, and *Grafana* to capture data similar to Windows Performance Monitor. These services collect data from SQL Server and provide a graphical interface to review the data. Utilizing these tools on Linux or Performance Monitor on Windows can be used in conjunction looking at SQL Server-specific data such as SQL Server wait statistics. Using these tools together allow you to identify bottlenecks in hardware or code. The following Performance Monitor counters are a sampling of useful Windows metrics, and can allow you to capture a good baseline for a SQL Server workload:

Processor(_Total)% Processor Time - This counter measures the CPU utilization of all of the processors on the server. It's a good indication of the overall workload, and when used with other counters, this counter can identify problems with query performance.

Paging File(_Total)% Usage - In a properly configured SQL Server, memory shouldn't page to the paging file on disk. However, in some configurations you may have other services running that consume system memory and lead to the operating system paging memory to disk resulting in performance degradation.

PhysicalDisk(_Total)\Avg. Disk sec/Read and Avg. Disk sec/Write - This counter provides a good metric for how the storage subsystem is working. Your latency values in most cases shouldn't be above 20 ms, and with Premium Storage you should see values less than 10 ms.

System\Processor Queue Length - This number indicates the number of threads that are waiting for the time on the processor. If it's greater than zero, it indicates CPU pressure, indicating your workload could benefit from more CPUs.

SQLServer:Buffer Manager\Page life expectancy - Page life expectancy indicates how long SQL Server expects a page to live in memory. There's no proper value for this setting. Older documentation refers to 300 seconds as proper, but that was written in a 32-bit era when servers had far less RAM. You should monitor this value over time, and evaluate sudden drops. Such drops in the counter's value could indicate poor query patterns, external memory pressure (for example, the server running a large SSIS package) or could just be normal system processing like running a consistency check on a large database.

SQLServer:SQL Statistics\Batch Requests/sec - This counter is good for evaluating how consistently busy a SQL Server is over time. Once again there's no good or bad value, but you can use this counter with % Processor time to better understand your workload and baselines.

SQLServer:SQL Statistics\SQL Compilations/sec and SQL Re-Compilations/sec - These counters are updated when SQL Server has to compile or recompile an execution plan for a query because there's no existing plan in the plan cache, or because a plan was invalidated because of a change. Recompiles can indicate T-SQL with recompile query hints, or be indicative of memory pressure on the plan cache caused by either many ad-hoc queries or simple memory pressure.

These counters are just a sample of the available performance monitor counters that are available to you. While these counters provide a good baseline of performance, you may need to examine more counters to identify specific performance problems.

Wait statistics

When a thread is being executed and is forced to wait on an unavailable resource, SQL Server keeps track of these metrics. This information is easily identifiable via the dynamic management view (DMV) `sys.dm_os_wait_stats`. This information is important to understanding the baseline performance of your database, and can help you identify specific performance issues both with query execution and hardware limitations. Identifying the appropriate wait type and corresponding resolution is critical for resolving performance issues. Wait statistics are available across the Azure SQL platform.

5. Explore extended events

<https://learn.microsoft.com/en-us/training/modules/describe-performance-monitoring/5-explore-extended-events>

Explore extended events

Completed

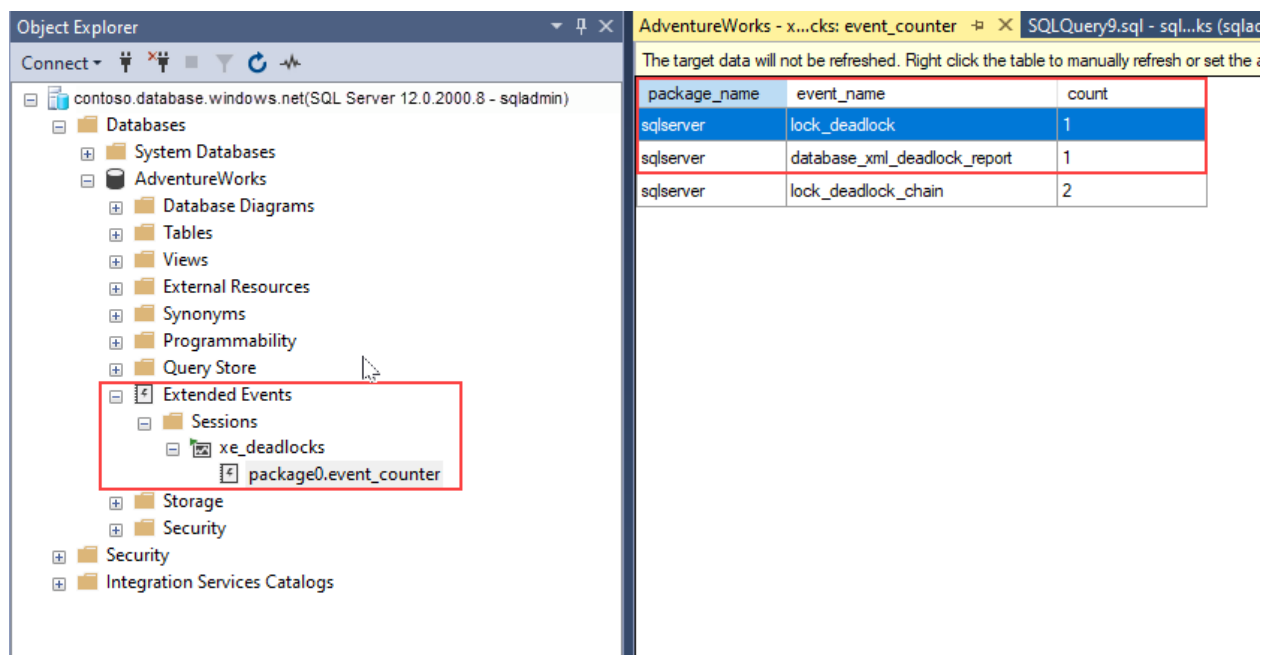
The extended events engine in Azure SQL is a lightweight and powerful monitoring system that allows you to capture granular information about activity in your databases and servers. The monitoring solutions on the Azure platform allow you to easily configure powerful monitoring for your environment and provide automated responses to error conditions.

Extended events build on the functionality of SQL Server Profiler by allowing you to trace queries and by exposing extra data (events) that you can monitor. Some examples of issues you might troubleshoot with Extended Events include:

- Troubleshooting blocking and deadlocking performance issues.
- Identifying long-running queries.
- Monitoring Data Definition Language (DDL) operations.
- Logging missing column statistics.
- Observing Memory Pressure in your database.
- Long-running physical I/O operations.

