Table of Contents

- Overview
  - Azure Monitor Overview
  - Metrics
  - Alerts
  - Autoscale
  - Activity log
  - Diagnostic Logs
  - Partner Integrations
- Get Started
  - Get Started with Azure Monitor
  - Roles Permissions and Security
- How to
  - Use alerts
    - Configure alerts in Azure portal
    - Configure alerts with CLI
    - Configure alerts with PowerShell
    - Configure a webhook on a metric alert
    - Create a metric alert with a Resource Manager template
  - Use autoscale
    - Best Practices for autoscale
    - Common metrics for autoscale
    - Autoscale VM Scale Sets using Resource Manager templates
    - Automatically scale machines in a virtual machine scale set
    - Configure webhooks and email notifications on autoscale
  - Use the activity log
    - View events in the activity log
    - Configure webhook on an activity log alert
    - Archive the activity log
    - Stream the activity log to Event Hubs

Audit operations with Resource Manager
Manage diagnostic logs
Archive
Stream to Event Hubs
Enable Diagnostic Settings using Resource Manager templates
Use the REST API
Walkthrough using REST API
Reference
PowerShell
.NET
REST
Resources
PowerShell Samples
Azure CLI 2.0 (Preview) Samples
List of supported metrics

Overview of Monitoring in Microsoft Azure

1/17/2017 • 5 min to read • Edit on GitHub

This article provides a conceptual overview of monitoring Azure resources. It provides pointers to information on specific types of resources. For high-level information on monitoring your application from non-Azure point of view, see Monitoring and diagnostics guidance.

Video walkthrough of Azure Monitor is available at Explore Microsoft Azure monitoring and diagnostics. Additional video explaining a scenario where you can use Azure Monitor is available at Explore Microsoft Azure monitoring and diagnostics.

Cloud applications are complex with many moving parts. Monitoring provides data to ensure that your application stays up and running in a healthy state. It also helps you to stave off potential problems or troubleshoot past ones. In addition, you can use monitoring data to gain deep insights about your application. That knowledge can help you to improve application performance or maintainability, or automate actions that would otherwise require manual intervention.

The following diagram shows a conceptual view of Azure monitoring, including the type of logs you can collect and what you can do with that data.

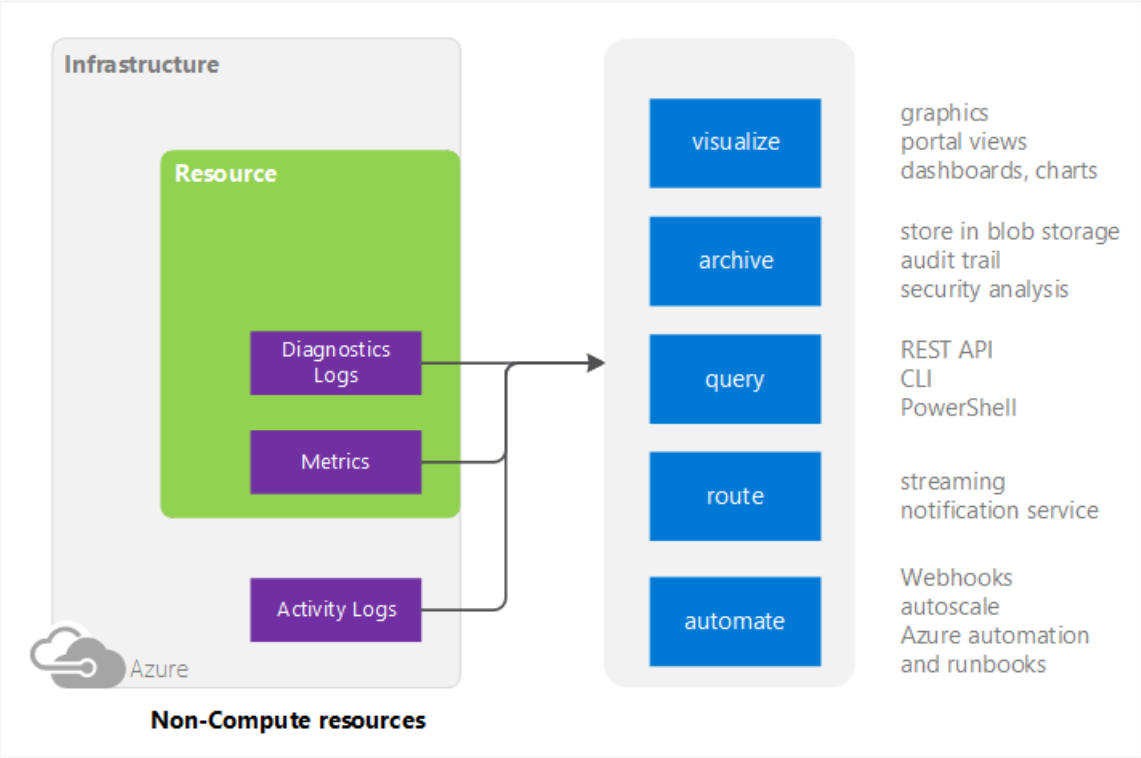


Figure 1: Conceptual Model for monitoring and diagnostics for non-compute resources

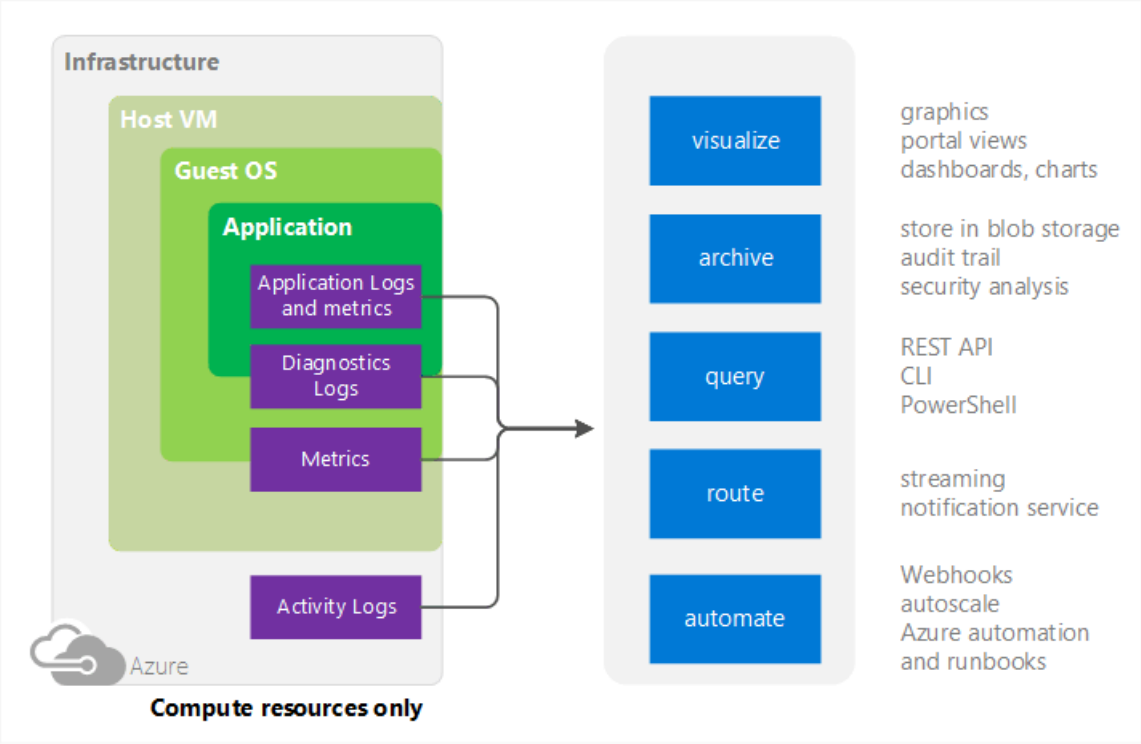


Figure 2: Conceptual Model for monitoring and diagnostics for compute resources

Monitoring Sources

Activity Logs

You can search the Activity Log (previously called Operational or Audit Logs) for information about your resource as seen by the Azure infrastructure. The log contains information such as times when resources are created or destroyed.

Host VM

Compute Only

Some compute resources like Cloud Services, Virtual Machines, and Service Fabric have a dedicated Host VM they interact with. The Host VM is the equivalent of Root VM in the Hyper-V hypervisor model. In this case, you can collect metrics on just the Host VM in addition to the Guest OS.

For other Azure services, there is not necessarily a 1:1 mapping between your resource and a particular Host VM so host VM metrics are not available.

Resource - Metrics and Diagnostics Logs

Collectable metrics vary based on the resource type. For example, Virtual Machines provides statistics on the Disk IO and Percent CPU. But those stats don't exist for a Service Bus queue, which instead provides metrics like queue size and message throughput.

For compute resources you can obtain metrics on the Guest OS and diagnostics modules like Azure Diagnostics. Azure Diagnostics helps gather and route gather diagnostic data to other locations, including Azure storage.

A list of currently collectable metrics is available at supported metrics.

Application - Diagnostics Logs, Application Logs, and Metrics

Compute Only

Applications can run on top of the Guest OS in the compute model. They emit their own set of logs and metrics.

Types of metrics include

- Performance counters
- Application Logs
- Windows Event Logs
- .NET Event Source
- IIS Logs
- Manifest based ETW
- Crash Dumps
- Customer Error Logs

Uses for Monitoring Data

Visualize

Visualizing your monitoring data in graphics and charts helps you find trends far more quickly than looking through the data itself.

A few visualization methods include:

- Use the Azure portal
- Route data to Azure Application Insights
- Route data to Microsoft PowerBI
- Route the data to a third-party visualization tool using either live streaming or by having the tool read from an archive in Azure storage

Archive

Monitoring data is typically written to Azure storage and kept there until you delete it.

A few ways to use this data:

- Once written, you can have other tools within or outside of Azure read it and process it.
- You download the data locally for a local archive or change your retention policy in the cloud to keep data for extended periods of time.
- You leave the data in Azure storage indefinitely, though you have to pay for Azure storage based on the amount of data you keep. -

Query

You can use the Azure Monitor REST API, cross platform Command-Line Interface (CLI) commands, PowerShell cmdlets, or the .NET SDK to access the data in the system or Azure storage

Examples include:

- Getting data for a custom monitoring application you have written
- Creating custom queries and sending that data to a third-party application.

Route

You can stream monitoring data to other locations in real time.

Examples include:

- Send to Application Insights so you can use the visualization tools there.
- Send to Event Hubs so you can route to third-party tools to perform real-time analysis.

Automate

You can use monitoring data to trigger events or even whole processes Examples include:

- Use data to autoscale compute instances up or down based on application load.

- Send emails when a metric crosses a predetermined threshold.
- Call a web URL (webhook) to execute an action in a system outside of Azure
- Start a runbook in Azure automation to perform any variety of tasks

Methods of Use

In general, you can manipulate data tracking, routing, and retrieval using one of the following methods. Not all methods are available for all actions or data types.

- Azure portal
- PowerShell
- Cross-platform Command Line Interface (CLI)
- REST API
- .NET SDK

Azure's Monitoring Offerings

Azure has offerings available for monitoring your services from bare-metal infrastructure to application telemetry. The best monitoring strategy combines use of all three to gain comprehensive, detailed insight into the health of your services.

- Azure Monitor – Offers visualization, query, routing, alerting, autoscale, and automation on data both from the Azure infrastructure (Activity Log) and each individual Azure resource (Diagnostic Logs). This article is part of the Azure Monitor documentation. The Azure Monitor name was released September 25 at Ignite 2016. The previous name was "Azure Insights."
- Application Insights – Provides rich detection and diagnostics for issues at the application layer of your service, well-integrated on top of data from Azure Monitoring. It's the default diagnostics platform for App Service Web Apps. You can route data from other services to it.
- Log Analytics part of Operations Management Suite – Provides a holistic IT management solution for both on-premises and third-party cloud-based infrastructure (such as AWS) in addition to Azure resources. Data from Azure Monitor can be routed directly to Log Analytics so you can see metrics and logs for your entire environment in one place.

Next steps

Learn more about

- Azure Monitor in a video from Ignite 2016
- Getting Started with Azure Monitor
- Azure Diagnostics if you are attempting to diagnose problems in your Cloud Service, Virtual Machine, or Service Fabric application.
- Application Insights if you are trying to diagnostic problems in your App Service Web app.
- Troubleshooting Azure Storage when using Storage Blobs, Tables, or Queues
- Log Analytics and the Operations Management Suite

Overview of metrics in Microsoft Azure

1/17/2017 • 5 min to read • Edit on GitHub

This article describes what metrics are in Microsoft Azure, their benefits, and how to start using them.

What are metrics?

Azure Monitor enables you to consume telemetry to gain visibility into the performance and health of your workloads on Azure. The most important type of Azure telemetry data are the metrics (also called performance counters) emitted by most Azure resources. Azure Monitor provides several ways to configure and consume these metrics for monitoring and troubleshooting.

What can you do with metrics?

Metrics are a valuable source of telemetry and enable you to do the following tasks:

- **Track the performance** of your resource (such as a VM, website, or logic app) by plotting its metrics on a portal chart and pinning that chart to a dashboard.
- **Get notified of an issue** that impacts the performance of your resource when a metric crosses a certain threshold.
- **Configure automated actions**, such as autoscaling a resource or firing a runbook when a metric crosses a certain threshold.
- **Perform advanced analytics** or reporting on performance or usage trends of your resource.
- **Archive** the performance or health history of your resource **for compliance or auditing** purposes.

What are the characteristics of metrics?

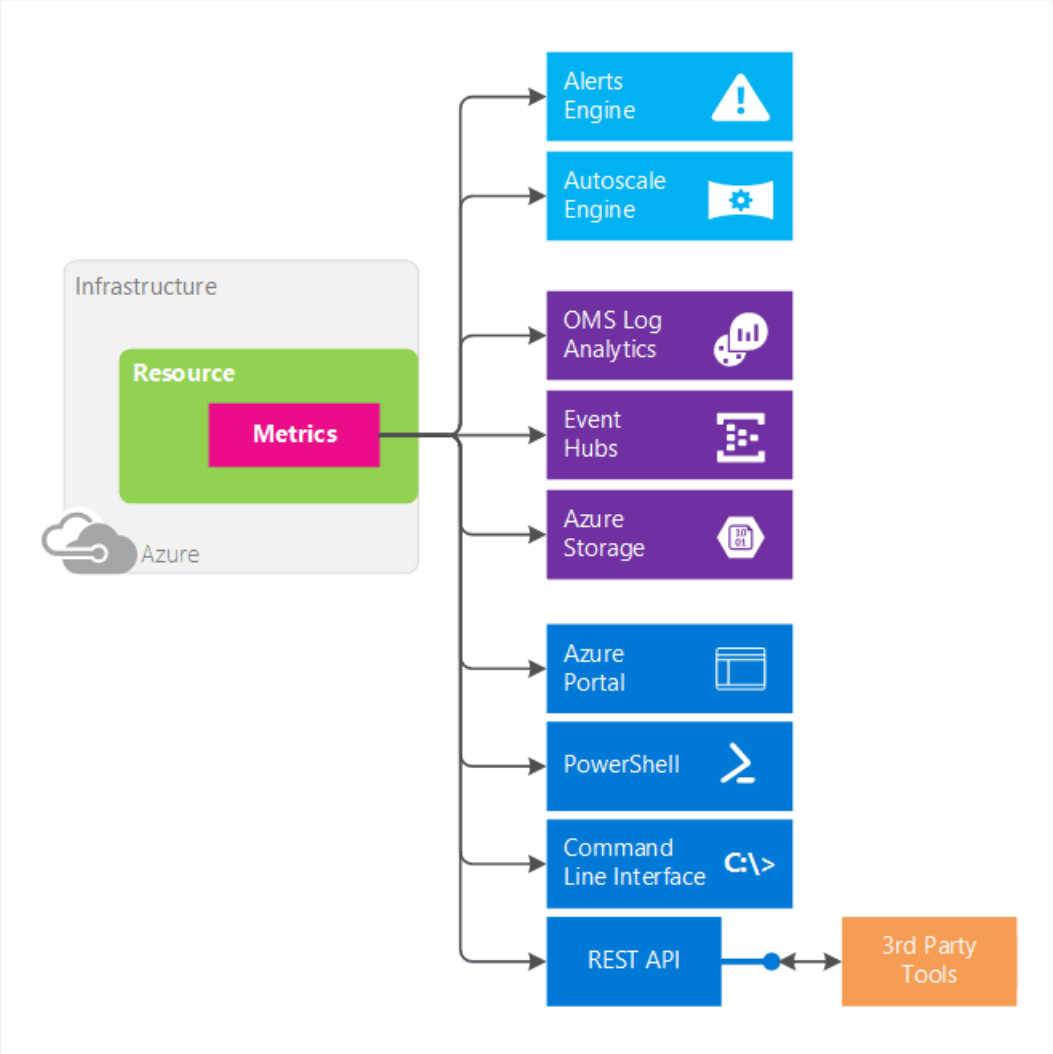
Metrics have the following characteristics:

- All metrics have **one-minute frequency**. You receive a metric value every minute from your resource, giving you near real-time visibility into the state and health of your resource.
- Metrics are **available out-of-the-box. This means you don't need to opt in** or set up additional diagnostics.
- You can access **30 days of history** for each metric. You can quickly look at the recent and monthly trends in the performance or health of your resource.

