+++ title = "Azure Monitor" description = "Guide for using Azure Monitor in Grafana" keywords = ["grafana", "microsoft", "azure", "monitor", "application", "insights", "log", "analytics", "guide"] aliases = ["/docs/grafana/latest/features/datasources/azuremonitor"] weight = 300 +++

Azure Monitor data source

Grafana includes built-in support for Azure Monitor, the Azure service to maximize the availability and performance of your applications and services in the Azure Cloud. The Azure Monitor data source supports visualizing data from three Azure services:

- **Azure Monitor Metrics** to collect numeric data from resources in your Azure account.
- **Azure Monitor Logs** to collect log and performance data from your Azure account, and query using the powerful Kusto Language.
- **Azure Resource Graph** to quickly query your Azure resources across subscriptions.

This topic explains configuring, querying, and other options specific to the Azure Monitor data source. Refer to [Add a data source]({{< relref "../add-a-data-source.md" >}}) for instructions on how to add a data source to Grafana.

Azure Monitor configuration

To access Azure Monitor configuration, hover your mouse over the **Configuration** (gear) icon, click **Data Sources**, and then select the Azure Monitor data source. If you haven't already, you'll need to [add the Azure Monitor data source]({{< relref "../add-a-data-source.md" >}}).

You must create an app registration and service principal in Azure AD to authenticate the data source. See the Azure documentation for configuration details. Alternatively, if you are hosting Grafana in Azure (e.g. App Service, or Azure Virtual Machines) you can configure the Azure Monitor data source to use Managed Identity to securely authenticate without entering credentials into Grafana. Refer to Configuring using Managed Identity for more details.

Name	Description
Authentication	Enables Managed Identity. Selecting Managed Identity will hide many of the fields below. See Configuring using Managed Identity for more details.
Azure Cloud	The national cloud for your Azure account. For most users, this is the default "Azure". For more information, see the Azure documentation.
Directory (tenant) ID	The directory/tenant ID for the Azure AD app registration to use for authentication. See Get tenant and app ID values for signing in from the Azure documentation.
Application (client) ID	The application/client ID for the Azure AD app registration to use for authentication.
Client secret	The application client secret for the Azure AD app registration to use for authentication. See Create a new application secret from the Azure documentation.
Default subscription	*(optional)* Sets a default subscription for template variables to use
Default workspace	*(optional)* Sets a default workspace for Log Analytics-based template variable queries to use

Azure Monitor query editor

The Azure Monitor data source has three different modes depending on which Azure service you wish to query:

- **Metrics** for Azure Monitor Metrics
- **Logs** for Azure Monitor Logs
- **Azure Resource Graph**

Querying Azure Monitor Metrics

Azure Monitor Metrics collects numeric data from supported resources and allows you to query them to investigate the health and utilization of your resources to maximise availability and performance.

Metrics are a lightweight format that only stores simple numeric data in a particular structure. Metrics is capable for supporting near real-time scenarios making it useful for fast detection of issues. Azure Monitor Logs can store a variety of different data types each with their own structure.

{{< figure src="/static/img/docs/azure-monitor/query-editor-metrics.png" max-width="800px" class="docs-image--no-shadow" caption="Azure Logs Metrics sample query visualizing CPU percentage over time" >}}

Your first Azure Monitor Metrics query

1. Select the Metrics service
2. Select a resource to pull metrics from using the subscription, resource group, resource type, and resource fields.
3. Some resources, such as storage accounts, organise metrics under multiple metric namespaces. Grafana will pick a default namespace, but change this to see which other metrics are available.
4. Select a metric from the Metric field.

Optionally, you can apply further aggregations or filter by dimensions for further analysis.

1. Change the aggregation from the default average to show minimum, maximum or total values.
2. Set a specific custom time grain. By default Grafana will automatically select a time grain interval based on your selected time range.
3. For metrics that have multiple dimensions, you can split and filter further the returned metrics. For example, the Application Insights dependency calls metric supports returning multiple time series for successful vs unsuccessful calls.