The extended event framework also allows you to use filters to limit the amount of data you collect in order to reduce the overhead of data collection, and allows you to more easily identify your performance problem by targeting your focus onto specific areas.

The following example shows an extended event session created on Azure SQL Database:



In this example, *xe_deadlocks* is the name an extended event session running on *AdventureWorks* database (on the left side of the image). The *event_counter* target node, which is under your event session node, counts the number of occurrences of each event in the event session. To view the target data in the SSMS Object Explorer, you can select the target node, and then select *View Target Data*. SSMS displays the data as we see on the left side of the image, and the count results for each event.

For more information about extended events on Azure SQL Database, see [Extended events in Azure SQL Database](#).

What can I monitor with extended events?

Extended events cover the full surface area of SQL Server, and are divided into four channels, which define the audience of an event.

- **Admin** - Admin events are targeted for end users and administrators. The events included indicate a problem within a well-defined set of actions an administrator can take. An example of this is the generation of an XML deadlock report to help identity the root cause of the deadlock.
- **Operational** - Operational events are used for analysis and diagnostics or common problems. These events can be used to trigger an action or task based on an occurrence of the event. An example of an operational event would be a database in an availability group changing state, which would indicate a failover.
- **Analytic** - Analytic events are typically related to performance events and are published in high volume. Tracing stored procedure or query execution would be an example of an analytic event.
- **Debug** - Debug events aren't fully documented and you should only use them when troubleshooting with Microsoft support.

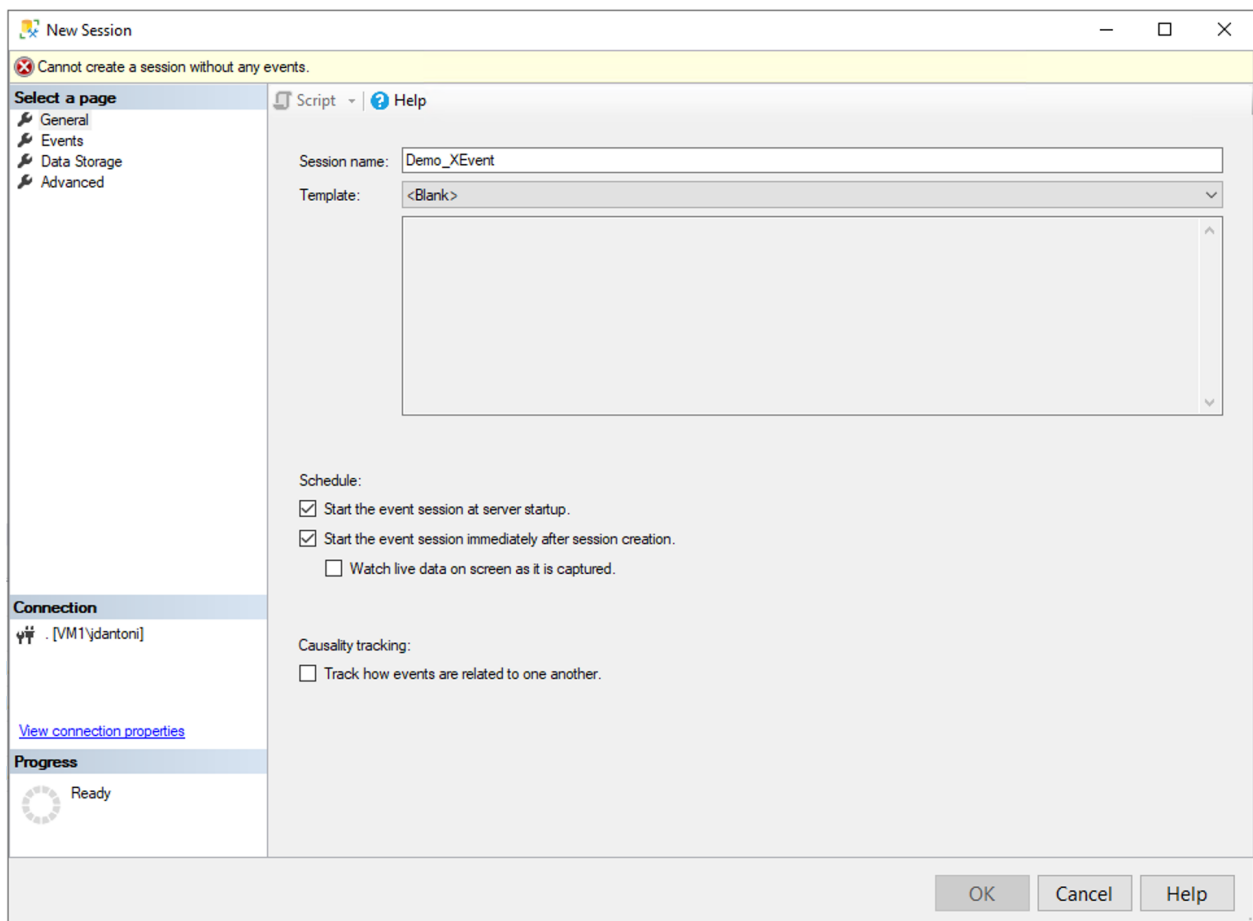
Events are added to sessions, which can host multiple events. Typically, multiple events are grouped together in a session to capture a related set of information.

You can run the following query to obtain a list of the available events, actions, and targets:

```
SELECT
    obj.object_type,
    pkg.name AS [package_name],
    obj.name AS [object_name],
    obj.description AS [description]
FROM sys.dm_xe_objects AS obj
    INNER JOIN sys.dm_xe_packages AS pkg ON pkg.guid = obj.package_guid
WHERE obj.object_type in ('action', 'event', 'target')
ORDER BY obj.object_type,
    pkg.name,
    obj.name;
```

Create extended events session

The following example shows the basic process of creating an extended events session using the *New Session* dialog from SQL Server Management Studio. You can get to this screen by expanding the *Management* node in SSMS, expanding the Extended Events node, select **Sessions**, and then **New Session**.