You can also:

- Easily discover, access, and **view all metrics** via the Azure portal when you select a resource and plot the metrics on a chart.
- Configure a metric **alert rule that sends a notification or takes automated action** when the metric crosses the threshold that you have set. Autoscale is a special automated action that enables you to scale out your resource to meet incoming requests or loads on your website or computing resources. You can configure an Autoscale setting rule to scale in or out based on a metric crossing a threshold.
- **Archive** metrics for longer or use them for offline reporting. You can route your metrics to Azure Blob storage when you configure diagnostic settings for your resource.
- **Stream** metrics to an event hub, enabling you to then route them to Azure Stream Analytics or to custom apps for near-real time analysis. You can do this by using diagnostic settings.
- **Route** all metrics to Log Analytics (OMS) to unlock instant analytics, search, and custom alerting on metrics data from your resources.

- **Consume** the metrics via the new Azure Monitor REST APIs.
- **Query** metrics by using the PowerShell cmdlets or the Cross-Platform REST API.

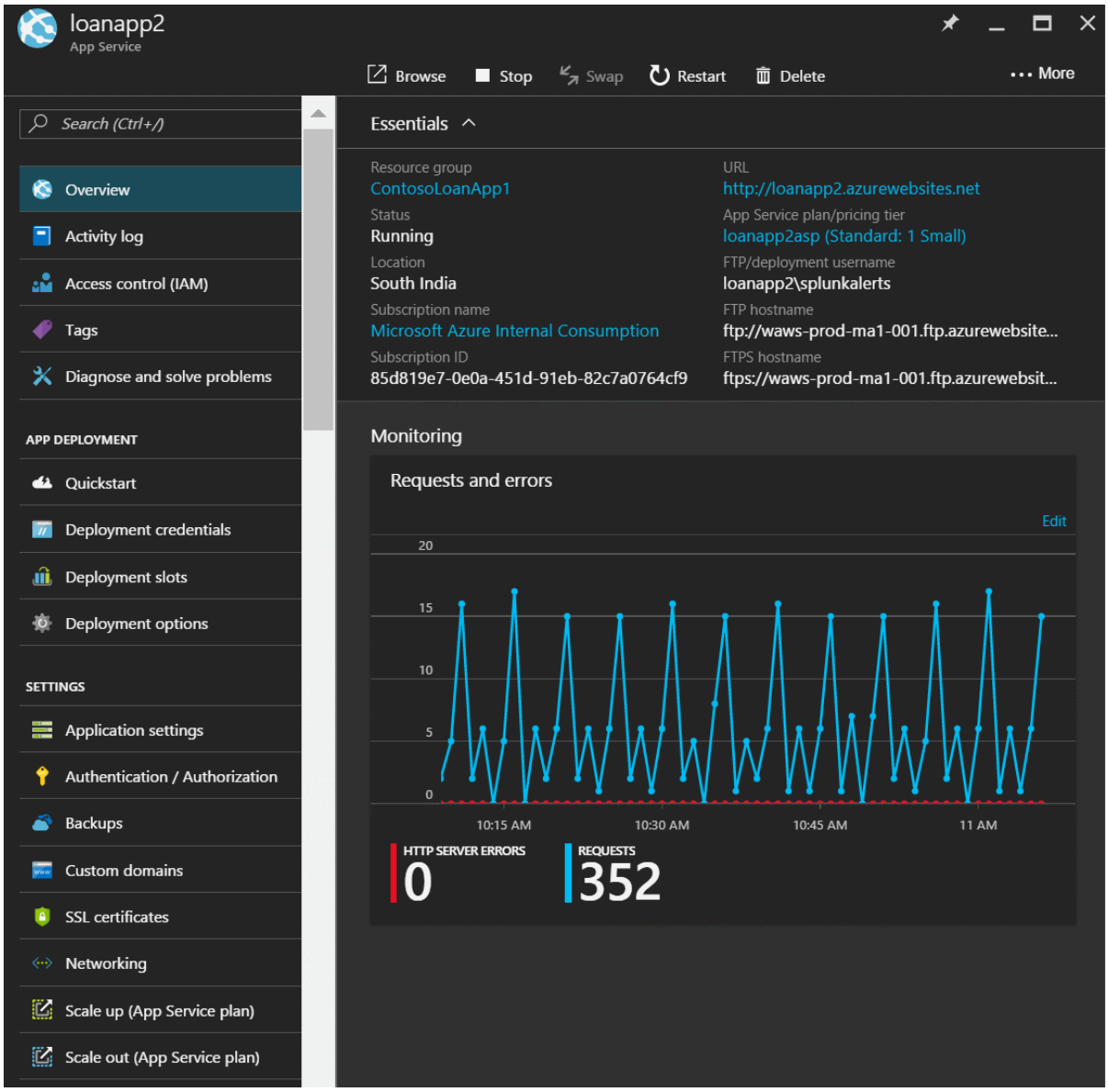


Access metrics via the portal

Following is a quick walkthrough of how to create a metric chart by using the Azure portal.

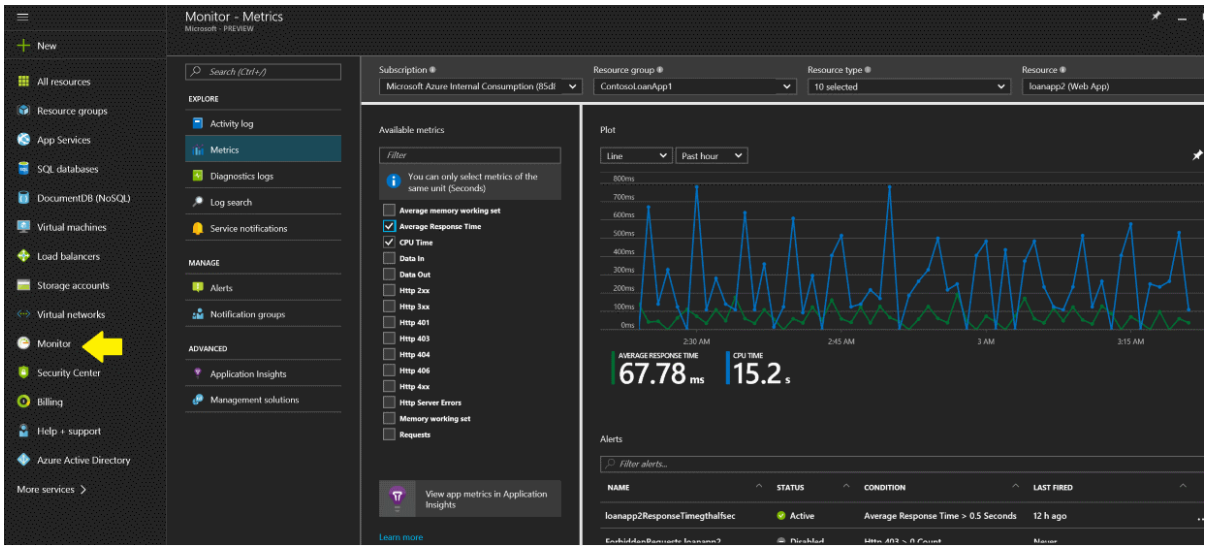
To view metrics after creating a resource

1. Open the Azure portal.
2. Create an Azure App Service website.
3. After you create a website, go to the **Overview** blade of the website.
4. You can view new metrics as a **Monitoring** tile. You can then edit the tile and select more metrics.



To access all metrics in a single place

1. Open the Azure portal.
2. Navigate to the new **Monitor** tab, and then and select the **Metrics** option underneath it.
3. Select your subscription, resource group, and the name of the resource from the drop-down list.
4. View the available metrics list. Then select the metric you are interested in and plot it.
5. You can pin it to the dashboard by clicking the pin on the upper-right corner.



> **NOTE**
> You can access host-level metrics from VMs (Azure Resource Manager-based) and virtual machine scale sets without any additional diagnostic setup. These new host-level metrics are available for Windows and Linux instances. These metrics are not to be confused with the Guest-OS-level metrics that you have access to when you turn on Azure Diagnostics on your VMs or virtual machine scale sets. To learn more about configuring Diagnostics, see What is Microsoft Azure Diagnostics.

Access metrics via the REST API

Azure Metrics can be accessed via the Azure Monitor APIs. There are two APIs that help you discover and access metrics:

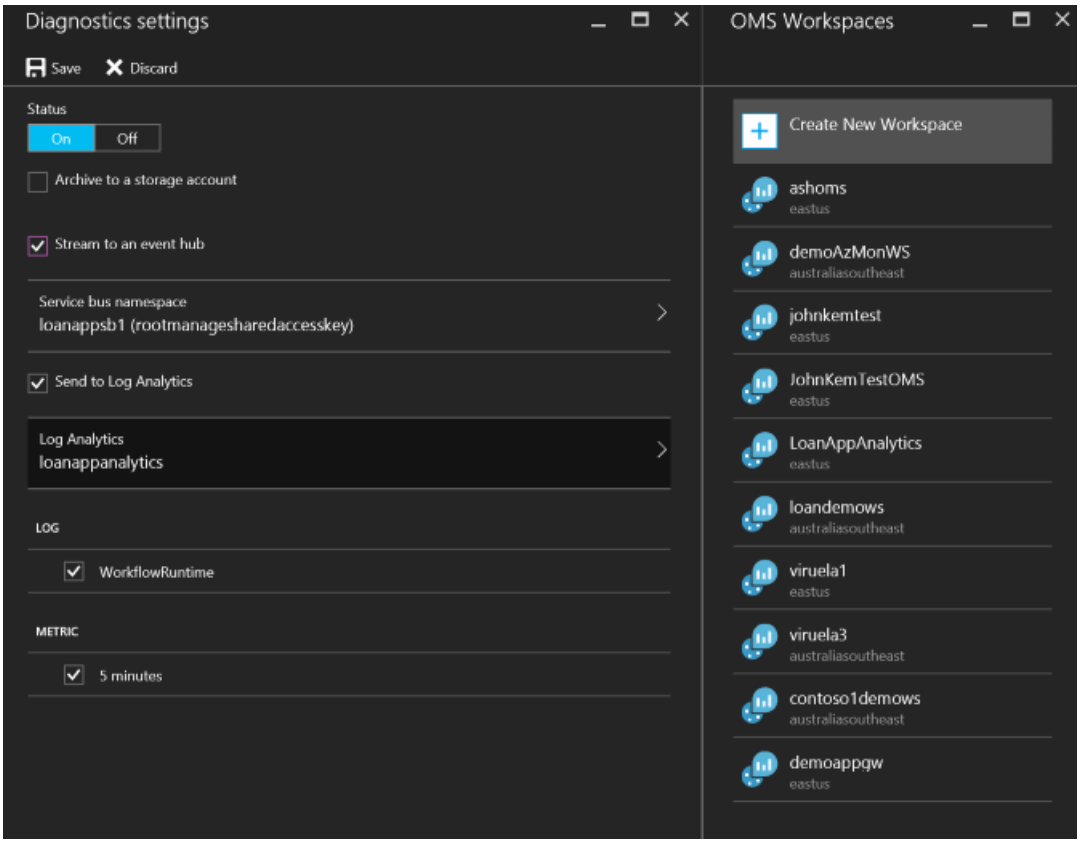
- Use the Azure Monitor Metric definitions REST API to access the list of metrics that are available for a service.
- Use the Azure Monitor Metrics REST API to access the actual metrics data.

> **NOTE**
> This article covers the metrics via the new API for metrics for Azure resources. The API version for the new metric definitions API is 2016-03-01 and the version for metrics API is 2016-09-01. The legacy metric definitions and metrics can be accessed with the API version 2014-04-01.

For a more detailed walkthrough using the Azure Monitor REST APIs, see Azure Monitor REST API walkthrough.

Export metrics

You can go to the **Diagnostics settings** blade under the **Monitor** tab and view the export options for metrics. You can select metrics (and diagnostic logs) to be routed to Blob storage, to Azure Event Hubs, or to OMS for use-cases that were mentioned previously in this article.



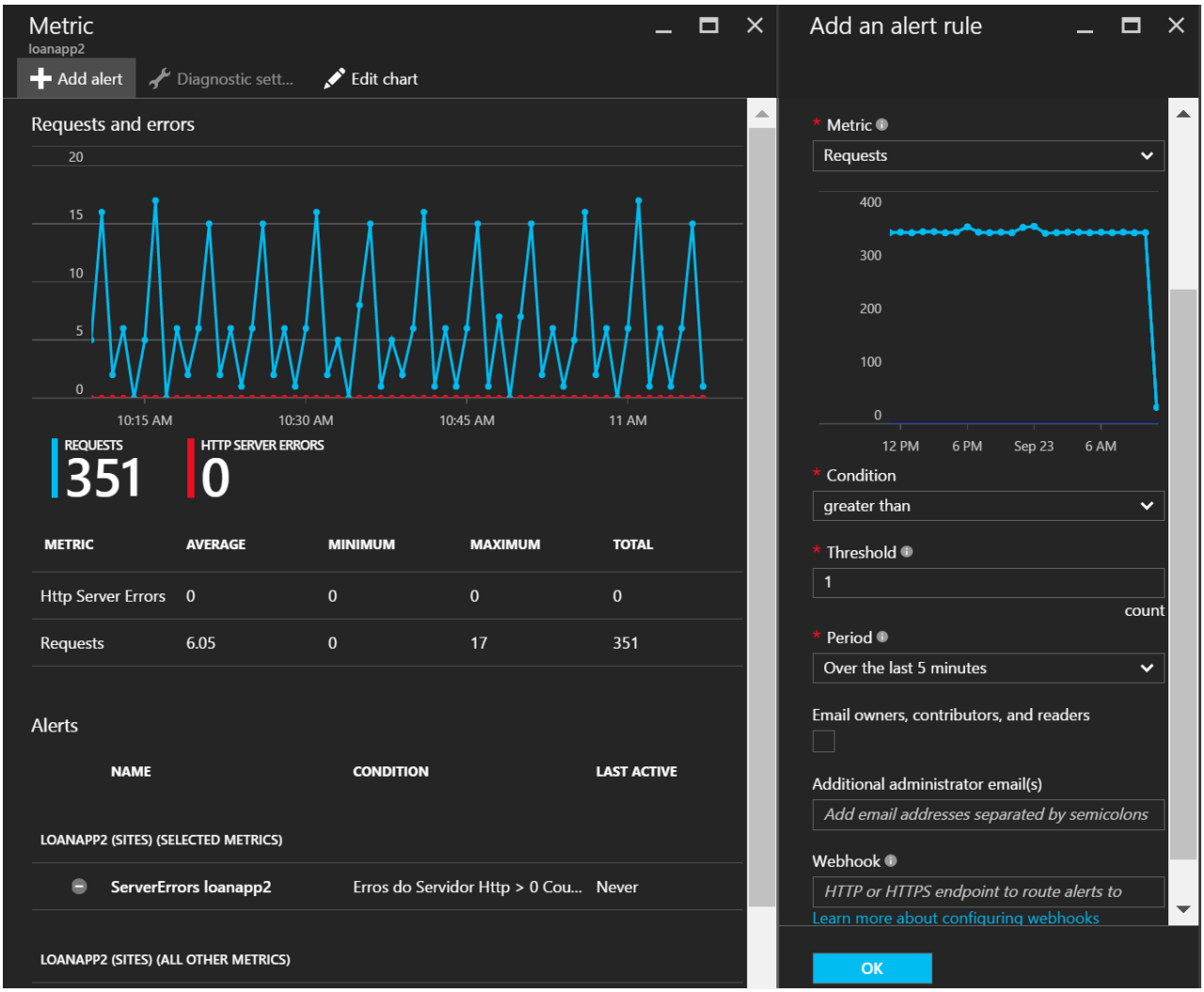
You can configure this via Resource Manager templates, PowerShell, Azure CLI, or REST APIs.

Take action on metrics

To receive notifications or take automated actions on metric data, you can configure alert rules or Autoscale settings.