{{< figure src="/static/img/docs/azure-monitor/query-editor-metrics-dimensions.png" max-width="800px" class="docs-image--no-shadow" caption="Azure Monitor Metrics screenshot showing Dimensions" >}}

The options available will change depending on what is most relevant to the selected metric.

Legend alias formatting

The legend label for Metrics can be changed using aliases. In the Legend Format field, you can combine aliases defined below any way you want e.g

- `Blob Type: {{ blobtype }}` becomes `Blob Type: PageBlob`, `Blob Type: BlockBlob`
- `{{ resourcegroup }} - {{ resourcename }}` becomes `production - web_server`

Alias pattern	Description
`{{ resourcegroup }}`	Replaced with the the resource group
`{{ namespace }}`	Replaced with the resource type / namespace (e.g. Microsoft.Compute/virtualMachines)
`{{ resourcename }}`	Replaced with the resource name

`{{ metric }}`	Replaced with the metric name (e.g. Percentage CPU)
`{{ arbitaryDimensionID }}`	Replaced with the value of the specified dimension. (e.g. {{ blobtype }} becomes BlockBlob)
`{{ dimensionname }}`	*(Legacy for backwards compatibility)* Replaced with the name of the first dimension
`{{ dimensionvalue }}`	*(Legacy for backwards compatibility)* Replaced with the value of the first dimension

Supported Azure Monitor metrics

Not all metrics returned by the Azure Monitor Metrics API have values. To make it easier for you when building a query, the Grafana data source has a list of supported metrics and ignores metrics which will never have values. This list is updated regularly as new services and metrics are added to the Azure cloud. For more information about the list of metrics, refer to current supported namespaces.

Querying Azure Monitor Logs

Azure Monitor Logs collects and organises log and performance data from supported resources and makes many sources of data available to query together with the sophisticated Kusto Query Language (KQL).

While Azure Monitor Metrics only stores simplified numerical data, Logs can store different data types each with their own structure and can perform complexe analysis of data using KQL.

{{< figure src="/static/img/docs/azure-monitor/query-editor-logs.png" max-width="800px" class="docs-image--no-shadow" caption="Azure Monitor Logs sample query comparing successful requests to failed requests" >}}

Your first Azure Monitor Logs query

1. Select the Logs service
2. Select a resource to query. Alternatively, you can dynamically query all resources under a single resource group or subscription.
3. Enter in your KQL query. See below for examples.

Kusto Query Language

Azure Monitor Logs queries are written using the Kusto Query Language (KQL), a rich language designed to be easy to read and write, which should be familiar to those know who SQL. The Azure documentation has plenty of resource to help with learning KQL:

- Log queries in Azure Monitor
- Getting started with Kusto
- Tutorial: Use Kusto queries in Azure Monitor
- SQL to Kusto cheat sheet

Here is an example query that returns a virtual machine's CPU performance, averaged over 5m time grains

```
Perf
# $__timeFilter is a special Grafana macro that filters the results to the time span
of the dashboard
| where $__timeFilter(TimeGenerated)
| where CounterName == "% Processor Time"
| summarize avg(CounterValue) by bin(TimeGenerated, 5m), Computer
| order by TimeGenerated asc
```

Time series queries are for values that change over time, usually for graph visualisations such as the Time series panel. Each query should return at least a datetime column and a numeric value column. The result must also be sorted in ascending order by the datetime column.

A query can also have one or more non-numeric/non-datetime columns, and those columns are considered dimensions and become labels in the response. For example, a query that returns the aggregated count grouped by hour, Computer, and the CounterName:

```
Perf
| where $__timeFilter(TimeGenerated)
| summarize count() by bin(TimeGenerated, 1h), Computer, CounterName
| order by TimeGenerated asc
```

You can also select additional number value columns (with, or without multiple dimensions). For example, getting a count and average value by hour, Computer, CounterName, and InstanceName:

```
Perf
| where $__timeFilter(TimeGenerated)
| summarize Samples=count(), ["Avg Value"]=avg(CounterValue)
    by bin(TimeGenerated, $__interval), Computer, CounterName, InstanceName
| order by TimeGenerated asc
```