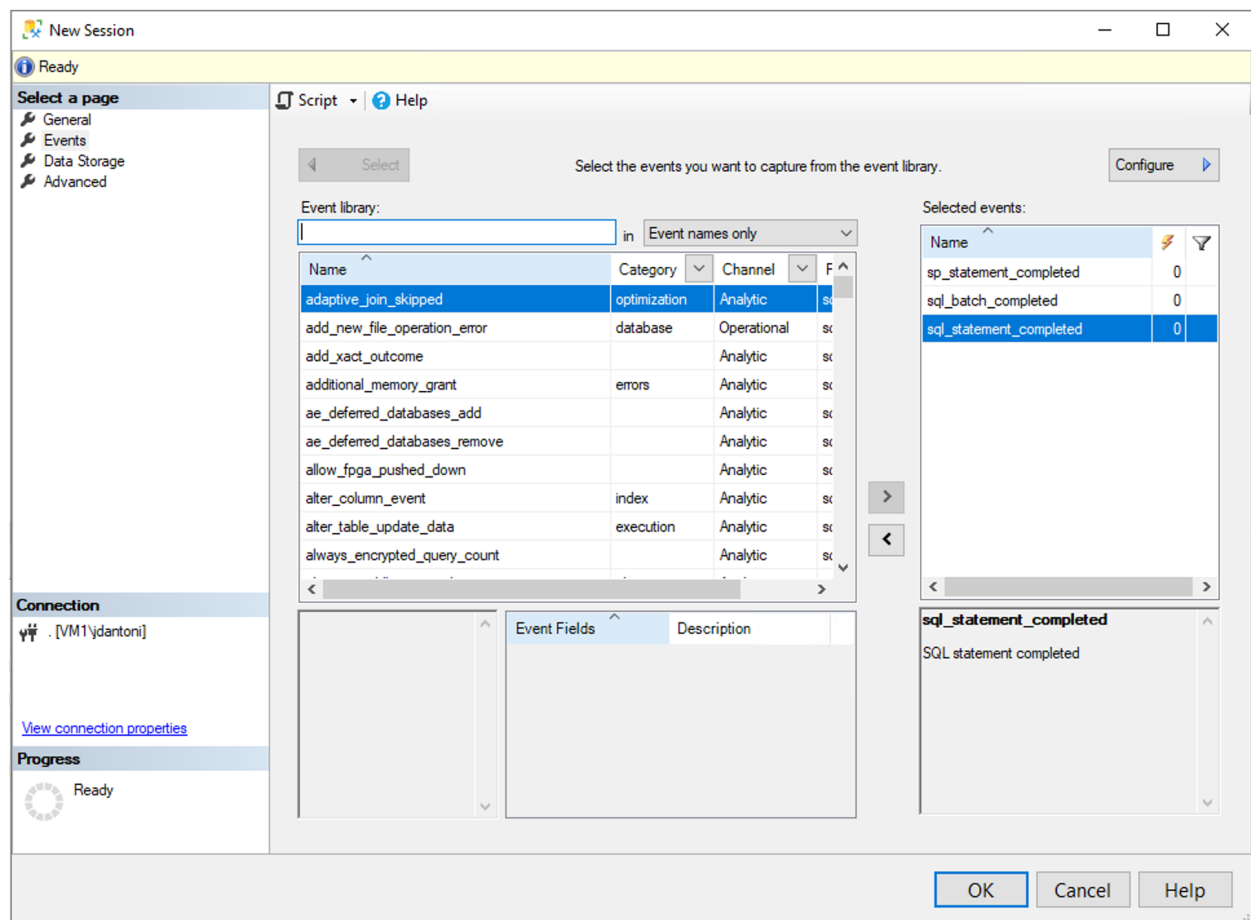


The image shows the *New Session* dialog for the extended events feature. You must first name the session. SQL Server provides numerous templates which are grouped into the following categories:

- Locks and Blocks
- Profiler Equivalents
- Query Execution
- System Monitoring

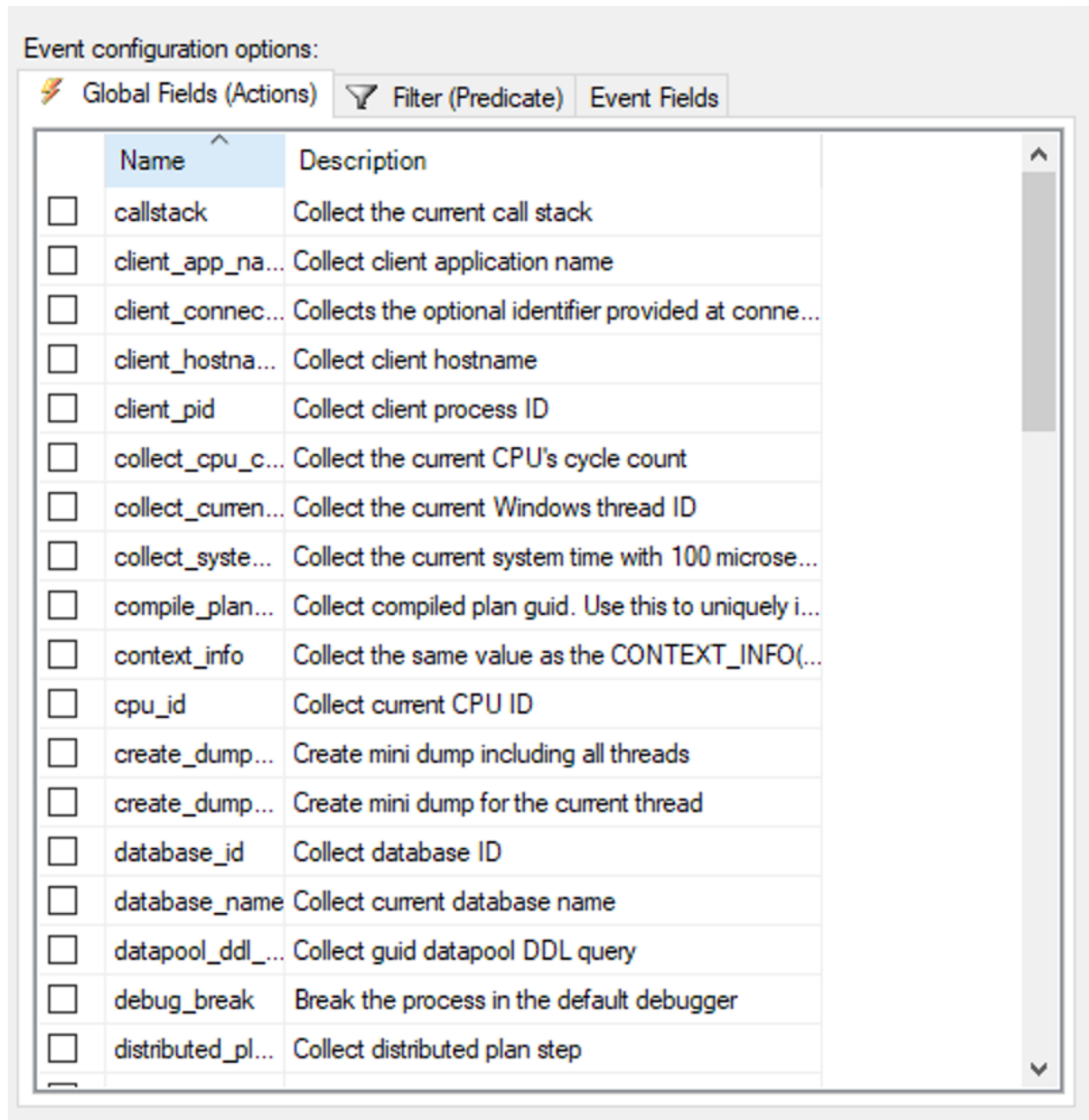
These predefined templates allow you to quickly get started with using extended events for monitoring. In this example, you see events manually added to the session to walk you through all of the options, but when you're getting started, using a template can be an easy way to create a basic session.