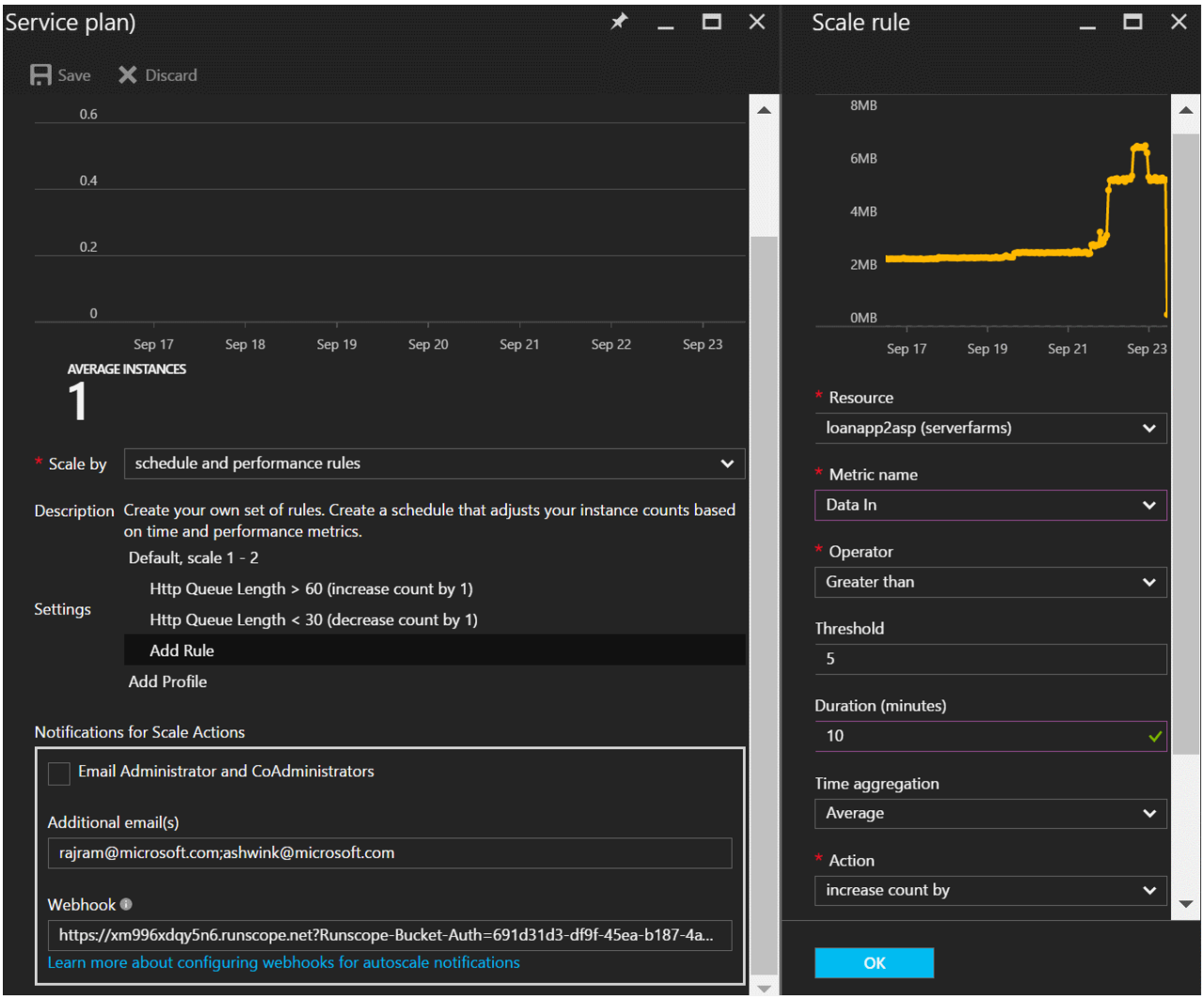
Configure alert rules

You can configure alert rules on metrics. These alert rules can check if a metric has crossed a certain threshold. They can then notify you via email or fire a webhook that can be used to run any custom script. You can also use the webhook to configure third-party product integrations.



Autoscale your Azure resources

Some Azure resources support the scaling out or in of multiple instances to handle your workloads. Autoscale applies to App Service (Web Apps), virtual machine scale sets, and classic Azure Cloud Services. You can configure Autoscale rules to scale out or in when a certain metric that impacts your workload crosses a threshold that you specify. For more information, see Overview of autoscaling.



Learn about supported services and metrics

Azure Monitor is a new metrics infrastructure. It provides support for the following Azure services in the Azure portal and the new version of the Azure Monitor API:

- VMs (Azure Resource Manager-based)
- Virtual machine scale sets
- Batch
- Event Hubs namespace
- Service Bus namespace (premium SKU only)
- SQL Database (version 12)
- Elastic SQL Pool
- Websites
- Web server farms
- Logic Apps
- IoT hubs
- Redis Cache
- Networking: Application gateways
- Search

You can view a detailed list of all the supported services and their metrics at Azure Monitor metrics--supported metrics per resource type.

Next steps

Refer to the links throughout this article. Additionally, learn about:

- Common metrics for autoscaling
- How to create alert rules

Overview of alerts in Microsoft Azure

1/17/2017 • 1 min to read • Edit on GitHub

This article describes what alerts are, their benefits, and how to get started with using them.

What are alerts?

Alerts are a method of monitoring Azure resource metrics, events, or logs and then being notified when a condition you specify is met.

You can receive alerts based on:

- **Metric values**: This alert triggers when the value of a specified metric crosses a threshold that you assign in either direction. That is, it triggers both when the condition is first met and then afterward when that condition is no longer being met.
- **Activity log events**: This alert can trigger on every event or only when a certain number of events occur.

Alerts in different Azure services

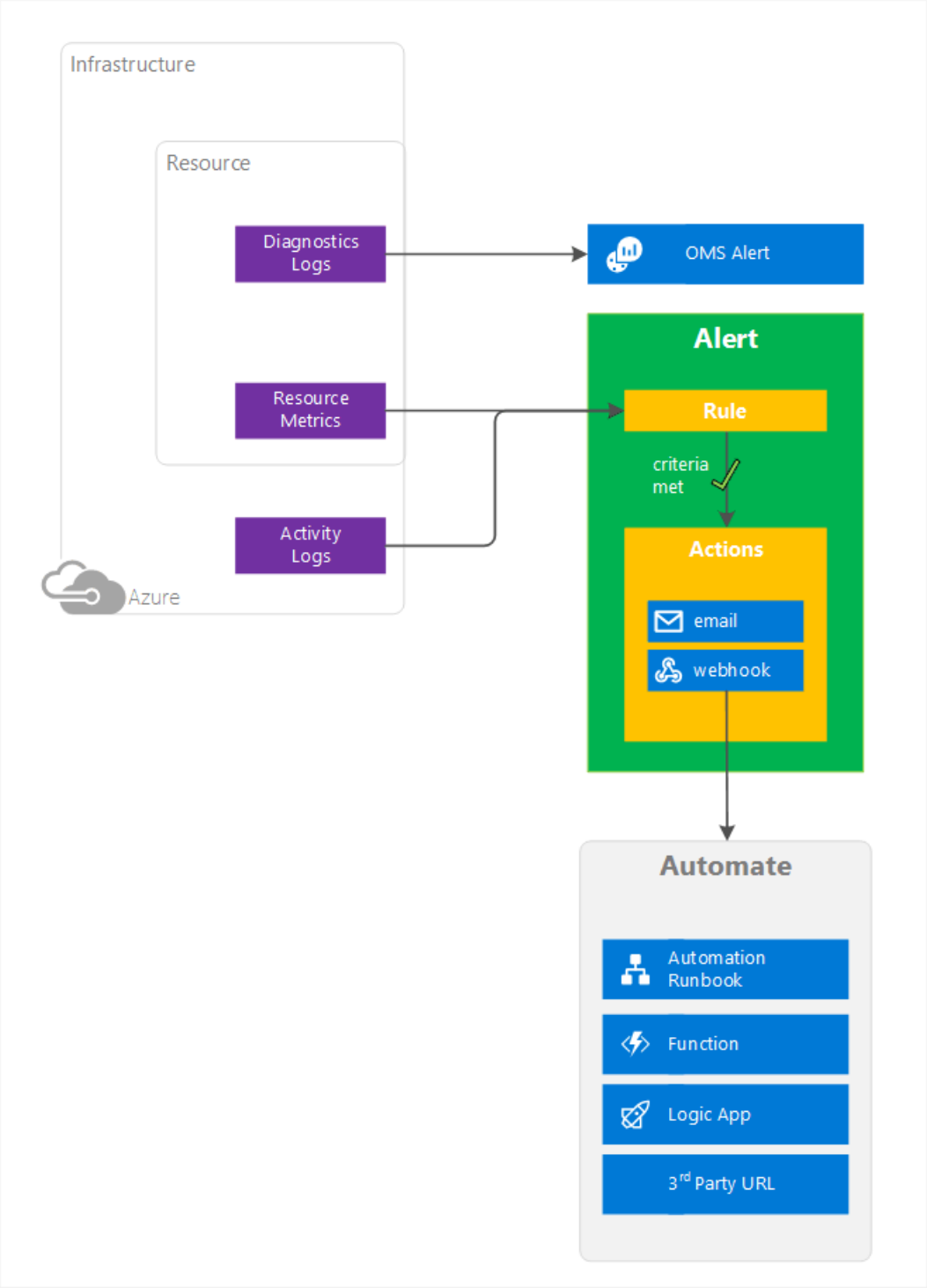
Alerts are available across different services, including:

- **Application Insights**: Enables web test and metric alerts. See Set alerts in Application Insights and Monitor availability and responsiveness of any website.
- **Log Analytics (Operations Management Suite)**: Enables the routing of diagnostic logs to Log Analytics. Operations Management Suite allows metric, log, and other alert types. For more information, see Alerts in Log Analytics.
- **Azure Monitor**: Enables alerts based on both metric values and activity log events. Azure Monitor includes the Azure Monitor REST API. For more information, see Using the Azure portal, PowerShell, or the command-line interface to create alerts.

Alert actions

You can configure an alert to do the following:

- Send email notifications to the service administrator, to co-administrators, or to additional email addresses that you specify.
- Call a webhook, which enables you to launch additional automation actions.



Next steps

Get information about alert rules and configuring them by using:

- Azure portal
- PowerShell
- Command-line interface (CLI)
- Azure Monitor REST API

Overview of autoscale in Microsoft Azure Virtual Machines, Cloud Services, and Web Apps

1/17/2017 • 4 min to read • Edit on GitHub

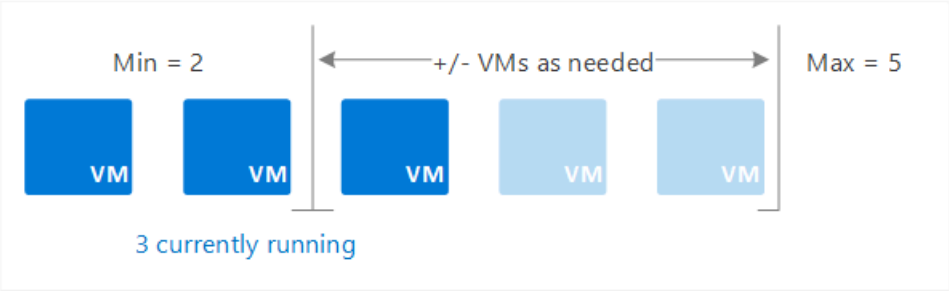
This article describes what Microsoft Azure autoscale is, its benefits, and how to get started using it.

Azure Monitor autoscale applies only to Virtual Machine Scale Sets, Cloud Services, and App Service - Web Apps.

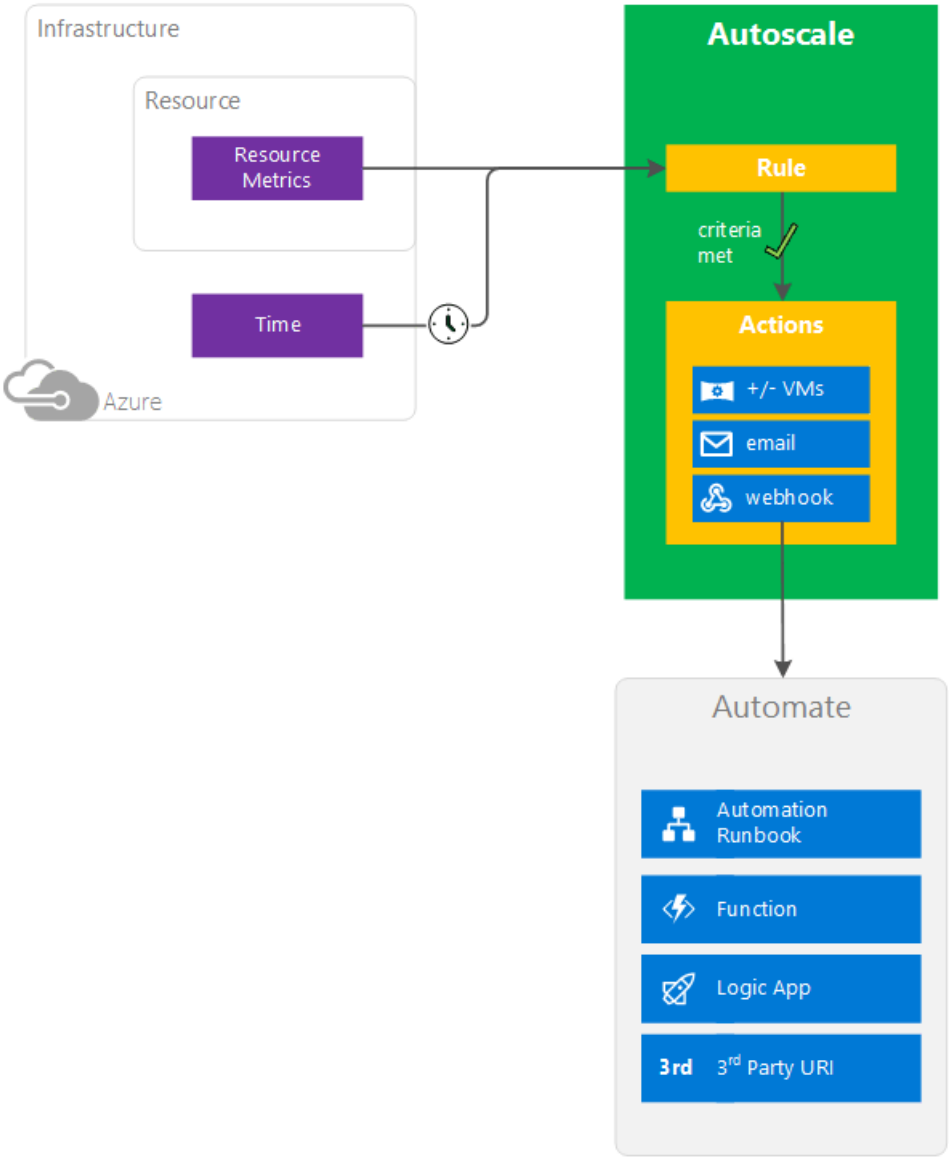
> **NOTE**
>
> Azure has two autoscale methods. An older version of autoscale applies to Virtual Machines (availability sets). This feature has limited support and we recommend migrating to VM Scale Sets for faster and more reliable autoscale support. A link on how to use the older technology is included in this article.

What is autoscale?

Autoscale allows you to have the right amount of resources running to handle the load on your application. It allows you to add resources to handle increases in load and also save money by removing resources which are sitting idle. You specify a minimum and maximum number of instances to run and add or remove VMs automatically based on a set of rules. Having a minimum makes sure your application is always running even under no load. Having a maximum limits your total possible hourly cost. You automatically scale between these two extremes using rules you create.



When rule conditions are met, one or more autoscale actions is triggered. You can add and remove VMs, or perform other actions. The following conceptual diagram shows this process.



Autoscale Process Explained

The following explanation apply to the pieces of the previous diagram.

Resource metrics

Resources emit metrics, which are later processed by rules. Metrics come via different methods. VM Scale Sets uses telemetry data from Azure diagnostics agents whereas telemetry for Web apps and Cloud services comes directly from the Azure Infrastructure. Some commonly used statistics include CPU Usage, memory usage, thread counts, queue length, and disk usage. For a list of what telemetry data you can use, see Autoscale Common Metrics.

Time

Schedule-based rules are based on UTC. You must set your time zone properly when setting up your rules.

Rules

The diagram shows only one autoscale rule, but you can have many of them. You can create complex overlapping rules as needed for your situation. Rule types include

- **Metric-based** - For example, do this action when CPU usage is above 50%.
- **Time-based** - For example, trigger a webhook every 8am on Saturday in a given time zone.

Metric-based rules measure application load and add or remove VMs based on that load. Schedule-based rules allow you to scale when you see time patterns in your load and want to scale before a possible load increase or decrease occurs.

Actions and automation

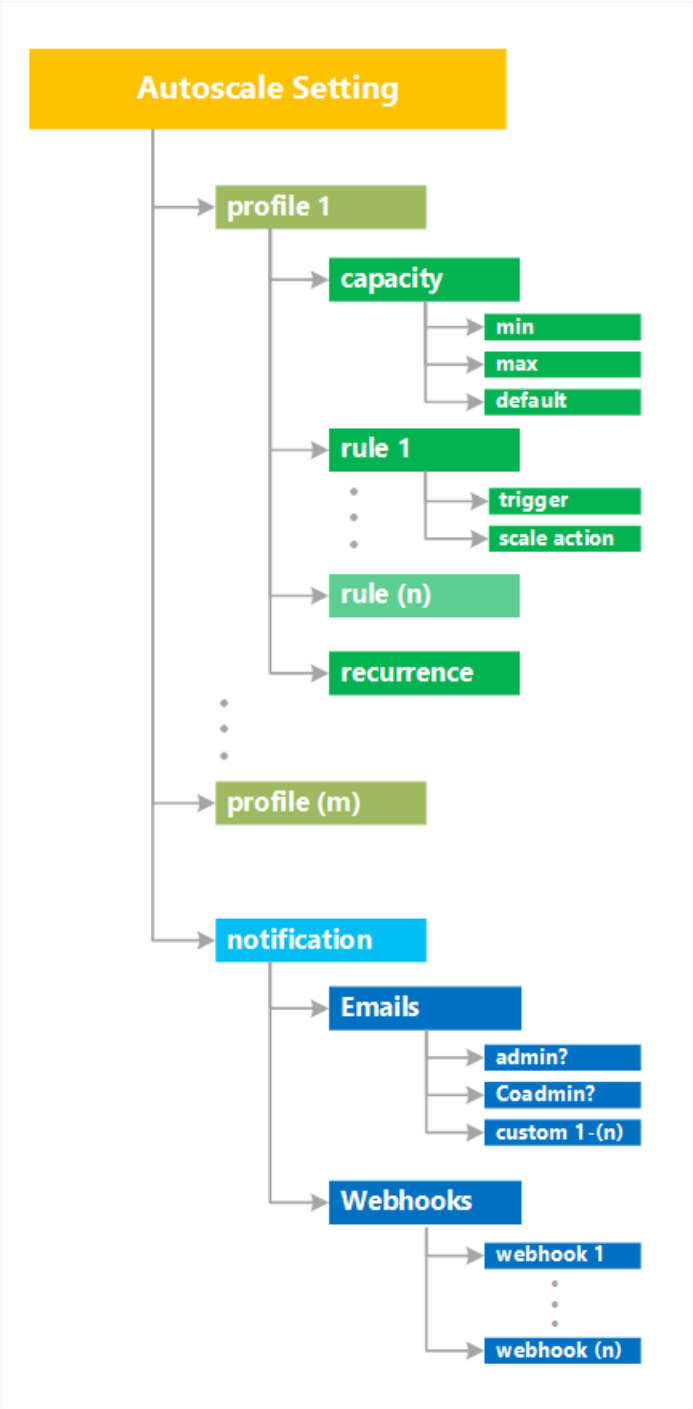
Rules can trigger one or more types of actions.