Table queries are mainly used in the Table panel and show a list of columns and rows. This example query returns rows with the six specified columns:

```
AzureActivity
| where $__timeFilter()
| project TimeGenerated, ResourceGroup, Category, OperationName, ActivityStatus,
Caller
| order by TimeGenerated desc
```

Logs macros

To make writing queries easier there are several Grafana macros that can be used in the where clause of a query:

Macro	Description
`$__timeFilter()`	Used to filter the results to the time range of the dashboard. Example: `TimeGenerated >= datetime(2018-06-05T18:09:58.907Z) and TimeGenerated <= datetime(2018-06-05T20:09:58.907Z).`
`$__timeFilter(datetimeColumn)`	Like `$__timeFilter()`, but specifies a custom field to filter on.
`$__timeFrom()`	Expands to the start of the dashboard time range. Example: `datetime(2018-06-05T18:09:58.907Z).`
`$__timeTo()`	Expands to the end of the dashboard time range. Example: `datetime(2018-06-05T20:09:58.907Z).`
`$__escapeMulti($myVar)`	Used with multi-value template variables that contain illegal characters. If `$myVar` has the following two values as a string `'\\grafana-vm\Network(eth0)\Total','\\hello!'`, then it expands to `@'\\grafana-`

	`vm\Network(eth0)\Total', @'\\hello!'.` If using single value variables there is no need for this macro, simply escape the variable inline instead - `@'\$myVar'`.
`$__contains(colName, $myVar)`	Used with multi-value template variables. If `$myVar` has the value `'value1','value2'`, it expands to: `colName in ('value1','value2')`. If using the `All` option, then check the `Include All Option` checkbox and in the `Custom all value` field type in the value `all`. If `$myVar` has value `all` then the macro will instead expand to `1 == 1`. For template variables with a lot of options, this will increase the query performance by not building a large "where..in" clause.

Additionally, Grafana has the built-in `$__interval` macro

Querying Azure Resource Graph

Azure Resource Graph (ARG) is a service in Azure that is designed to extend Azure Resource Management by providing efficient and performant resource exploration, with the ability to query at scale across a given set of subscriptions so that you can effectively govern your environment. By querying ARG, you can query resources with complex filtering, iteratively explore resources based on governance requirements, and assess the impact of applying policies in a vast cloud environment.

{{< figure src="/static/img/docs/azure-monitor/query-editor-arg.png" max-width="800px" class="docs-image--no-shadow" caption="Azure Resource Graph sample query listing virtual machines on an account" >}}

Your first Azure Resource Graph query

ARG queries are written in a variant of the Kusto Query Language, but not all Kusto language features are available in ARG. An Azure Resource Graph query is formatted as table data.

If your credentials give you access to multiple subscriptions, then you can choose multiple subscriptions before entering queries.

Sort results by resource properties

Here is an example query that returns all resources in the selected subscriptions, but only the name, type, and location properties:

```
Resources
| project name, type, location
| order by name asc
```

The query uses `order by` to sort the properties by the `name` property in ascending (`asc`) order. You can change what property to sort by and the order (`asc` or `desc`). The query uses `project` to show only the listed properties in the results. You can add or remove properties.

Query resources with complex filtering

Filtering for Azure resources with a tag name of `environment` that have a value of `Internal` . You can change these to any desired tag key and value. The `=~` in the `type` match tells Resource Graph to be case insensitive.

You can project by other properties or add/remove more.