You have a couple of check box options for when to start this session. You can choose to have your new session started whenever the server starts, and you can also choose to start the session as soon as it's been created. Administrators can start, and stop extended event sessions at any time through the *Extended Events* node in SQL Server Management Studio. You also have the option of enabling causality tracking, which adds a globally unique identifier (GUID) and sequence number to the output of each event, which allows you to easily step through the order that the events occurred.

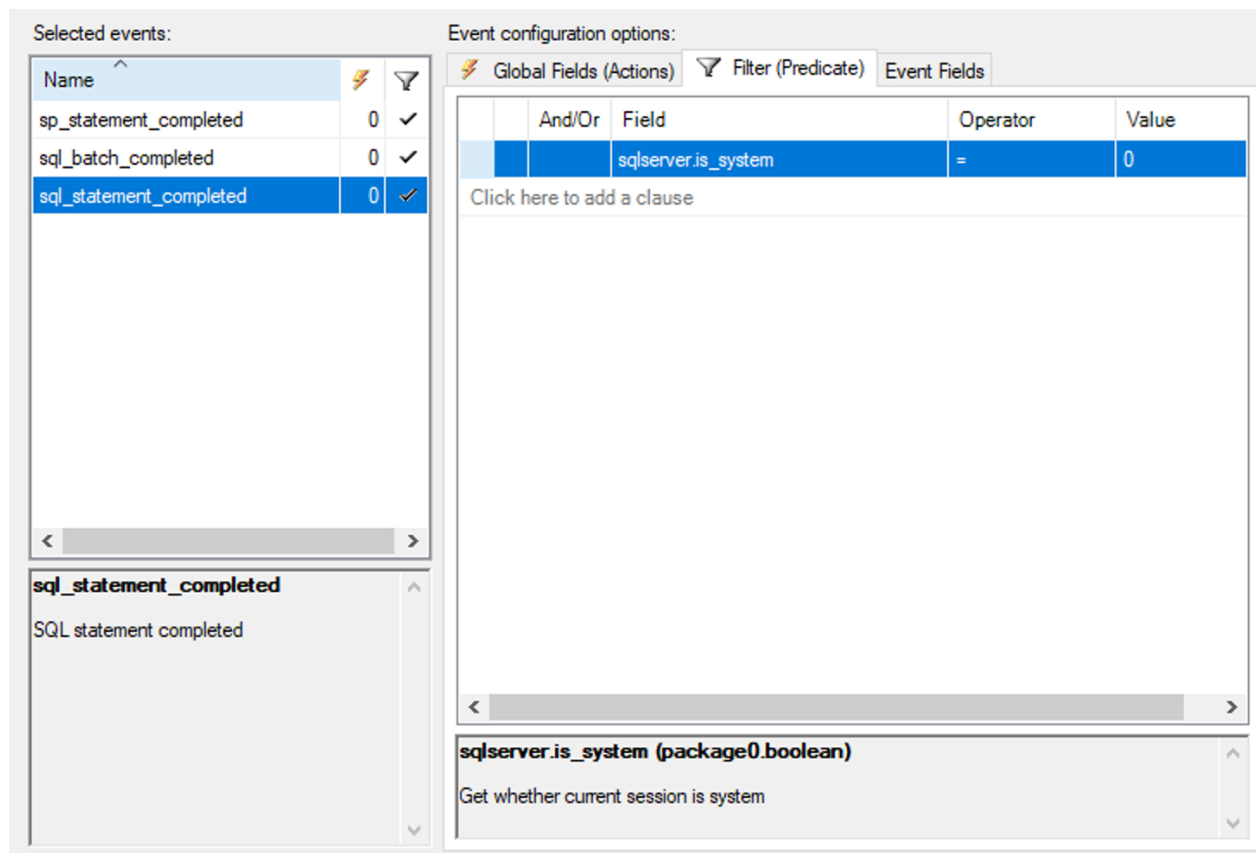


The image shows the screen where you add the events to your session. An event represents a point of interest within the database engine code, and these can represent purely internal system operations, or they can be associated with user actions like query execution. In this example, you can

see that the events `sp_statement_completed`, `sql_batch_completed`, and `sql_statement_completed` have been added to this event session. By default, this session would capture all instances of these events taking place on your instance. You can limit collection by selecting the configure option.



The event configuration screen allows for defining what data you're collecting as it relates to your events. Global fields, allow you to choose the data you're collecting, when your event occurs. Global fields are also known as actions, as the action is to add extra data fields to the event. These fields represent the data that is collected when the extended event occurs, and are common across most extended events. The image shows the filter options for an extended event.



Filters are a powerful feature of Extended Events that allow you to use granular control to capture only the specific occurrences of the event you want to capture. In this example, you can see that filter is being applied on the field `sqlserver.is_system` where it's equal to zero, which indicates that the query isn't an internal operation. In other words, the session won't capture completion of statements that are submitted by system connections, and we only want to capture statements submitted by users or user applications.

Filters apply to a single field on a single event. If you want to make sure that you aren't tracing system activities for any events, you need a separate filter for each: for the `sql_statement_completed` event, for the `sql_batch_completed` event, and for the `sp_statement_completed` event.