- **Scale** - Scale VMs in or out
- **Email** - Send email to subscription admins, co-admins, and/or additional email address you specify
- **Automate via webhooks** - Call webhooks, which can trigger multiple complex actions inside or outside Azure. Inside Azure, you can start an Azure Automation runbook, Azure Function, or Azure Logic App. Example 3rd party URL outside Azure include services like Slack and Twilio.

Autoscale Settings

Autoscale use the following terminology and structure.

- An **autoscale setting** is read by the autoscale engine to determine whether to scale up or down. It contains one or more profiles, information about the target resource, and notification settings.
  - An **autoscale profile** is a combination of a capacity setting, a set of rules governing the triggers, and scale actions for the profile, and a recurrence. You can have multiple profiles, which allow you to take care of different overlapping requirements.
    - A **capacity setting** indicates the minimum, maximum, and default values for number of instances. [appropriate place to use fig 1]
    - A **rule** includes a trigger—either a metric trigger or a time trigger—and a scale action, indicating whether autoscale should scale up or down when that rule is satisfied.
    - A **recurrence** indicates when autoscale should put this profile into effect. You can have different autoscale profiles for different times of day or days of the week, for example.
- A **notification setting** defines what notifications should occur when an autoscale event occurs based on satisfying the criteria of one of the autoscale setting's profiles. Autoscale can notify one or more email addresses or make calls to one or more webhooks.



The full list of configurable fields and descriptions is available in the Autoscale REST API.

For code examples, see

- Advanced Autoscale configuration using Resource Manager templates for VM Scale Sets
- Autoscale REST API

Horizontal vs vertical scaling

Autoscale only scales horizontally, which is an increase ("out") or decrease ("in") in the number of VM instances. Horizontal is more flexible in a cloud situation as it allows you to run potentially thousands of VMs to handle load.

In contrast, vertical scaling is different. It keeps the same number of VMs, but makes the VMs more ("up") or less ("down") powerful. Power is measured in memory, CPU speed, disk space, etc. Vertical scaling has more limitations. It's dependent on the availability of larger hardware, which quickly hits an upper limit and can vary by region. Vertical scaling also usually requires a VM to stop and restart.

For more information, see Vertically scale Azure virtual machine with Azure Automation.

Methods of access

You can set up autoscale via

- Azure portal
- PowerShell
- Cross-platform Command Line Interface (CLI)
- Azure Monitor REST API

Supported services for autoscale

SERVICE	SCHEMA & DOCS
Web Apps	Scaling Web Apps
Cloud Services	Autoscale a Cloud Service
Virtual Machines : Classic	Scaling Classic Virtual Machine Availability Sets
Virtual Machines : Windows Scale Sets	Scaling VM Scale Sets in Windows
Virtual Machines : Linux Scale Sets	Scaling VM Scale Sets in Linux
Virtual Machines : Windows Example	Advanced Autoscale configuration using Resource Manager templates for VM Scale Sets

Next steps

To learn more about autoscale, use the Autoscale Walkthroughs listed previously or refer to the following resources:

- Azure Monitor autoscale common metrics
- Best practices for Azure Monitor autoscale
- Use autoscale actions to send email and webhook alert notifications
- Autoscale REST API
- Troubleshooting Virtual Machine Scale Sets Autoscale

Overview of the Azure Activity Log

1/17/2017 • 7 min to read • Edit on GitHub

The **Azure Activity Log** is a log that provides insight into the operations that were performed on resources in your subscription. The Activity Log was previously known as "Audit Logs" or "Operational Logs," since it reports control-plane events for your subscriptions. Using the Activity Log, you can determine the 'what, who, and when' for any write operations (PUT, POST, DELETE) taken on the resources in your subscription. You can also understand the status of the operation and other relevant properties. The Activity Log does not include read (GET) operations.

The Activity Log differs from Diagnostic Logs, which are all logs emitted by a resource. These logs provide data about the operation of that resource, rather than operations on that resource.

You can retrieve events from your Activity Log using the Azure portal, CLI, PowerShell cmdlets, and Azure Monitor REST API.

View this video introducing the Activity Log.

What you can do with the Activity Log

Here are some of the things you can do with the Activity Log:

- Query and view it in the **Azure portal**.
- Query it via REST API, PowerShell Cmdlet, or CLI.
- Create an email or webhook alert that triggers off an Activity Log event.
- Save it to a **Storage Account** for archival or manual inspection. You can specify the retention time (in days) using **Log Profiles**.
- Analyze it in PowerBI using the **PowerBI content pack**.
- Stream it to an **Event Hub** for ingestion by a third-party service or custom analytics solution such as PowerBI.

The storage account or event hub namespace does not have to be in the same subscription as the subscription emitting logs as long as the user who configures the setting has appropriate RBAC access to both subscriptions.

Export the Activity Log with Log Profiles

A **Log Profile** controls how your Activity Log is exported. Using a Log Profile, you can configure:

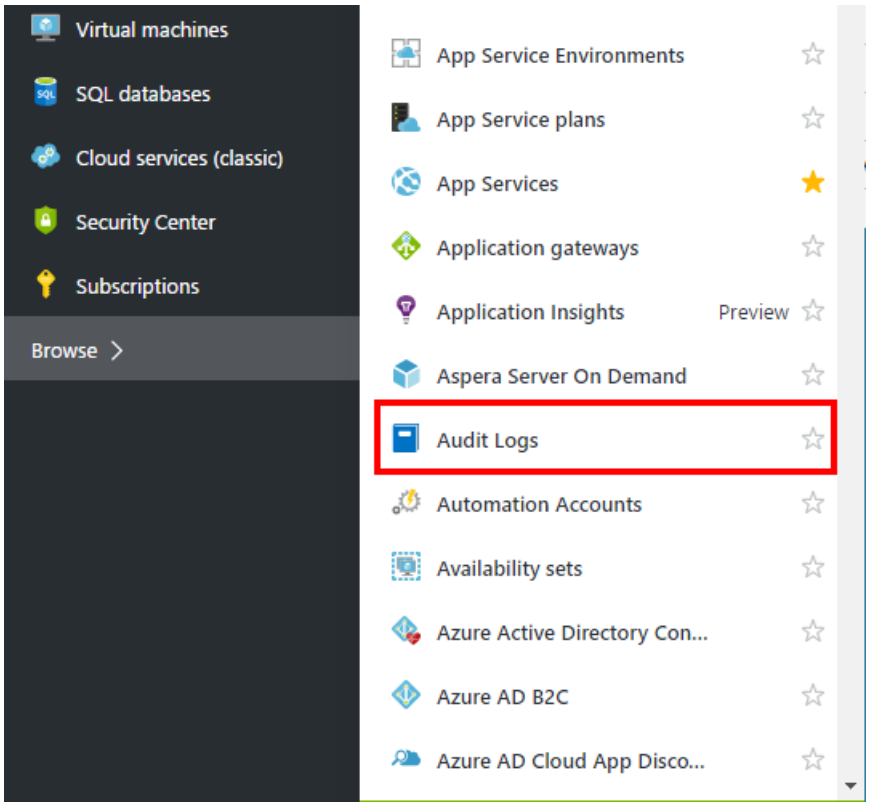
- Where the Activity Log should be sent (Storage Account or Event Hubs)
- Which event categories (Write, Delete, Action) should be sent
- Which regions (locations) should be exported
- How long the Activity Log should be retained in a Storage Account – a retention of zero days means logs are kept forever. Otherwise, the value can be any number of days between 1 and 2147483647. If retention policies are set but storing logs in a Storage Account is disabled (for example, if only Event Hubs or OMS options are selected), the retention policies have no effect. Retention policies are applied per-day, so at the end of a day (UTC), logs from the day that is now beyond the retention policy will be deleted. For example, if you had a retention policy of one day, at the beginning of the day today the logs from the day before yesterday would be deleted.

These settings can be configured via the "Export" option in the Activity Log blade in the portal. They can also be configured programmatically using the Azure Monitor REST API, PowerShell cmdlets, or CLI. A subscription can only have one log profile.

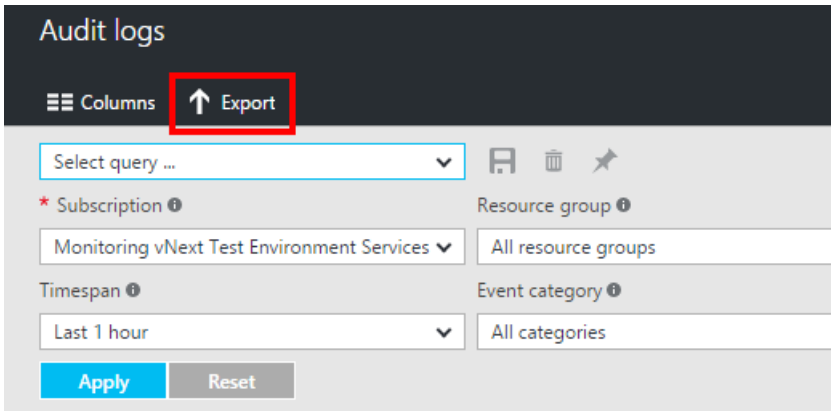
Configure log profiles using the Azure portal

You can stream the Activity Log to an Event Hub or store them in a Storage Account by using the "Export" option in the Azure portal.

1. Navigate to the **Activity Log** blade using the menu on the left side of the portal.

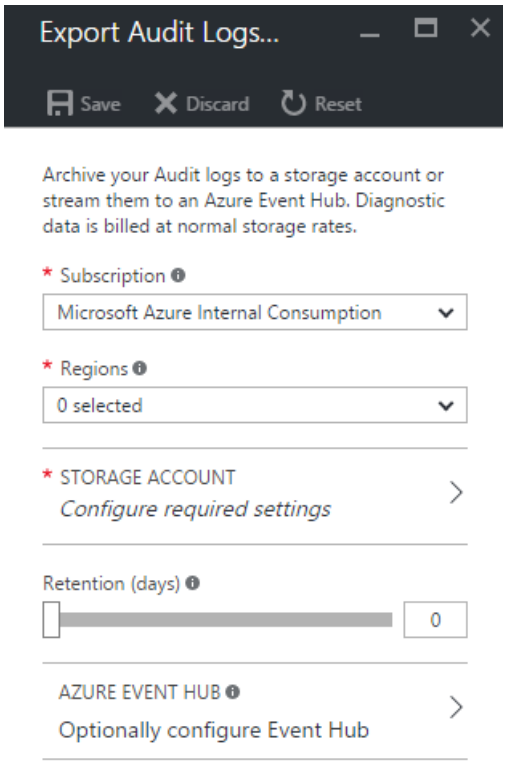


2. Click the **Export** button at the top of the blade.



3. In the blade that appears, you can select:

- regions for which you would like to export events
- the Storage Account to which you would like to save events
- the number of days you want to retain these events in storage. A setting of 0 days retains the logs forever.
- the Service Bus Namespace in which you would like an Event Hub to be created for streaming these events.



4. Click **Save** to save these settings. The settings are immediately be applied to your subscription.

Configure log profiles using the Azure PowerShell Cmdlets

Get existing log profile

```
Get-AzureRmLogProfile
```

Add a log profile

```
Add-AzureRmLogProfile -Name my_log_profile -StorageAccountId
/subscriptions/s1/resourceGroups/myrg1/providers/Microsoft.Storage/storageAccounts/my_storage -serviceBusRuleId
/subscriptions/s1/resourceGroups/Default-ServiceBus-
EastUS/providers/Microsoft.ServiceBus/namespaces/mytestSB/authorizationrules/RootManageSharedAccessKey -
Locations global,westus,eastus -RetentionInDays 90 -Categories Write,Delete,Action
```

PROPERTY	REQUIRED	DESCRIPTION
Name	Yes	Name of your log profile.
StorageAccountId	No	Resource ID of the Storage Account to which the Activity Log should be saved.
serviceBusRuleId	No	Service Bus Rule ID for the Service Bus namespace you would like to have event hubs created in. Is a string with this format: `{service bus resource ID}/authorizationrules/{key name}` .
Locations	Yes	Comma-separated list of regions for which you would like to collect Activity Log events.

PROPERTY	REQUIRED	DESCRIPTION
RetentionInDays	Yes	Number of days for which events should be retained, between 1 and 2147483647. A value of zero stores the logs indefinitely (forever).
Categories	No	Comma-separated list of event categories that should be collected. Possible values are Write, Delete, and Action.

Remove a log profile

```
Remove-AzureRmLogProfile -name my_log_profile
```

Configure log profiles Using the Azure Cross-Platform CLI

Get existing log profile

```
azure insights logprofile list
```

```
azure insights logprofile get --name my_log_profile
```

The `name` property should be the name of your log profile.

Add a log profile

```
azure insights logprofile add --name my_log_profile --storageId /subscriptions/s1/resourceGroups/insights-integration/providers/Microsoft.Storage/storageAccounts/my_storage --serviceBusRuleId /subscriptions/s1/resourceGroups/Default-ServiceBus-EastUS/providers/Microsoft.ServiceBus/namespaces/mytestSB/authorizationrules/RootManageSharedAccessKey --locations global,westus,eastus,northeurope --retentionInDays 90 –categories Write,Delete,Action
```

PROPERTY	REQUIRED	DESCRIPTION
name	Yes	Name of your log profile.
storageId	No	Resource ID of the Storage Account to which the Activity Log should be saved.
serviceBusRuleId	No	Service Bus Rule ID for the Service Bus namespace you would like to have event hubs created in. Is a string with this format: `{service bus resource ID}/authorizationrules/{key name}` .
locations	Yes	Comma-separated list of regions for which you would like to collect Activity Log events.
retentionInDays	Yes	Number of days for which events should be retained, between 1 and 2147483647. A value of zero stores the logs indefinitely (forever).

PROPERTY	REQUIRED	DESCRIPTION
categories	No	Comma-separated list of event categories that should be collected. Possible values are Write, Delete, and Action.