For example, a query that returns a list of resources with an `environment` tag value of `Internal`:

```
Resources
| where tags.environment=~'internal'
| project name
```

Group and aggregate the values by property

You can also use `summarize` and `count` to define how to group and aggregate the values by property. For example, returning count of healthy, unhealthy, and not applicable resources per recommendation:

```
securityresources
| where type == 'microsoft.security/assessments'
| extend resourceId=id,
    recommendationId=name,
    resourceType=type,
    recommendationName=properties.displayName,
    source=properties.resourceDetails.Source,
    recommendationState=properties.status.code,
    description=properties.metadata.description,
    assessmentType=properties.metadata.assessmentType,
    remediationDescription=properties.metadata.remediationDescription,
    policyDefinitionId=properties.metadata.policyDefinitionId,
    implementationEffort=properties.metadata.implementationEffort,
    recommendationSeverity=properties.metadata.severity,
    category=properties.metadata.categories,
    userImpact=properties.metadata.userImpact,
    threats=properties.metadata.threats,
    portalLink=properties.links.azurePortal
| summarize numberOfResources=count(resourceId) by tostring(recommendationName), tostr
```

In Azure Resource Graph many nested properties (`properties.displayName`) are of a `dynamic` type, and should be cast to a string with `tostring()` to operate on them.

The Azure documentation also hosts many sample queries to help you get started

Azure Resource Graph macros

You can use Grafana macros when constructing a query. Use the macros in the where clause of a query:

- `$__timeFilter()` - Expands to `timestamp ≥ datetime(2018-06-05T18:09:58.907Z) and timestamp ≤ datetime(2018-06-05T20:09:58.907Z)` where the from and to datetimes are from the Grafana time picker.
- `$__timeFilter(datetimeColumn)` - Expands to `datetimeColumn ≥ datetime(2018-06-05T18:09:58.907Z) and datetimeColumn ≤ datetime(2018-06-05T20:09:58.907Z)` where the from and to datetimes are from the Grafana time picker.

- `$__timeFrom()` - Returns the From datetime from the Grafana picker. Example: `datetime(2018-06-05T18:09:58.907Z)`.
- `$__timeTo()` - Returns the To datetime from the Grafana picker. Example: `datetime(2018-06-05T20:09:58.907Z)`.
- `$__escapeMulti($myVar)` - Use with multi-value template variables that contain illegal characters. If `$myVar` has the two values as a string `'\\grafana-vm\Network(eth0)\Total','\\hello!'`, then it expands to: `@'\\grafana-vm\Network(eth0)\Total', @'\\hello!'`. If you are using single value variables, then there is no need for this macro, simply escape the variable inline instead - `@'\$myVar'`.
- `$__contains(colName, $myVar)` - Use with multi-value template variables. If `$myVar` has the value `'value1','value2'`, the it expands to: `colName in ('value1','value2')`.

  If using the `All` option, then check the `Include All Option` checkbox and in the `Custom all value` field type in the following value: `all`. If `$myVar` has value `all` then the macro will instead expand to `1 == 1`. For template variables with a lot of options, this will increase the query performance by not building a large "where..in" clause.

Going further with Azure Monitor

See the following topics to learn more about the Azure Monitor data source:

- [Azure Monitor template variables]({{< relref "./template-variables.md" >}}) for more interactive, dynamic, and reusable dashboards.
- [Provisioning Azure Monitor]({{< relref "./provisioning.md" >}}) for configuring the Azure Monitor data source using YAML files
- [Deprecating Application Insights]({{< relref "./provisioning.md" >}}) and migrating to Metrics and Logs queries

Configuring using Managed Identity

Customers who host Grafana in Azure (e.g. App Service, Azure Virtual Machines) and have managed identity enabled on their VM, will now be able to use the managed identity to configure Azure Monitor in Grafana. This will simplify the data source configuration, requiring the data source to be securely authenticated without having to manually configure credentials via Azure AD App Registrations for each data source. For more details on Azure managed identities, refer to the Azure documentation.

To enable managed identity for Grafana, set the `managed_identity_enabled` flag in the `[azure]` section of the Grafana server config.

```
[azure]
managed_identity_enabled = true
```

Then, in the Azure Monitor data source configuration and set Authentication to Managed Identity. The directory ID, application ID and client secret fields will be hidden and the data source will use managed identity for authenticating to Azure Monitor Metrics, Logs, and Azure Resource Graph.

{{< figure src="/static/img/docs/azure-monitor/managed-identity.png" max-width="800px" class="docs-image--no-shadow" caption="Azure Monitor Metrics screenshot showing Dimensions" >}}