It's good practice to configure a filter for each event that you're capturing. This helps improve the efficiency of data collection, and allows you to narrow the focus of your search.

The image shows the event fields that are collected. These are specific to the event being triggered, and can include optional fields for collection. In the above event, you can see the optional collection options are `statement`, and `parameterized_plan_handle`.

Selected events:

Name ^		
sp_statement_completed	0	✓
sql_batch_completed	0	✓
sql_statement_completed	0	✓

sql_statement_completed

SQL statement completed

Event configuration options:

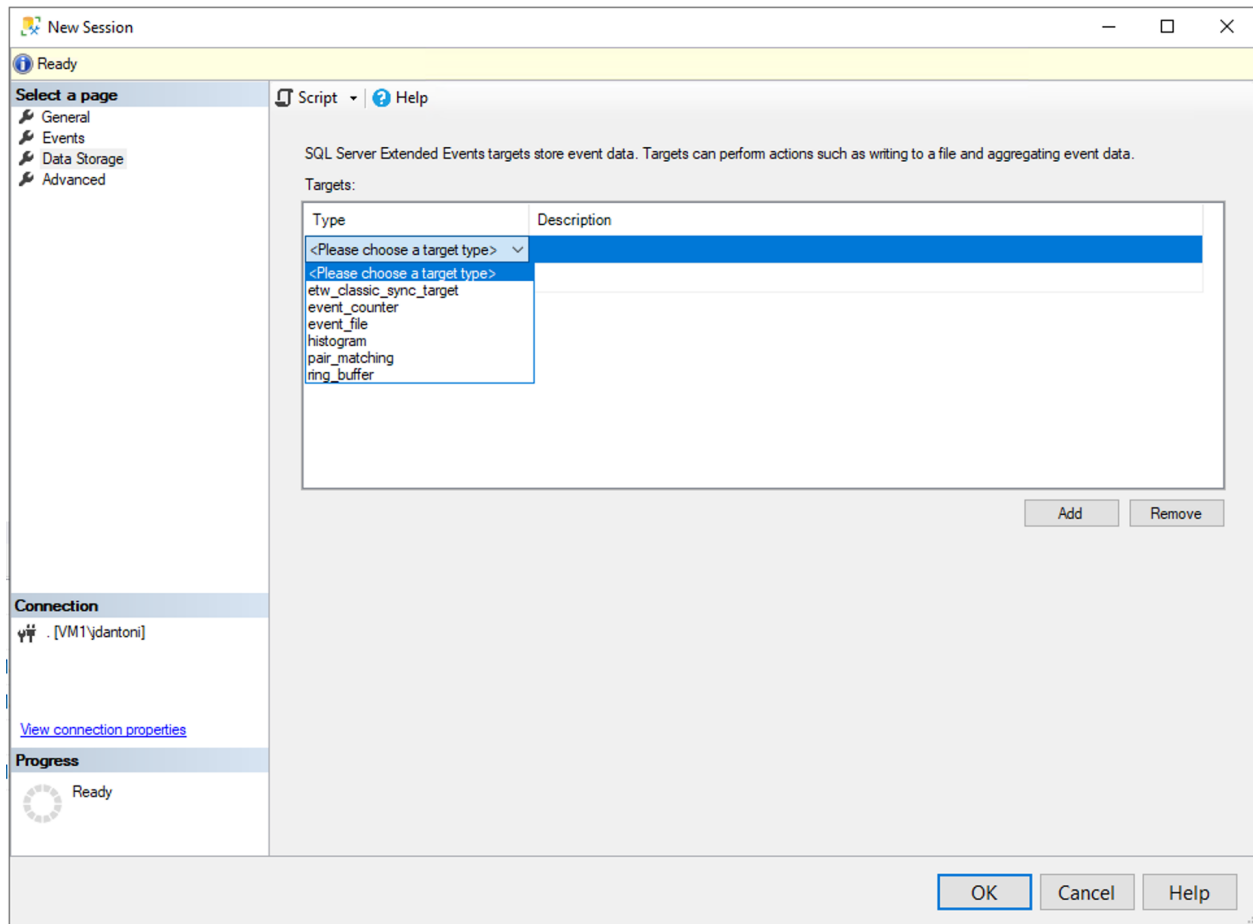
Global Fields (Actions)

Filter (Predicate)

Event Fields

Name ^	Description	Data Type
cpu_time	Indicates the CPU time (in microseconds) ...	uint64
duration	The time (in microseconds) that it took to ...	int64
last_row_count	The last row count for this statement.	uint64
line_number	The statement line number, in relation to t...	int32
logical_reads	The number of logical page reads that we...	uint64
offset	The statement start offset, in relation to th...	int32
offset_end	The statement end offset, in relation to th...	int32
page_server_reads	The number of remote page reads that we...	uint64
☐ parameterized_plan_han...	The plan handle of the cache entry of the...	binary_data
physical_reads	The number of physical page reads that w...	uint64
row_count	The number of rows that were touched by...	uint64
spills	The number of pages that were written w...	uint64
☒ statement	The text of the statement that triggered th...	unicode_string
writes	The number of page writes that were issu...	uint64

Once you have defined an event session, you define a storage target, as shown in the following image.



An extended event session has a target, which is a place for the engine to keep track of occurrences of an event. Two of the more common targets are *event file* which is a file on the file system that can store events, and in Azure SQL PaaS offerings this data is written to a blob storage. Another common target is the *ring buffer* which is within SQL Server's memory. The *ring buffer* is most commonly used for live observation of an event session as it's a circular buffer, and data isn't persisted beyond a session. Most targets process data asynchronously, which means that the event data is written to memory before being persisted to disk. The exception is the Event Tracing for Windows target (ETW), and Event Counter targets, which are processed synchronously.

The following table contains information, and uses for each type of Extended Events target.