Remove a log profile

```
azure insights logprofile delete --name my_log_profile
```

Event schema

Each event in the Activity Log has a JSON blob similar to this example:


```
{
  "value": [ {
    "authorization": {
      "action": "microsoft.support/supporttickets/write",
      "role": "Subscription Admin",
      "scope":
"/subscriptions/s1/resourceGroups/MSSupportGroup/providers/microsoft.support/supporttickets/115012112305841"
    },
    "caller": "admin@contoso.com",
    "channels": "Operation",
    "claims": {
      "aud": "https://management.core.windows.net/",
      "iss": "https://sts.windows.net/72f988bf-86f1-41af-91ab-2d7cd011db47/",
      "iat": "1421876371",
      "nbf": "1421876371",
      "exp": "1421880271",
      "ver": "1.0",
      "http://schemas.microsoft.com/identity/claims/tenantid": "1e8d8218-c5e7-4578-9acc-9abbd5d23315 ",
      "http://schemas.microsoft.com/claims/authnmethodsreferences": "pwd",
      "http://schemas.microsoft.com/identity/claims/objectidentifier": "2468adf0-8211-44e3-95xq-85137af64708",
      "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn": "admin@contoso.com",
      "puid": "20030000801A118C",
      "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier":
"9vckmEGF7zDKk1YzIY8k0t1_EAPaXoeHyPRn6f413zM",
      "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname": "John",
      "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname": "Smith",
      "name": "John Smith",
      "groups": "cacfe77c-e058-4712-83qw-f9b08849fd60,7f71d11d-4c41-4b23-99d2-d32ce7aa621c,31522864-0578-4ea0-
9gdc-e66cc564d18c",
      "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name": " admin@contoso.com",
      "appid": "c44b4083-3bq0-49c1-b47d-974e53cbdf3c",
      "appidacr": "2",
      "http://schemas.microsoft.com/identity/claims/scope": "user_impersonation",
      "http://schemas.microsoft.com/claims/authnclassreference": "1"
    },
    "correlationId": "1e121103-0ba6-4300-ac9d-952bb5d0c80f",
    "description": "",
    "eventDataId": "44ade6b4-3813-45e6-ae27-7420a95fa2f8",
    "eventName": {
      "value": "EndRequest",
      "localizedValue": "End request"
    },
    "eventSource": {
      "value": "Microsoft.Resources",
      "localizedValue": "Microsoft Resources"
    },
    "httpRequest": {
      "clientRequestId": "27003b25-91d3-418f-8eb1-29e537dcb249",
      "clientIpAddress": "192.168.35.115",
      "method": "PUT"
```

```
      "method": "PUT"
    },
    "id":
"/subscriptions/s1/resourceGroups/MSSupportGroup/providers/microsoft.support/supporttickets/115012112305841/even
ts/44ade6b4-3813-45e6-ae27-7420a95fa2f8/ticks/635574752669792776",
    "level": "Informational",
    "resourceGroupName": "MSSupportGroup",
    "resourceProviderName": {
      "value": "microsoft.support",
      "localizedValue": "microsoft.support"
    },
    "resourceUri":
"/subscriptions/s1/resourceGroups/MSSupportGroup/providers/microsoft.support/supporttickets/115012112305841",
    "operationId": "1e121103-0ba6-4300-ac9d-952bb5d0c80f",
    "operationName": {
      "value": "microsoft.support/supporttickets/write",
      "localizedValue": "microsoft.support/supporttickets/write"
    },
    "properties": {
      "statusCode": "Created"
    },
    "status": {
      "value": "Succeeded",
      "localizedValue": "Succeeded"
    },
    "subStatus": {
      "value": "Created",
      "localizedValue": "Created (HTTP Status Code: 201)"
    },
    "eventTimestamp": "2015-01-21T22:14:26.9792776Z",
    "submissionTimestamp": "2015-01-21T22:14:39.9936304Z",
    "subscriptionId": "s1"
  } ],
"nextLink": "https://management.azure.com/########-####-####-####-############$skiptoken=######"
}
```


ELEMENT NAME	DESCRIPTION
authorization	Blob of RBAC properties of the event. Usually includes the “action”, “role” and “scope” properties.
caller	Email address of the user who has performed the operation, UPN claim, or SPN claim based on availability.
channels	One of the following values: “Admin”, “Operation”
correlationId	Usually a GUID in the string format. Events that share a correlationId belong to the same uber action.
description	Static text description of an event.
eventDataId	Unique identifier of an event.
eventSource	Name of the Azure service or infrastructure that has generated this event.
httpRequest	Blob describing the Http Request. Usually includes the “clientRequestId”, “clientIpAddress” and “method” (HTTP method. For example, PUT).
level	Level of the event. One of the following values: “Critical”, “Error”, “Warning”, “Informational” and “Verbose”

ELEMENT NAME	DESCRIPTION
resourceGroupName	Name of the resource group for the impacted resource.
resourceProviderName	Name of the resource provider for the impacted resource
resourceUri	Resource id of the impacted resource.
operationId	A GUID shared among the events that correspond to a single operation.
operationName	Name of the operation.
properties	Set of `<Key, Value>` pairs (that is, a Dictionary) describing the details of the event.
status	String describing the status of the operation. Some common values are: Started, In Progress, Succeeded, Failed, Active, Resolved.
subStatus	Usually the HTTP status code of the corresponding REST call, but can also include other strings describing a substatus, such as these common values: OK (HTTP Status Code: 200), Created (HTTP Status Code: 201), Accepted (HTTP Status Code: 202), No Content (HTTP Status Code: 204), Bad Request (HTTP Status Code: 400), Not Found (HTTP Status Code: 404), Conflict (HTTP Status Code: 409), Internal Server Error (HTTP Status Code: 500), Service Unavailable (HTTP Status Code: 503), Gateway Timeout (HTTP Status Code: 504).
eventTimestamp	Timestamp when the event was generated by the Azure service processing the request corresponding the event.
submissionTimestamp	Timestamp when the event became available for querying.
subscriptionId	Azure Subscription Id.
nextLink	Continuation token to fetch the next set of results when they are broken up into multiple responses. Typically needed when there are more than 200 records.

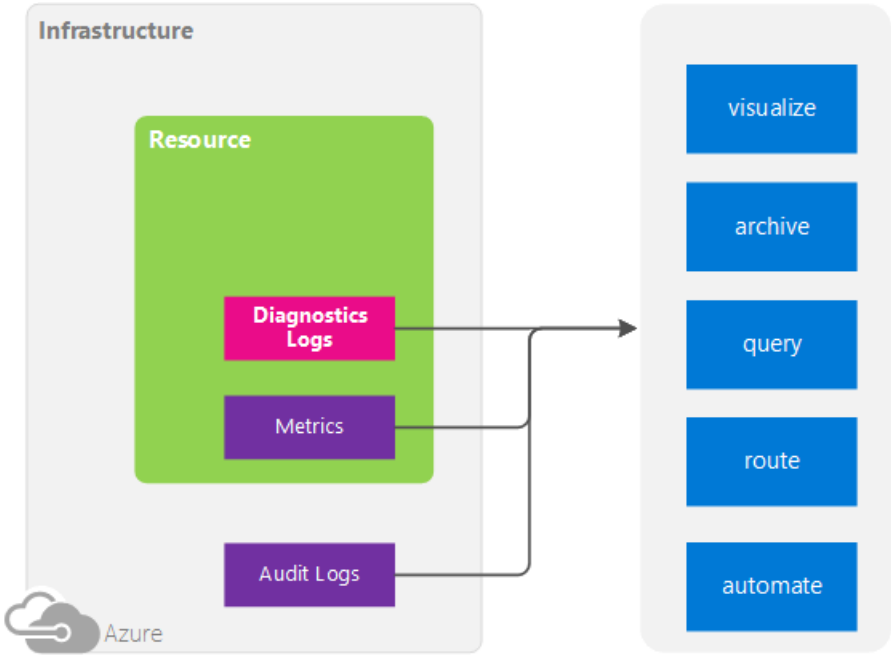
Next Steps

- Learn more about the Activity Log (formerly Audit Logs)
- Stream the Azure Activity Log to Event Hubs

Overview of Azure Diagnostic Logs

1/17/2017 • 7 min to read • Edit on GitHub

Azure Diagnostic Logs are logs emitted by a resource that provide rich, frequent data about the operation of that resource. The content of these logs varies by resource type (for example, Windows event system logs are one category of Diagnostic Log for VMs and blob, table, and queue logs are categories of Diagnostic Logs for storage accounts) and differ from the Activity Log (formerly known as Audit Log or Operational Log), which provides insight into the operations that were performed on resources in your subscription. Not all resources support the new type of Diagnostic Logs described here. The list of Supported Services below shows which resource types support the new Diagnostic Logs.



What you can do with Diagnostic Logs

Here are some of the things you can do with Diagnostic Logs:

- Save them to a **Storage Account** for auditing or manual inspection. You can specify the retention time (in days) using the **Diagnostic Settings**.
- Stream them to **Event Hubs** for ingestion by a third-party service or custom analytics solution such as PowerBI.
- Analyze them with OMS Log Analytics

The storage account or event hub namespace does not have to be in the same subscription as the resource emitting logs as long as the user who configures the setting has appropriate RBAC access to both subscriptions.

Diagnostic Settings

Diagnostic Logs for non-Compute resources are configured using Diagnostic Settings. **Diagnostic Settings** for a resource control:

- Where Diagnostic Logs are sent (Storage Account, Event Hubs, and/or OMS Log Analytics).
- Which Log Categories are sent.
- How long each log category should be retained in a Storage Account – a retention of zero days means that

logs are kept forever. Otherwise, this value can range from 1 to 2147483647. If retention policies are set but storing logs in a Storage Account is disabled (for example if only Event Hubs or OMS options are selected), the retention policies have no effect. Retention policies are applied per-day, so at the end of a day (UTC), logs from the day that is now beyond the retention policy will be deleted. For example, if you had a retention policy of one day, at the beginning of the day today the logs from the day before yesterday would be deleted.

These settings are easily configured via the Diagnostics blade for a resource in the Azure portal, via Azure PowerShell and CLI commands, or via the Azure Monitor REST API.

> **WARNING**
> Diagnostic logs and metrics for Compute resources (for example, VMs or Service Fabric) use a separate mechanism for configuration and selection of outputs.

How to enable collection of Diagnostic Logs

Collection of Diagnostic Logs can be enabled as part of creating a resource or after a resource is created via the resource's blade in the Portal. You can also enable Diagnostic Logs at any point using Azure PowerShell or CLI commands, or using the Azure Monitor REST API.

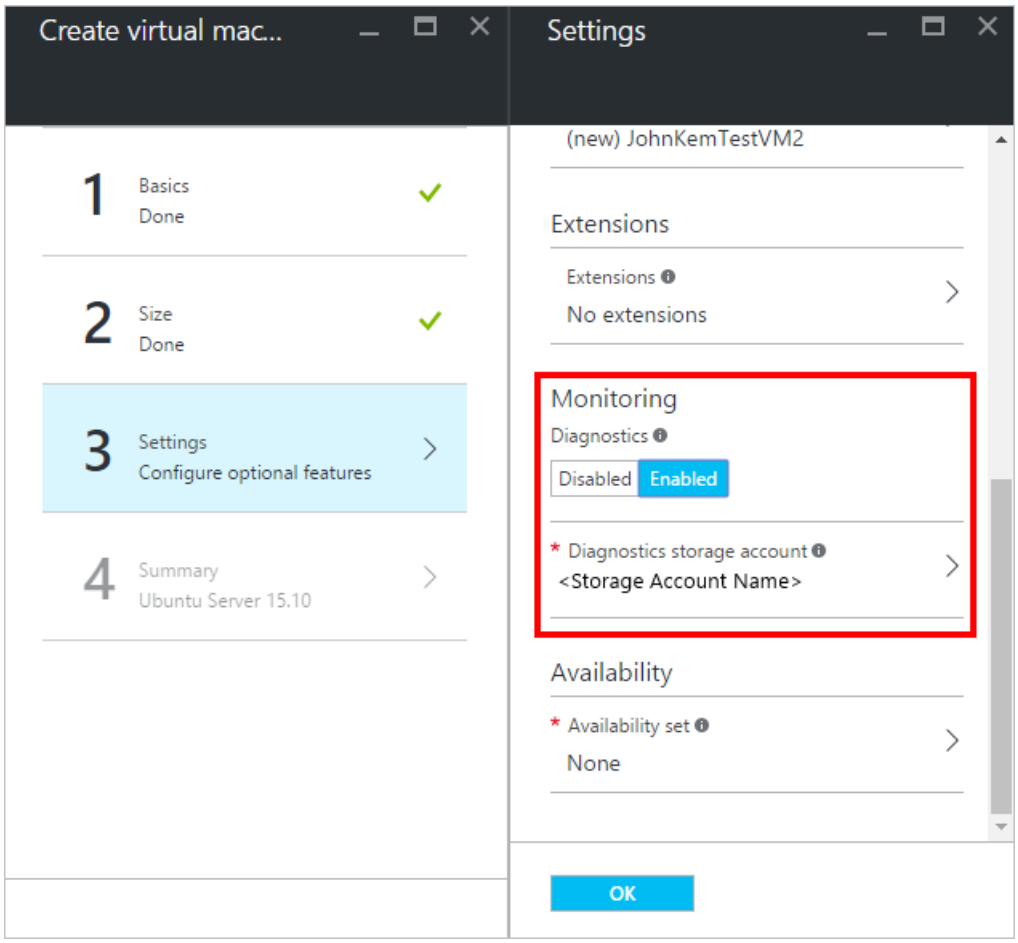
> **TIP**
> These instructions may not apply directly to every resource. See the schema links at the bottom of this page to understand special steps that may apply to certain resource types.

This article shows how you can use a resource template to enable Diagnostic Settings when creating a resource

Enable Diagnostic Logs in the portal

You can enable Diagnostic Logs in the Azure portal when you create compute resource types by enabling the Windows or Linux Azure Diagnostics extension:

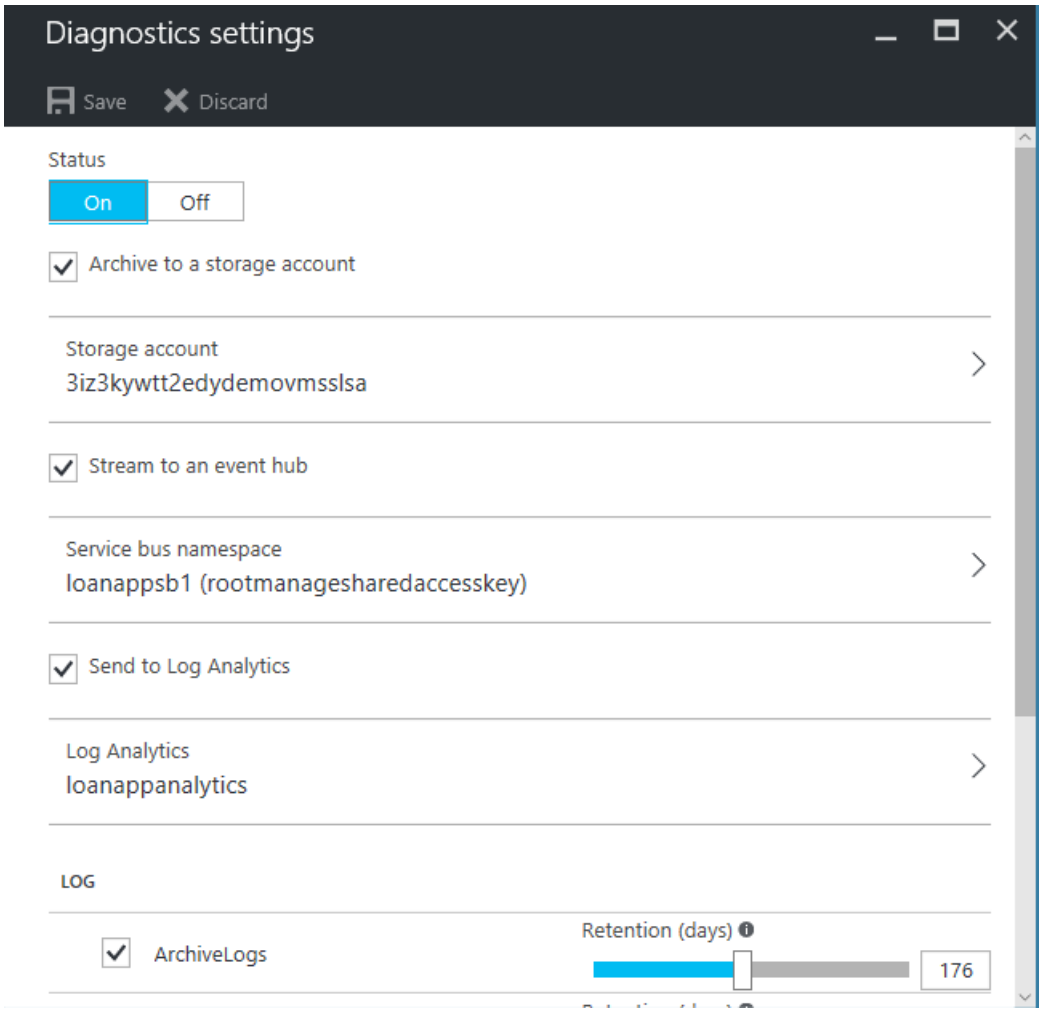
1. Go to **New** and choose the resource you are interested in.
2. After configuring the basic settings and selecting a size, in the **Settings** blade, under **Monitoring**, select **Enabled** and choose a storage account where you would like to store the Diagnostic Logs. You are charged normal data rates for storage and transactions when you send diagnostics to a storage account.



3. Click **OK** and create the resource.

For non-compute resources, you can enable Diagnostic Logs in the Azure portal after a resource has been created by doing the following:

1. Go to the blade for the resource and open the **Diagnostics** blade.
2. Click **On** and pick a Storage Account and/or Event Hub.



3. Under **Logs**, select which **Log Categories** you would like to collect or stream.
4. Click **Save**.

Enable Diagnostic Logs via PowerShell

To enable Diagnostic Logs via the Azure PowerShell Cmdlets, use the following commands.

To enable storage of Diagnostic Logs in a Storage Account, use this command:

```
Set-AzureRmDiagnosticSetting -ResourceId [your resource id] -StorageAccountId [your storage account id] -
Enabled $true
```

The Storage Account ID is the resource id for the storage account to which you want to send the logs.

To enable streaming of Diagnostic Logs to an Event Hub, use this command:

```
Set-AzureRmDiagnosticSetting -ResourceId [your resource id] -ServiceBusRuleId [your service bus rule id] -
Enabled $true
```

The Service Bus Rule ID is a string with this format: `{service bus resource ID}/authorizationrules/{key name}`.

To enable sending of Diagnostic Logs to a Log Analytics workspace, use this command:

```
Set-AzureRmDiagnosticSetting -ResourceId [your resource id] -WorkspaceId [resource id of the log analytics
workspace] -Enabled $true
```

You can obtain the resource id of your Log Analytics workspace using the following command:

```
(Get-AzureRmOperationalInsightsWorkspace).ResourceId
```

You can combine these parameters to enable multiple output options.

Enable Diagnostic Logs via CLI

To enable Diagnostic Logs via the Azure CLI, use the following commands:

To enable storage of Diagnostic Logs in a Storage Account, use this command:

```
azure insights diagnostic set --resourceId <resourceId> --storageId <storageAccountId> --enabled true
```

The Storage Account ID is the resource id for the storage account to which you want to send the logs.

To enable streaming of Diagnostic Logs to an Event Hub, use this command:

```
azure insights diagnostic set --resourceId <resourceId> --serviceBusRuleId <serviceBusRuleId> --enabled true
```

The Service Bus Rule ID is a string with this format: `{service bus resource ID}/authorizationrules/{key name}`.

To enable sending of Diagnostic Logs to a Log Analytics workspace, use this command:

```
azure insights diagnostic set --resourceId <resourceId> --workspaceId <resource id of the log analytics
workspace> --enabled true
```

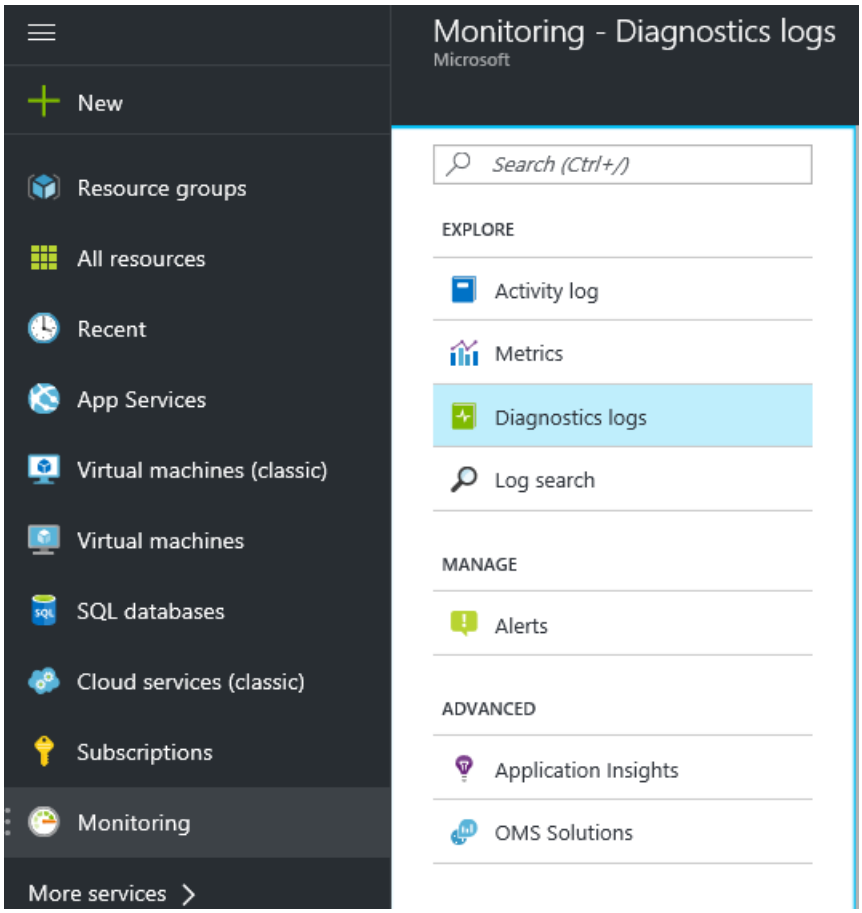
You can combine these parameters to enable multiple output options.

Enable Diagnostic Logs via REST API

To change Diagnostic Settings using the Azure Monitor REST API, see this document.

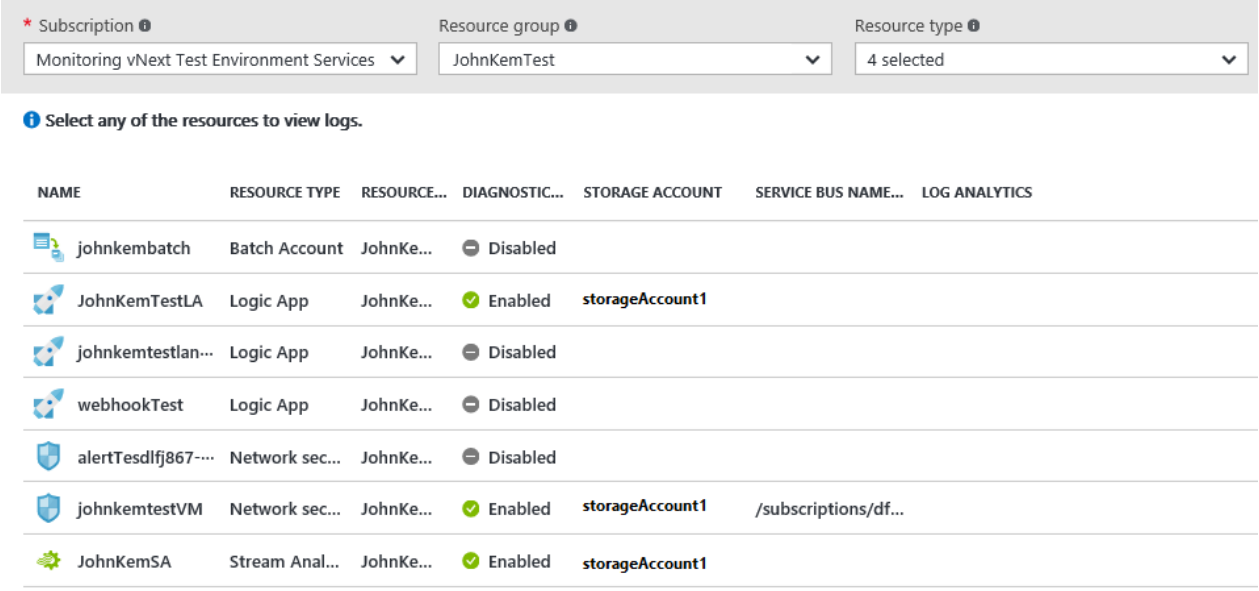
Manage Diagnostic Settings in the portal

In order to ensure that all of your resources are correctly set up with diagnostic settings, you can navigate to the **Monitoring** blade in the portal and open the **Diagnostic Logs** blade.

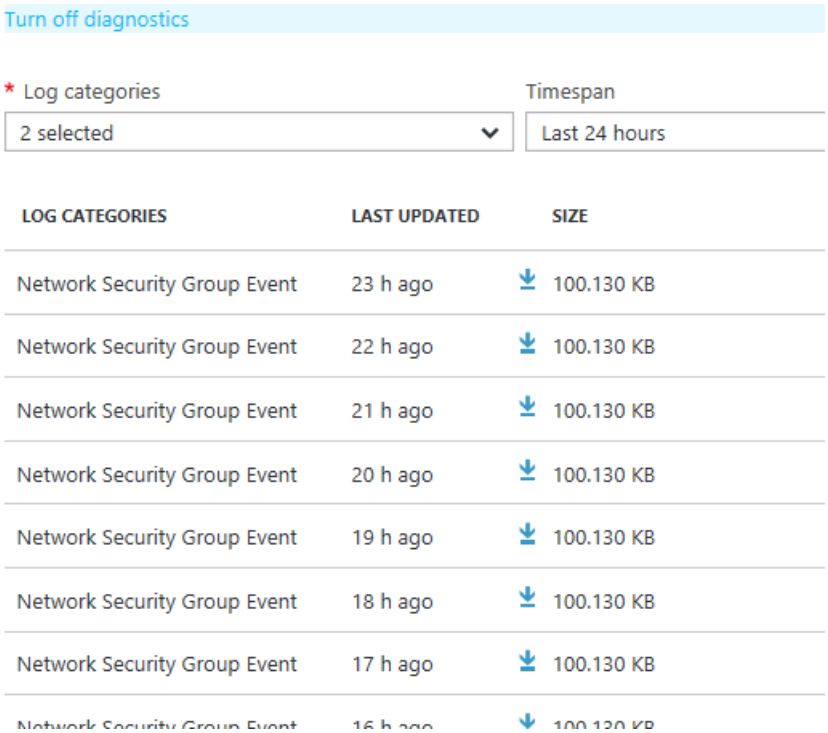


You may have to click "More services" to find the Monitoring blade.

In this blade, you can view and filter all resources that support Diagnostic Logs to see if they have diagnostics enabled and which storage account, event hub, and/or Log Analytics workspace those logs are flowing to.



Clicking on a resource will show all logs that have been stored in the storage account and give you the option to turn off or modify the diagnostic settings. Click the download icon to download logs for a particular time period.



> **NOTE**
>
> Diagnostic logs will only appear in this view and be available for download if you have configured diagnostic settings to save them to a storage account.

Clicking on the link for **Diagnostic Settings** will bring up the Diagnostic Settings blade, where you can enable, disable, or modify your diagnostic settings for the selected resource.

Supported services and schema for Diagnostic Logs

The schema for Diagnostic Logs varies depending on the resource and log category. Below are the supported services and their schema.

SERVICE	SCHEMA & DOCS
Load Balancer	Log analytics for Azure Load Balancer (Preview)
Network Security Groups	Log analytics for network security groups (NSGs)
Application Gateways	Diagnostics Logging for Application Gateway
Key Vault	Azure Key Vault Logging
Azure Search	Enabling and using Search Traffic Analytics
Data Lake Store	Accessing diagnostic logs for Azure Data Lake Store
Data Lake Analytics	Accessing diagnostic logs for Azure Data Lake Analytics
Logic Apps	No schema available.
Azure Batch	Azure Batch diagnostic logging
Azure Automation	Log analytics for Azure Automation
Event Hub	No schema available.
Service Bus	No schema available.
Stream Analytics	No schema available.

Supported log categories per resource type

RESOURCE TYPE	CATEGORY	CATEGORY DISPLAY NAME
Microsoft.Automation/automationAccounts	JobLogs	Job Logs
Microsoft.Automation/automationAccounts	JobStreams	Job Streams
Microsoft.Batch/batchAccounts	ServiceLog	Service Logs
Microsoft.DataLakeAnalytics/accounts	Audit	Audit Logs
Microsoft.DataLakeAnalytics/accounts	Requests	Request Logs
Microsoft.DataLakeStore/accounts	Audit	Audit Logs
Microsoft.DataLakeStore/accounts	Requests	Request Logs
Microsoft.EventHub/namespaces	ArchiveLogs	Archive Logs

RESOURCE TYPE	CATEGORY	CATEGORY DISPLAY NAME
Microsoft.EventHub/namespaces	OperationalLogs	Operational Logs
Microsoft.KeyVault/vaults	AuditEvent	Audit Logs
Microsoft.Logic/workflows	WorkflowRuntime	Workflow runtime diagnostic events
Microsoft.Logic/integrationAccounts	IntegrationAccountTrackingEvents	Integration Account track events
Microsoft.Network/networksecuritygroups	NetworkSecurityGroupEvent	Network Security Group Event
Microsoft.Network/networksecuritygroups	NetworkSecurityGroupRuleCounter	Network Security Group Rule Counter
Microsoft.Network/networksecuritygroups	NetworkSecurityGroupFlowEvent	Network Security Group Rule Flow Event
Microsoft.Network/loadBalancers	LoadBalancerAlertEvent	Load Balancer Alert Events
Microsoft.Network/loadBalancers	LoadBalancerProbeHealthStatus	Load Balancer Probe Health Status
Microsoft.Network/applicationGateways	ApplicationGatewayAccessLog	Application Gateway Access Log
Microsoft.Network/applicationGateways	ApplicationGatewayPerformanceLog	Application Gateway Performance Log
Microsoft.Network/applicationGateways	ApplicationGatewayFirewallLog	Application Gateway Firewall Log
Microsoft.Search/searchServices	OperationLogs	Operation Logs
Microsoft.ServerManagement/nodes	RequestLogs	Request Logs
Microsoft.ServiceBus/namespaces	OperationalLogs	Operational Logs
Microsoft.StreamAnalytics/streamingjobs	Execution	Execution
Microsoft.StreamAnalytics/streamingjobs	Authoring	Authoring

Next Steps

- Stream Diagnostic Logs to **Event Hubs**
- Change Diagnostic Settings using the Azure Monitor REST API
- Analyze the logs with OMS Log Analytics

Azure Monitor partner integrations

1/17/2017 • 3 min to read • Edit on GitHub

PARTNERS		
AlertLogic	AppDynamics	Atlassian
CloudMonix	Cloudyn	DataDog
Dynatrace	NewRelic	OpsGenie
PagerDuty	Splunk	Sumo Logic

AlertLogic Log Manager

Alert Logic Log Manager collects VM, Application, and Azure platform logs for security analysis and retention. This includes Azure Audit Logs via the Azure Monitor API. This information is used to detect malfeasance and meet compliance requirements.

Go to the documentation.

AppDynamics

AppDynamics Application Performance Management (APM) enables application owners to rapidly troubleshoot performance bottlenecks and optimize the performance of their applications running in Azure environment. AppDynamics APM is seamlessly integrated with Azure Marketplace and available for monitor Azure Cloud Services (PaaS) (Including web & worker roles), Virtual Machines (IaaS), Remote Service Detection (Microsoft Azure Service Bus), Microsoft Azure Queue Microsoft Azure Remote Services (Azure Blob), Azure Queue (Microsoft

Service Bus), Data Storage, Microsoft Azure Blob Storage.

Go to the documentation.

Atlassian JIRA

You can create JIRA tickets on Azure Monitor alerts.

Go to the documentation.

CloudMonix

CloudMonix offers monitoring, automation and self-healing services for Microsoft Azure platform.

Go to the documentation.

Cloudyn

Cloudyn manages and optimizes multi-platform, hybrid cloud deployments to help enterprises fully realize their cloud potential. The SaaS solution delivers visibility into usage, performance and cost, coupled with insights and actionable recommendations for smart optimization and cloud governance. Cloudyn enables accountability through accurate chargeback and hierarchical cost allocation management. Cloudyn is integrated with Azure Monitoring in order to provide insights and actionable recommendations in order to optimize your Azure deployment.

Go to the documentation.

DataDog

Datadog is the world's leading monitoring service for cloud-scale applications, bringing together data from servers, databases, tools and services to present a unified view of your entire stack. These capabilities are provided on a SaaS-based data analytics platform that enables Dev and Ops teams to work collaboratively to avoid downtime, resolve performance problems, and ensure that development and deployment cycles finish on time. By integrating Datadog and Azure, you can collect and view metrics from across your infrastructure, correlate VM metrics with application-level metrics, and slice and dice your metrics using any combination of properties and custom tags.

Go to the documentation.

Dynatrace

The Dynatrace OneAgent integrates with Azure VMs and App Services via the according Azure extension mechanisms. This way we can gather performance metrics about hosts, network and services. Besides just displaying metrics we visualize environments end-to-end, showing transactions from the client side to the database layer. AI-based correlation of problems and fully integrated root-cause-analysis, including method level insights into code and database, make troubleshooting and performance optimizations much easier.

Go to the documentation.

NewRelic

Learn more.

OpsGenie

OpsGenie acts as a dispatcher for the alerts generated by Azure. OpsGenie determines the right people to notify based on on-call schedules and escalations, by notifying them using email, text messages (SMS), phone calls, push

notifications. Simply, Azure generates alerts for detected problems, and OpsGenie ensures the right people are working on them.

Go to the documentation.

PagerDuty

PagerDuty, the leading incident management solution, has provided first-class support for Azure Alerts on metrics. Today, PagerDuty now supports notifications on Azure Monitor Alerts, Autoscale Notifications, and Audit Log Events, in addition to notifications on platform-level metrics for Azure services. These enhancements give users increased visibility into the core Azure Platform while enabling them to take full advantage of PagerDuty's incident management capabilities for real-time response. Our expanded Azure integration is made possible via webhooks, allowing for quick and easy set-up and customization.

Go to the documentation.

Splunk Add-on for Microsoft Cloud Services

The Splunk Add-on for Microsoft Cloud Services is available in the Splunkbase here.

Go to the documentation.

Sumo Logic

Sumo Logic is a secure, cloud-native, machine data analytics service, delivering real-time, continuous intelligence from structured, semi-structured and unstructured data across the entire application lifecycle and stack. More than 1,000 customers around the globe rely on Sumo Logic for the analytics and insights to build, run and secure their modern applications and cloud infrastructures. With Sumo Logic, customers gain a multi-tenant, service-model advantage to accelerate their shift to continuous innovation, increasing competitive advantage, business value and growth.

Learn more.