Target	Description	Processing
Event Counter	Counts all events that occurred during an Extended Event session. This is used to obtain information about workload characteristics about a workload without the overhead of a full event collection.	Synchronous
Event File	Writes event session output from memory onto persistent file on disk.	Asynchronous
Event Pairing	Many events that generally occur in pairs (for example, lock acquire, lock release), and this collection can be used to identity when those events do no occur in a matched set.	Asynchronous
Event Tracing for Windows (ETW)	Used to correlate SQL Server events with the Windows OS event data.	Synchronous

Target	Description	Processing
Histogram	This is similar to event counter, which counts the occurrences of an event. The difference is that the histogram can count based on a specific event column or action.	Asynchronous
Ring Buffer	Used to hold data in memory. Data isn't persisted to disk and maybe frequently flushed from the buffer	Asynchronous

Alternatively, you can create an extended events session using T-SQL. The following T-SQL commands provide an example of how to create an extended events session:

```
IF EXISTS (SELECT * FROM sys.server_event_sessions WHERE name='test_session')
    DROP EVENT session test_session ON SERVER;
GO

CREATE EVENT SESSION test_session
ON SERVER
    ADD EVENT sqllos.async_io_requested,
    ADD EVENT sqlserver.lock_acquired
    ADD TARGET package0.etw_classic_sync_target (SET default_etw_session_logfile_path
    WITH (MAX_MEMORY=4MB, MAX_EVENT_SIZE=4MB);
GO
```

Event sessions can be scoped to a server or a database. In this example, you're adding two events, and using the Event Tracing for Windows (ETW) path with a file location. After you create the session, you'll have to start it. You can do this through T-SQL, and `ALTER` the session using the `STATE` option, or you can use SQL Server Management Studio for it.

6. Describe database watcher (preview)

<https://learn.microsoft.com/en-us/training/modules/describe-performance-monitoring/6-describe-database-watcher>

Describe database watcher (preview)

Completed

One of the benefits of using any of the products part of the Azure SQL family is the monitoring capability that is built into Azure platform.

Database watcher is a robust monitoring solution designed specifically for [Azure SQL Database](#) and [Azure SQL Managed Instance](#). This tool provides comprehensive insights into the performance, configuration, and overall health of your databases. It collects detailed monitoring data from various

sources, including databases, elastic pools, and SQL managed instances, and ensures that you have a clear and detailed view of your SQL estate.

Note

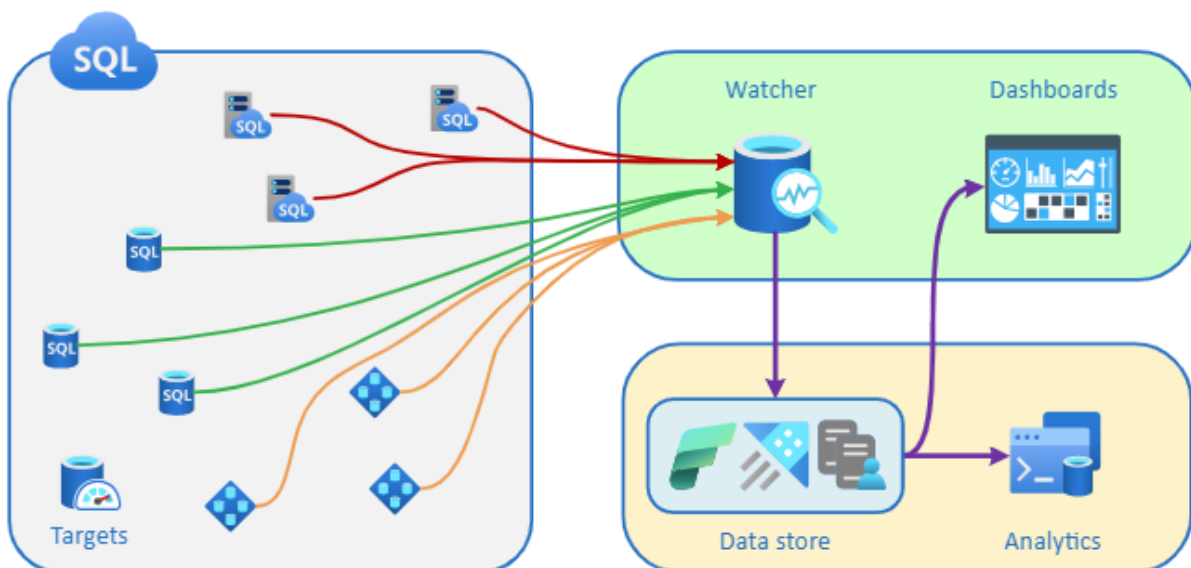
Database watcher is currently in preview.

How database watcher works

One of the key features of Database watcher is its ability to gather and store monitoring data in a central data store within your Azure subscription. This data can then be analyzed using tools like Azure Data Explorer or Real-Time Analytics in Microsoft Fabric, allowing for quick ingestion and analysis of time-series monitoring data.

Database watcher enables you to set targets, datasets, frequency, and retention according to your specific needs. It supports low latency, with data collection and ingestion happening within seconds. This ensures that you always have up-to-date information about your database's performance and health.

This tool also offers a range of dashboards in the Azure portal, providing a single-pane-of-glass view of your entire SQL estate. These dashboards offer detailed insights into each monitored resource, allowing you to quickly identify and address any issues. Additionally, Database Watcher supports parameterized templates for common alert rules, and the ability to create custom alerts, ensuring that you're always notified of critical events.



Database watcher requires a data store for the monitoring data it collects. You can use a database on an [Azure Data Explorer](#) cluster or on a free Azure Data Explorer cluster or you can use a [Real-Time Analytics](#) database in Microsoft Fabric.

Configure database watcher

Here's a simple guide to configure database watcher for Azure SQL Database and Azure SQL Managed Instance:

1. Go to the Azure portal and sign in with your credentials.
2. In the Azure portal, search for "Database Watchers" and select it. Select **Create**.
3. Select your subscription and resource group. Provide a name, and select the region where you want to deploy the database watcher.
4. On the **Data store** tab, configure the data store for the monitoring data.
5. On the **SQL targets** tab, choose the Azure SQL Database or Azure SQL Managed Instance you want to monitor.
6. Review your configuration settings and select **Create** to deploy the database watcher resource.

To learn more about database watcher, see [Quickstart: Create a database watcher to monitor Azure SQL](#).

7. Explore Query Performance Insight

<https://learn.microsoft.com/en-us/training/modules/describe-performance-monitoring/7-explore-query-performance-insight>

Explore Query Performance Insight

Completed

Identifying which queries are consuming the most resources is the first step in any database performance tuning endeavor. In older versions of SQL Server, this required extensive tracing and a series of complex SQL scripts, which could make the process of data gathering cumbersome.