Next Steps

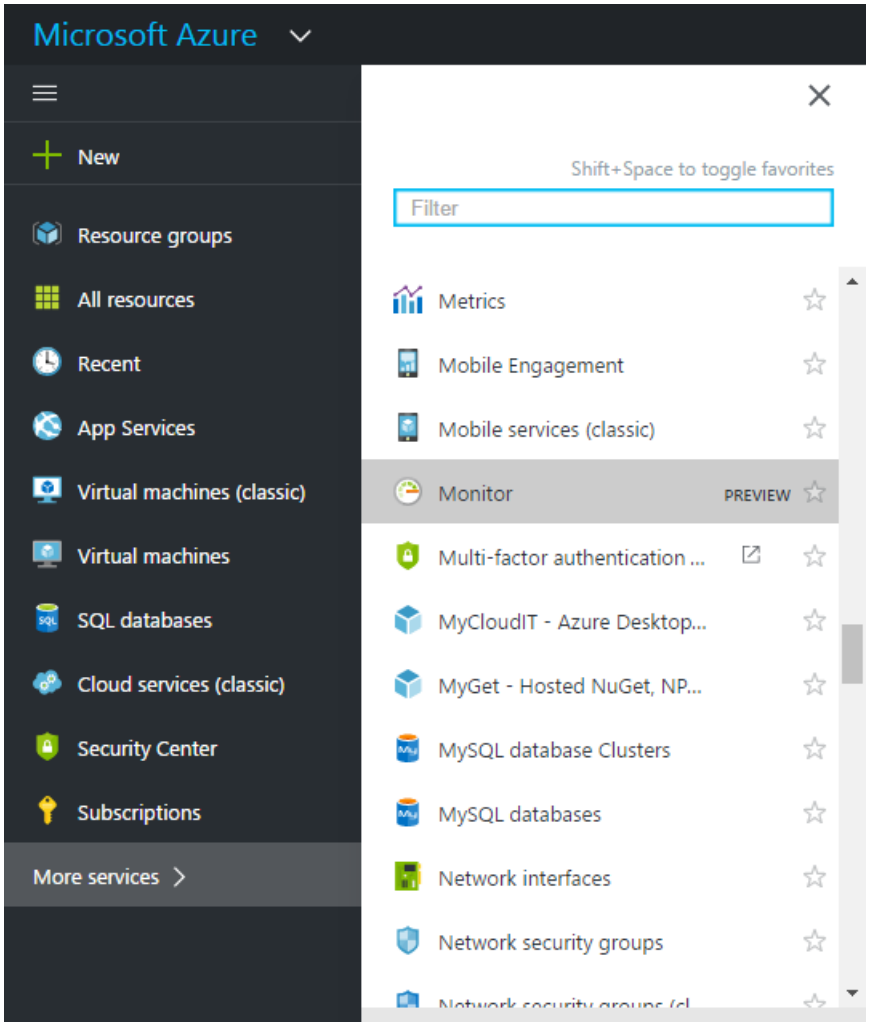
- Learn more about the Activity Log (formerly Audit Logs)
- Stream the Azure Activity Log to Event Hubs

Get started with Azure Monitor

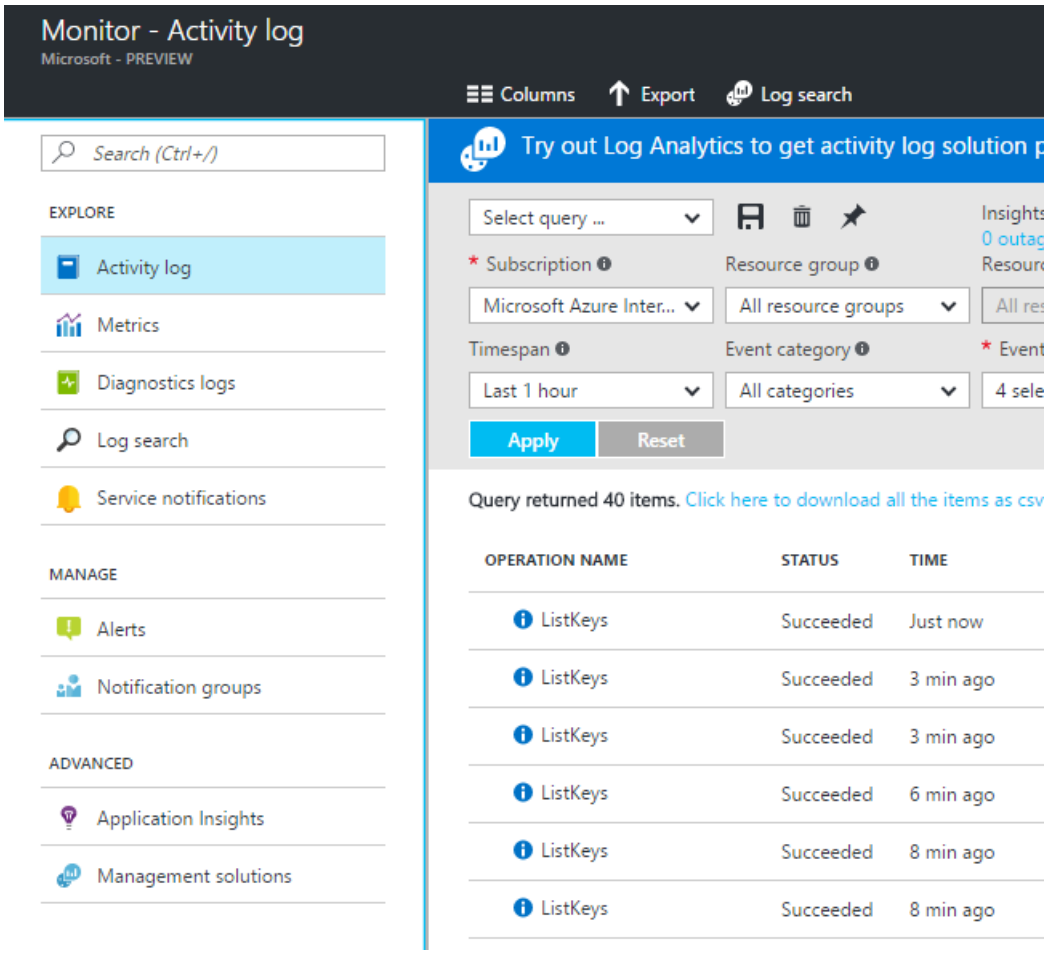
1/17/2017 • 4 min to read • Edit on GitHub

Azure Monitor is the platform service that provides a single source for monitoring Azure resources. With Azure Monitor, you can visualize, query, route, archive, and take action on the metrics and logs coming from resources in Azure. You can work with this data using the Monitor portal blade, Monitor PowerShell Cmdlets, Cross-Platform CLI, or Azure Monitor REST APIs. In this article, we walk through a few of the key components of Azure Monitor, using the portal for demonstration.

1. In the portal, navigate to **More services** and find the **Monitor** option. Click the star icon to add this option to your favorites list so that it is always easily accessible from the left-hand navigation bar.



2. Click the **Monitor** option to open up the **Monitor** blade. This blade brings together all your monitoring settings and data into one consolidated view. It first opens to the **Activity log** section.

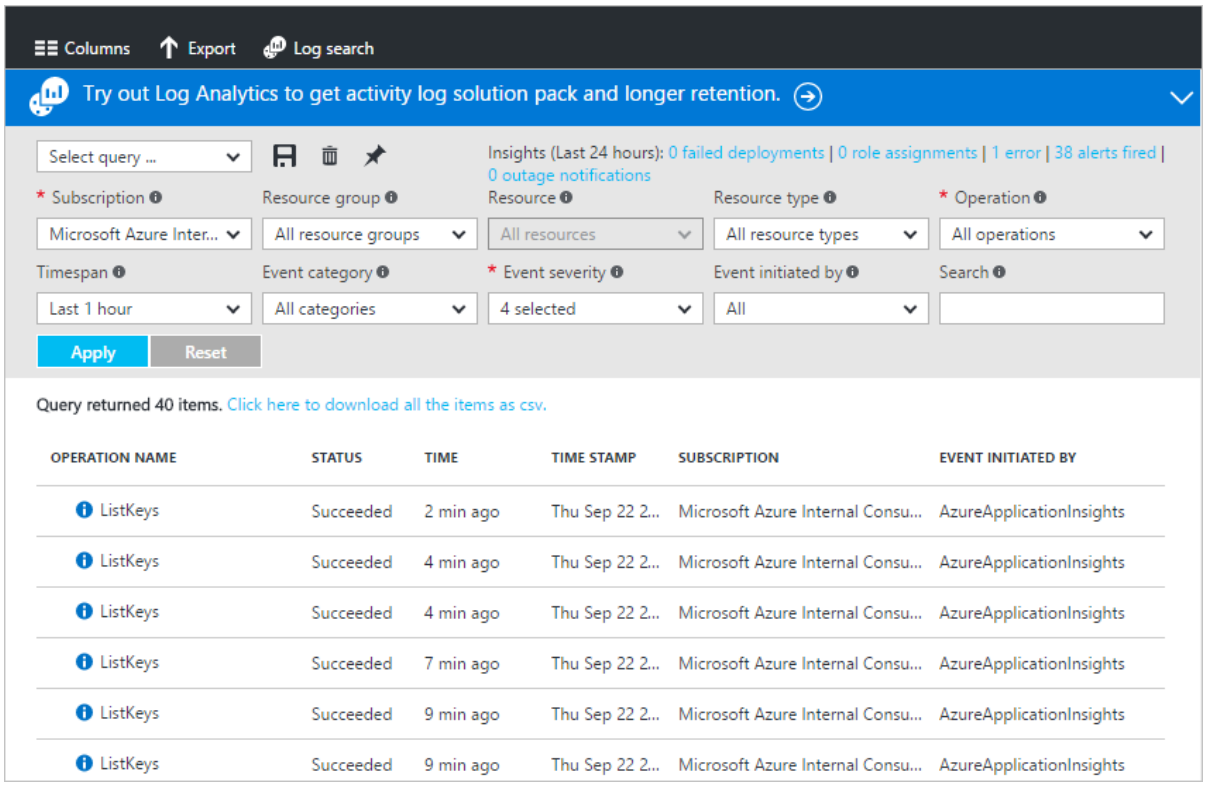


WARNING

The **Service notifications** and **Notification groups** options displayed above will only appear to those who have joined the Private Preview of these features.

Azure Monitor has three basic categories of monitoring data: The **activity log**, **metrics**, and **diagnostic logs**.

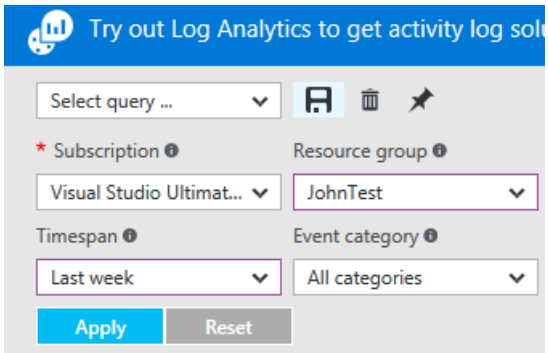
3. Click **Activity log** to ensure that the activity log section is displayed.



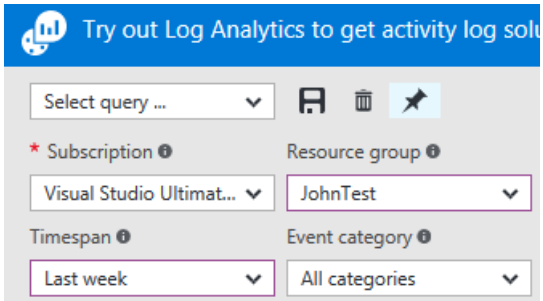
The **activity log** describes all operations performed on resources in your subscription. Using the Activity Log, you can determine the 'what, who, and when' for any create, update, or delete operations on resources in your subscription. For example, the Activity Log tells you when a web app was stopped and who stopped it. Activity Log events are stored in the platform and available to query for 90 days.

You can create and save queries for common filters, then pin the most important queries to a portal dashboard so you'll always know if events that meet your criteria have occurred.

4. Filter the view to a particular resource group over the last week, then click the **Save** button.



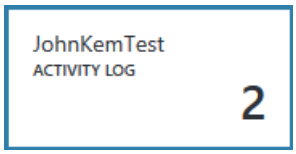
5. Now, click the **Pin** button.



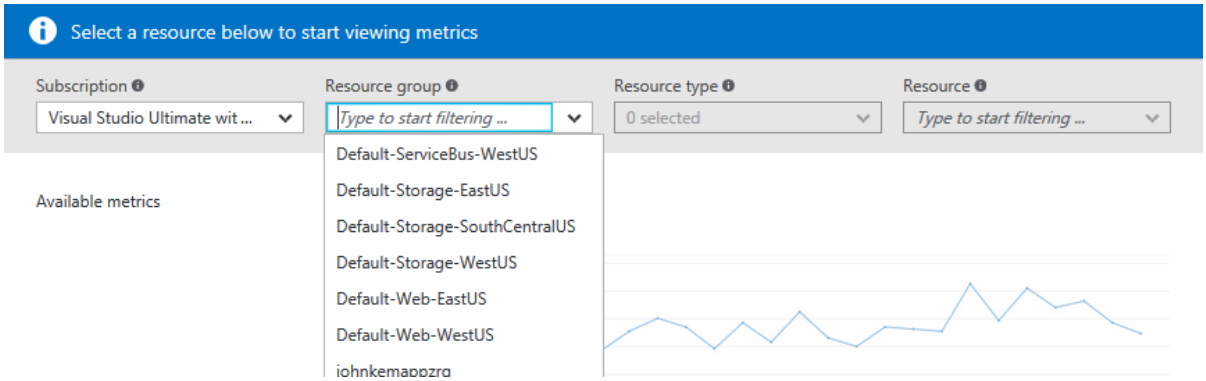
Most of the views in this walkthrough can be pinned to a dashboard. This helps you create a single source of information for operational data on your services.

6. Return to your dashboard. You can now see that the query (and number of results) is displayed in your

dashboard. This is useful if you want to quickly see any high-profile actions that have occurred recently in your subscription, eg. a new role was assigned or a VM was deleted.

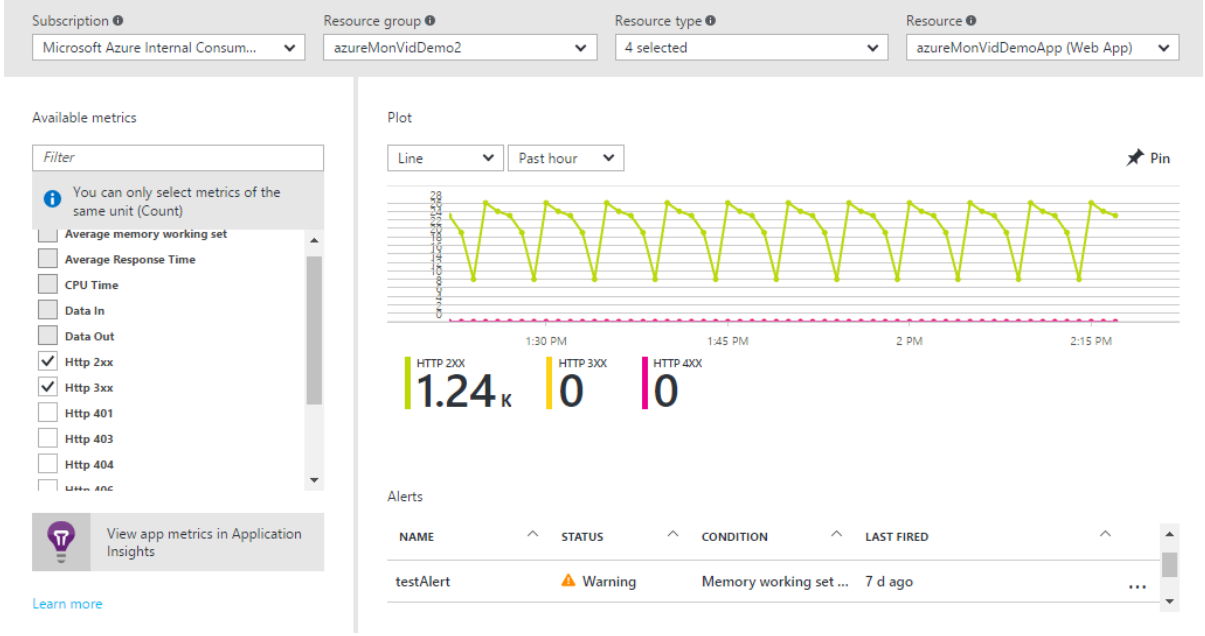


7. Return to the **Monitor** tile and click the **Metrics** section. You first need to select a resource by filtering and selecting using the drop down options at the top of the blade.



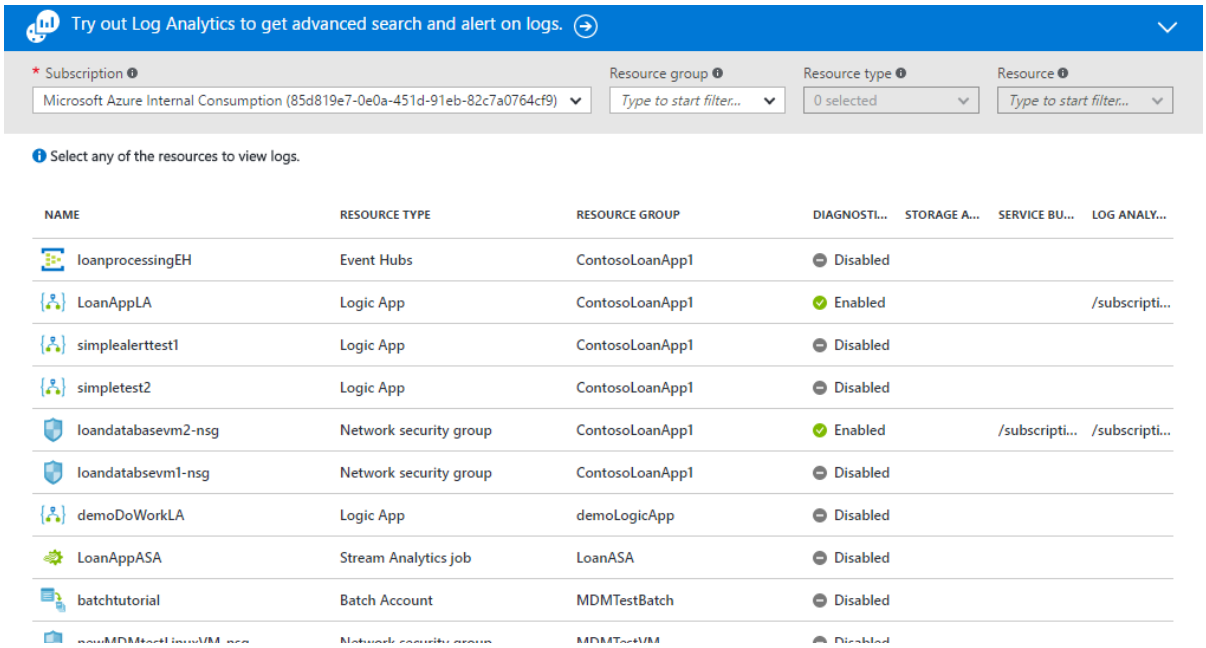
All Azure resources emit **metrics**. This view brings together all metrics in a single pane of glass so you can easily understand how your resources are performing.