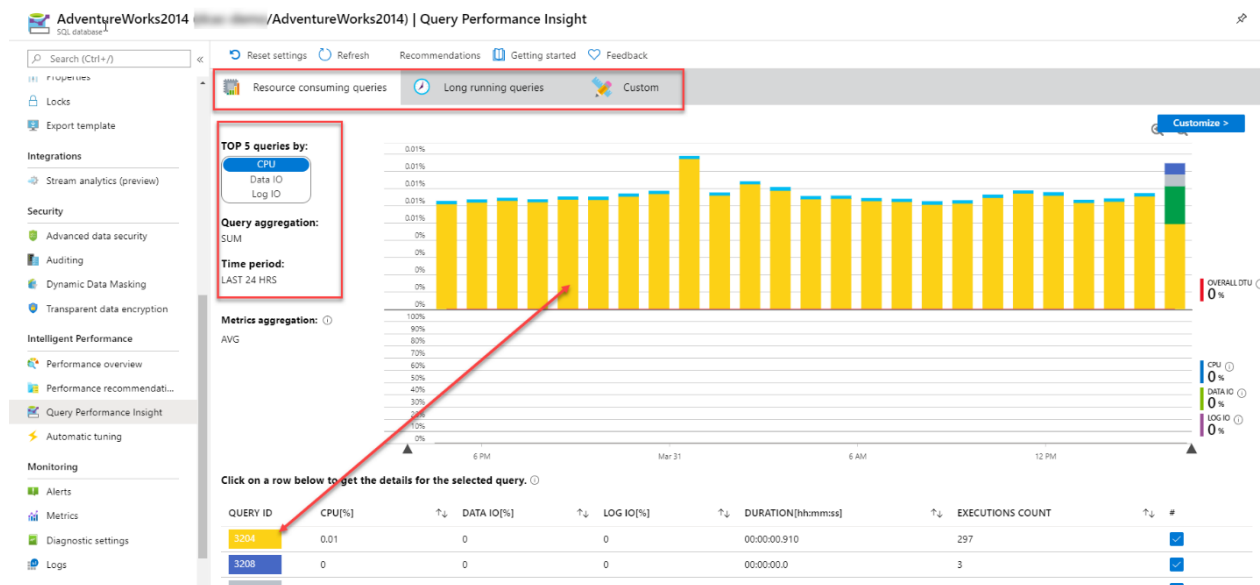
Identify problematic queries

Query Performance Insight allows the administrator to quickly identify expensive queries. You can navigate to Query Performance Insight in the main blade for your Azure SQL Database in the Intelligent Performance section.

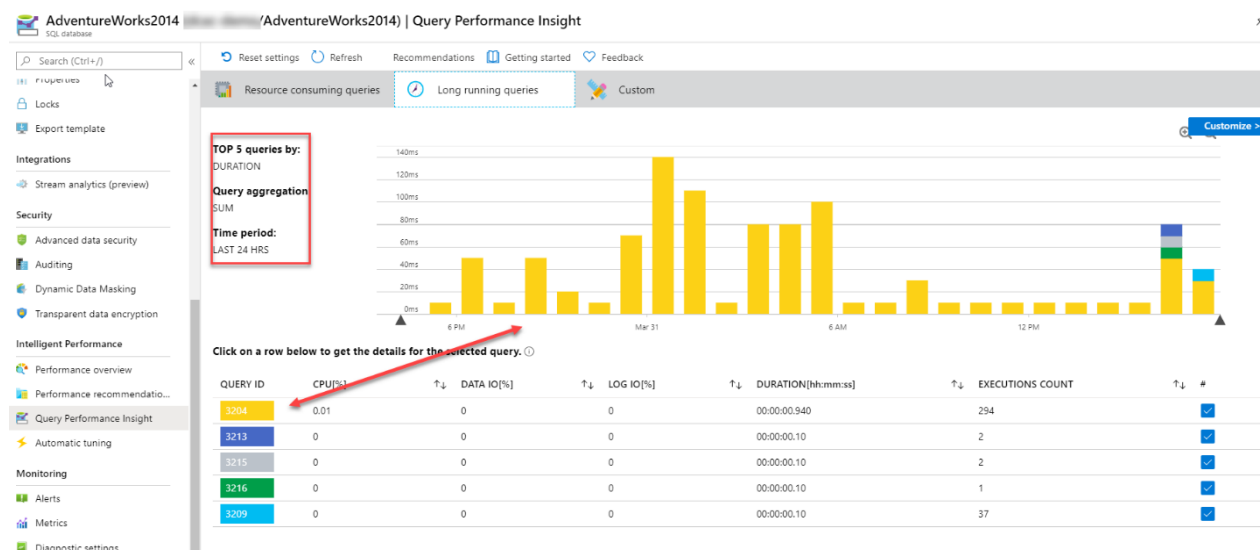
When you launch Query Performance Insight, you discover three buttons to allow you to filter for long running queries, top resource consuming queries, or a custom filter. The default value is

Resource Consuming Queries. This tab shows you the top five queries sorted by the particular resource that you select. In this case, it was sorted by CPU. You also have other options of sorting by Data IO and Log IO metrics.

You can drill into individual queries by selecting the row within the lower grid. Each row is identified with a unique color that correlates to the color within the bar graph above it.

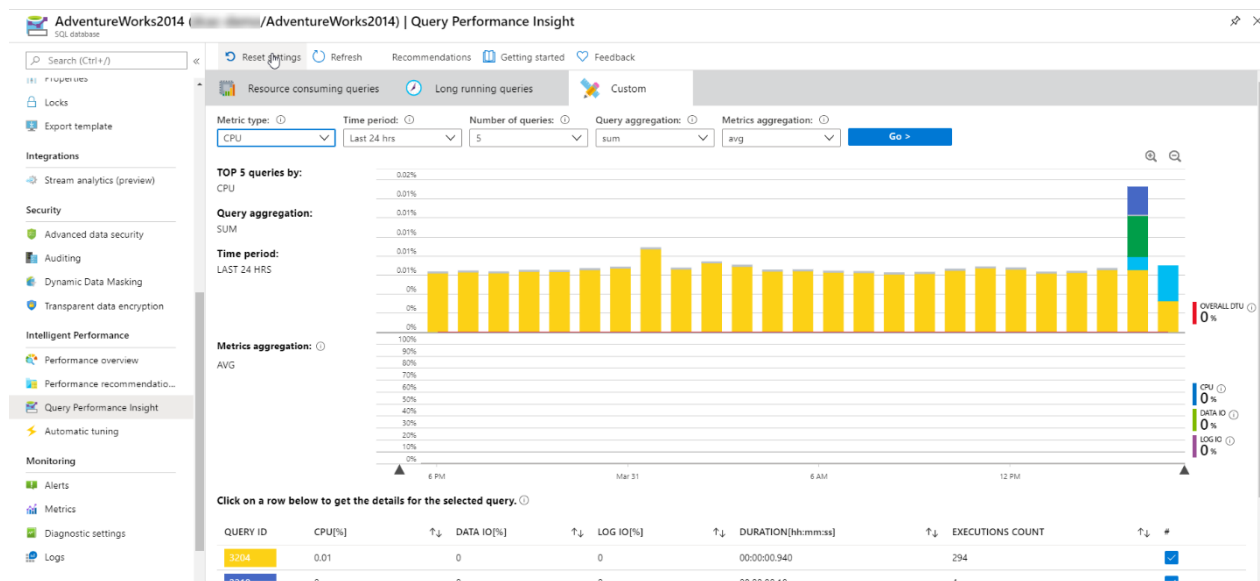


Switching to Long Running Queries, you can see a similar layout as before. In this case, the metrics are limited to the top five queries sorted by duration from the previous 24 hours and is a sum aggregation. In the grid below the graph, you can examine specific queries by selecting the row.



Switching to the custom tab offers more flexibility compared to the other two options.

Within this tab, we can further define how we wish to examine performance data. It offers us several drop-down menus that drive the visual representation of the data. The key metrics are CPU, Log IO, Data IO, and memory. These metrics are the aspects of database performance, the upper limits of which are determined by the service tier and compute resources of your Azure SQL Database.



If we drill into an individual query, we're able to see the query ID and the query itself, as well as the query aggregation type and associated time period. Furthermore, the query ID also correlates to the query ID located within the Query Store. Metrics gleaned from Query Performance Insights can then be easily located within the Query Store itself for deeper analysis or possibly problem resolution if needed.

Query details

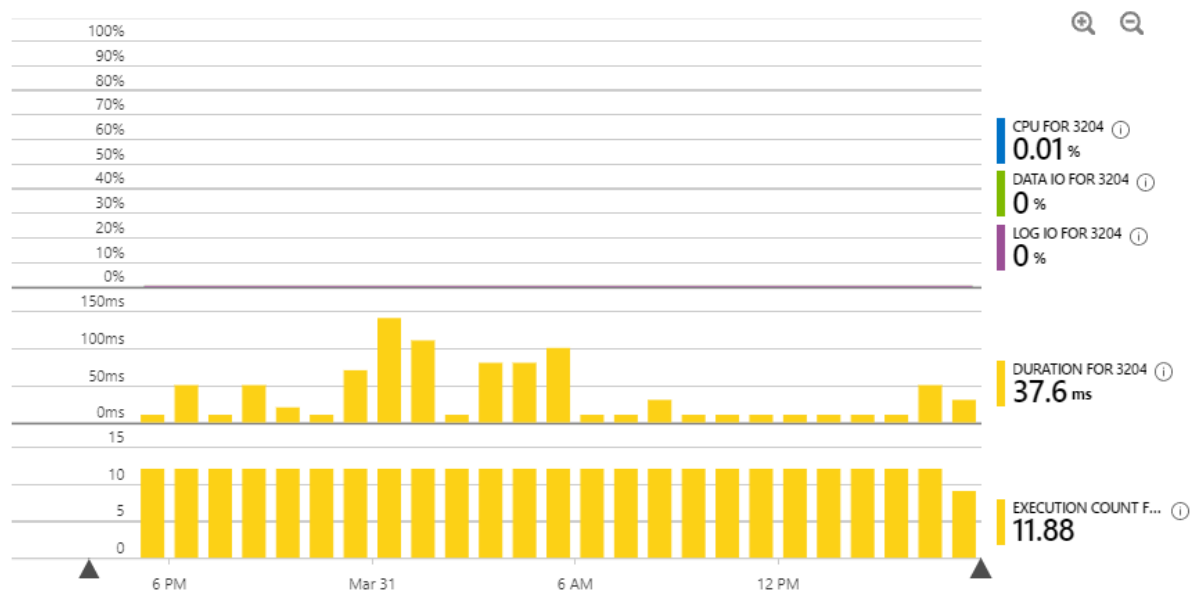
AdventureWorks2014 - Query ID 3204

Settings Refresh Recommendations </> Query Text

Query ID 3204:

```
1 SELECT c.*,
2       i.object_id, i.unique_index_id, i.is_enabled,
3       i.change_tracking_state_desc, i.has_crawl_completed,
4       i.crawl_type_desc, i.crawl_start_date, crawl_end_date,
5       i.incremental_timestamp, i.stoplist_id, i.data_space_id,
6       i.property_list_id,
7       cast(OBJECTPROPERTYEX(i.object_id, 'TableFullTextMergeStatus') as
8       int) as merge_status,
9       cast(OBJECTPROPERTYEX(i.object_id, 'TableFulltextDocsProcessed')
10      as int) as docs_processed,
```

Details of Query ID 3204 (Query aggregation: sum) Last 24 hrs



While Query Performance Insight doesn't show the query's execution plan, you can quickly identify that query, and use the information to extract the plan from the Query Store in your database.

8. Exercise: Isolate problems with monitoring

<https://learn.microsoft.com/en-us/training/modules/describe-performance-monitoring/8-exercise-isolate-problems>

Exercise: Isolate problems with monitoring

Completed

In this exercise, you'll learn how to isolate performance problems through monitoring.

Note

To complete this exercise, you'll need an [Azure subscription](#).

Launch the exercise and follow the instructions.

[Launch Exercise](#)

9. Module assessment

<https://learn.microsoft.com/en-us/training/modules/describe-performance-monitoring/9-knowledge-check>

Module assessment

Completed

10. Summary

<https://learn.microsoft.com/en-us/training/modules/describe-performance-monitoring/10-summary>

Summary

Completed

The Azure platform allows you to use the Azure Monitor to collect baseline performance data about both PaaS and IaaS resources. Within Windows, the Performance Monitor allows you to collect detailed performance information about your SQL Server. Azure SQL Database includes extra detailed query performance information through Query Performance Insight, and provides advanced monitoring capabilities through extended events.

Having a solid understand of the baseline workload of your server and database are critical to understanding performance anomalies.