8. Once you have selected a resource, all available metrics appear on the left side of the blade. You can chart multiple metrics at once by selecting metrics and modify the graph type and time range. You can also view all metric alerts set on this resource.



> **NOTE**
>
> Some metrics are only available by enabling Application Insights and/or Windows or Linux Azure Diagnostics on your resource.

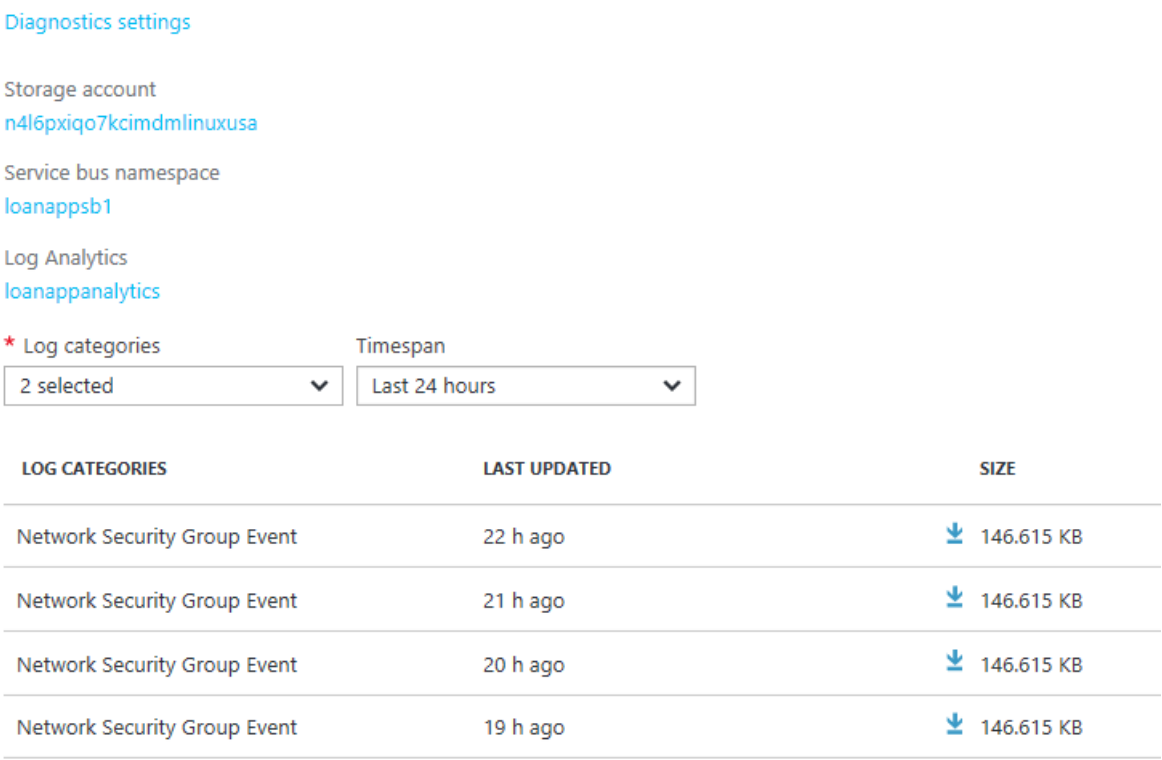
9. When you are happy with your chart, you can use the **Pin** button to pin it to your dashboard.
10. Return to the **Monitor** blade and click **Diagnostic logs**.



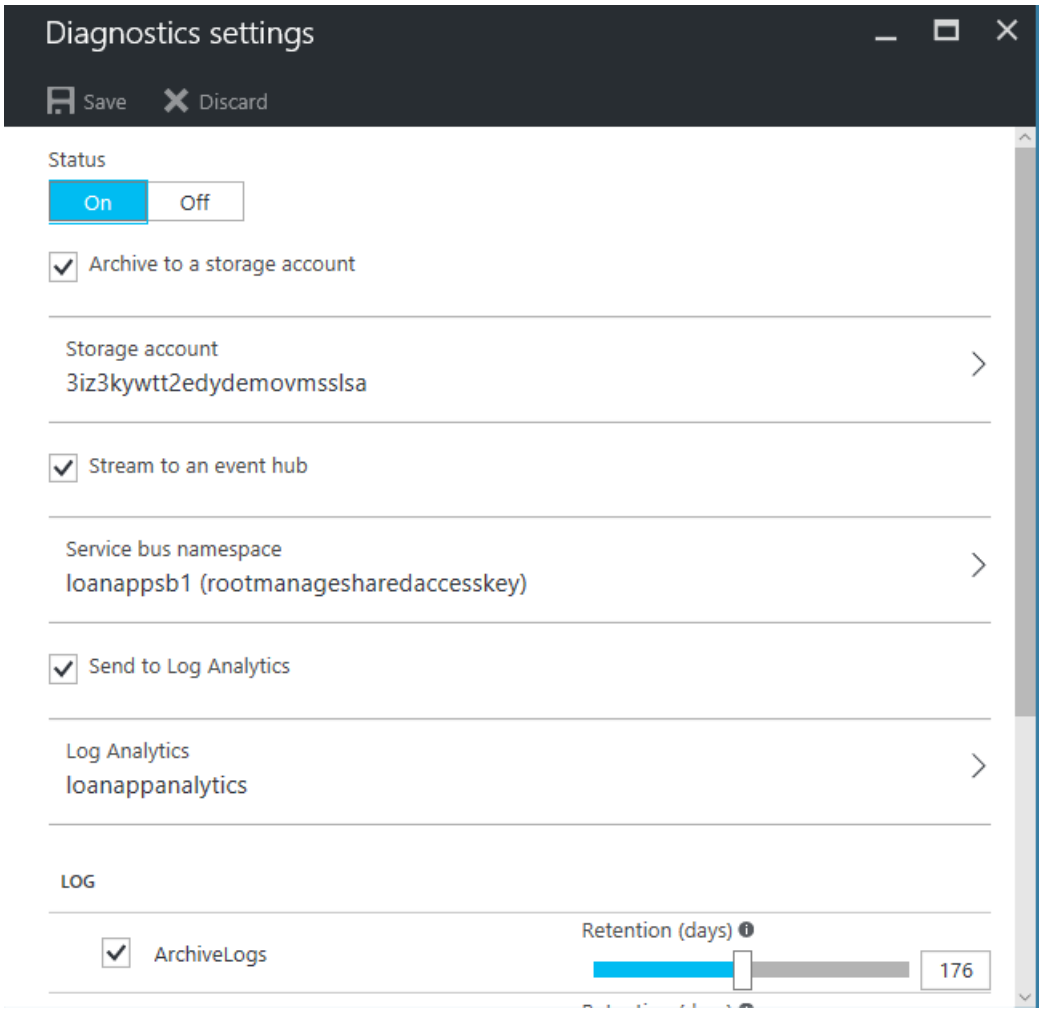
Diagnostic logs are logs emitted *by* a resource that provide data about the operation of that particular resource. For example, Network Security Group Rule Counters and Logic App Workflow Logs are both types of diagnostic logs. These logs can be stored in a storage account, streamed to an Event Hub, and/or sent to Log Analytics. Log Analytics is Microsoft's operational intelligence product for advanced searching and alerting.

In the portal you can view and filter a list of all resources in your subscription to identify if they have diagnostic logs enabled.

11. Click a resource in the diagnostic logs blade. If diagnostic logs are being stored in a storage account, you will see a list of hourly logs that you can directly download.

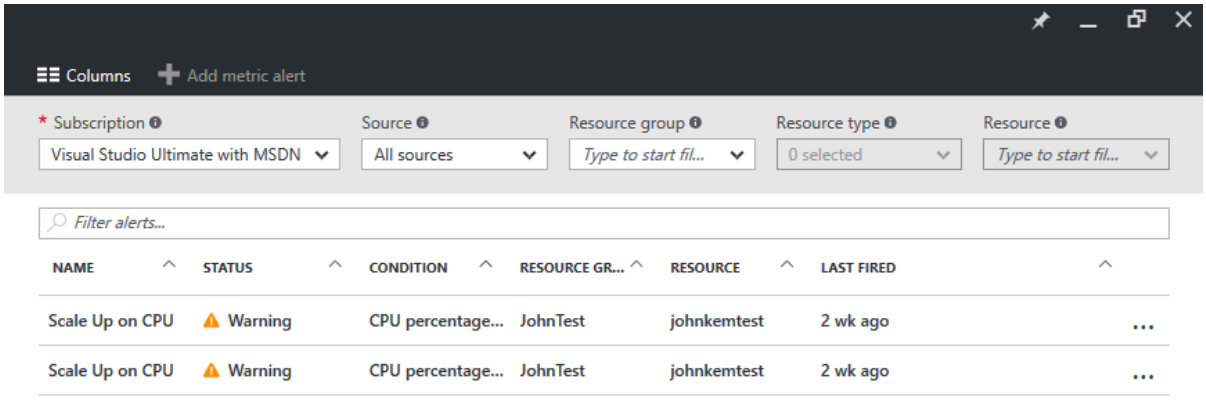


You can also click **Diagnostic Settings**, which allows you to set up or modify your settings for archival to a storage account, streaming to Event Hubs, or sending to a Log Analytics workspace.



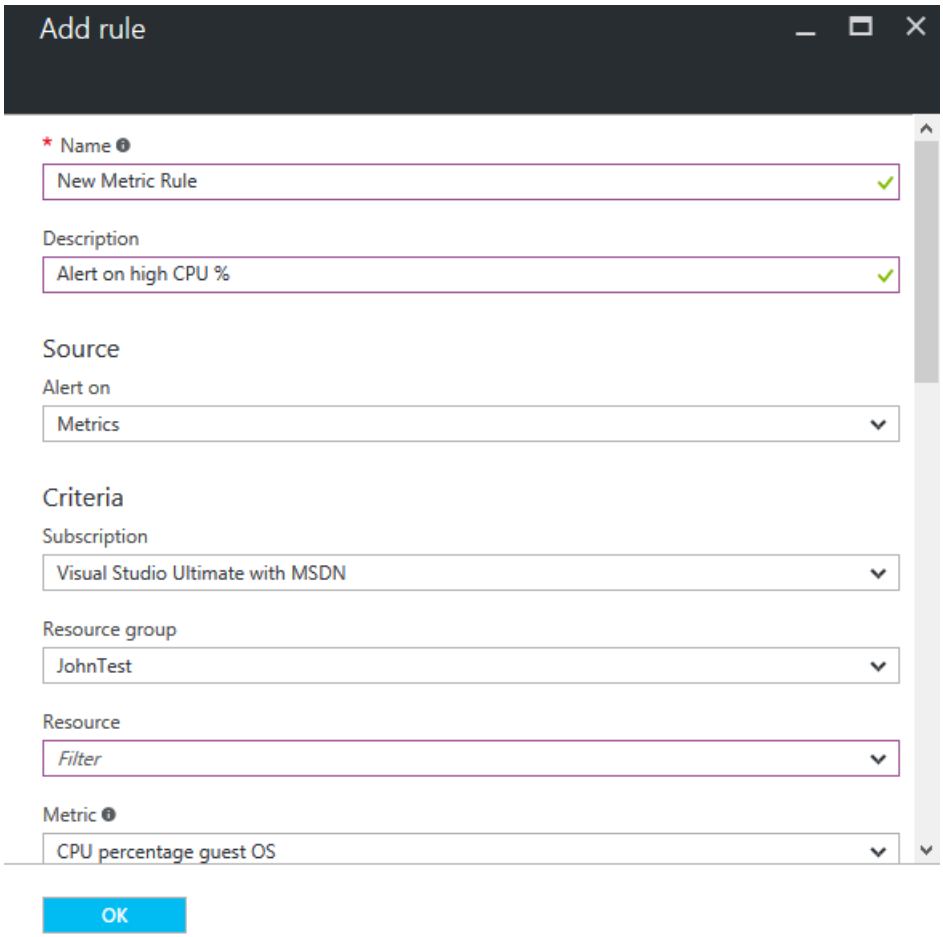
If you have set up diagnostic logs to Log Analytics, you can then search them in the **Log search** section of the portal.

12. Navigate to the **Alerts** section of the Monitor blade.



Here you can manage all **alerts** on your Azure resources. This includes alerts on metrics, activity log events (in private preview), Application Insights web tests (Locations), and Application Insights proactive diagnostics. Alerts can trigger an email to be sent or an HTTP POST to a webhook URL.

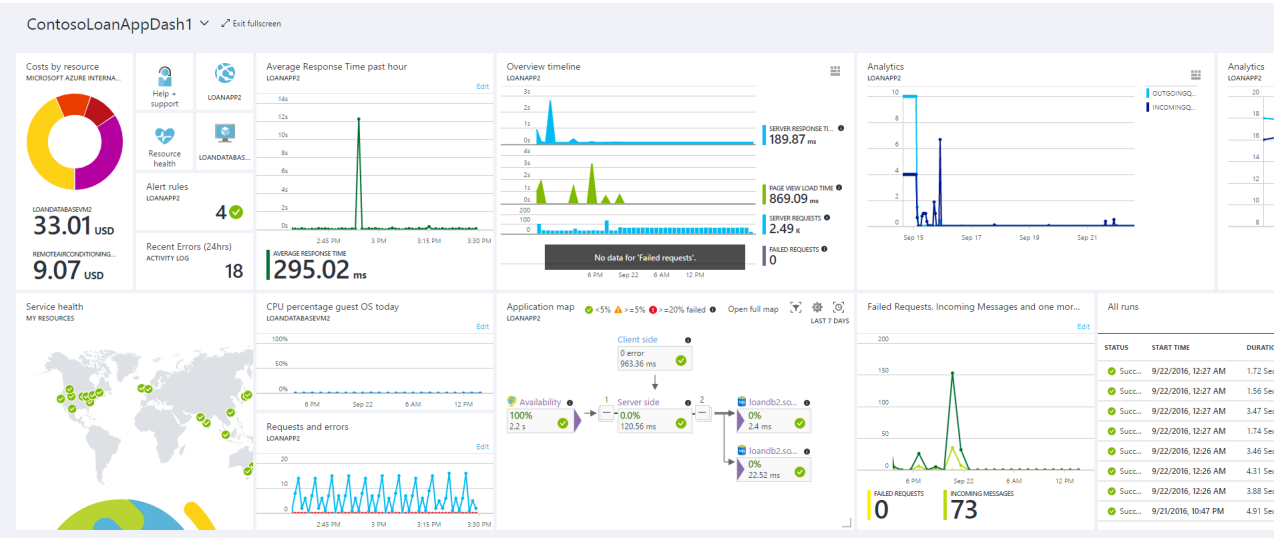
13. Click **Add metric alert** to create an alert.



You can then pin an alert to your dashboard to easily see its state at any time.

14. The Monitor section also includes links to Application Insights applications and Log Analytics management solutions. These other Microsoft products have deep integration with Azure Monitor.
15. If you are not using Application Insights or Log Analytics, chances are that Azure Monitor has a partnership with your current monitoring, logging, and alerting products. See our partners page for a full list and instructions for how to integrate.

By following these steps and pinning all relevant tiles to a dashboard, you can create comprehensive views of your application and infrastructure like this one:



Next Steps

- Read the Overview of Azure Monitor

Get started with roles, permissions, and security with Azure Monitor

1/17/2017 • 7 min to read • Edit on GitHub

Many teams need to strictly regulate access to monitoring data and settings. For example, if you have team members who work exclusively on monitoring (support engineers, devops engineers) or if you use a managed service provider, you may want to grant them access to only monitoring data while restricting their ability to create, modify, or delete resources. This article shows how to quickly apply a built-in monitoring RBAC role to a user in Azure or build your own custom role for a user who needs limited monitoring permissions. It then discusses security considerations for your Azure Monitor-related resources and how you can limit access to the data they contain.

Built-in monitoring roles

Azure Monitor's built-in roles are designed to help limit access to resources in a subscription while still enabling those responsible for monitoring infrastructure to obtain and configure the data they need. Azure Monitor provides two out-of-the-box roles: A Monitoring Reader and a Monitoring Contributor.

Monitoring Reader

People assigned the Monitoring Reader role can view all monitoring data in a subscription but cannot modify any resource or edit any settings related to monitoring resources. This role is appropriate for users in an organization, such as support or operations engineers, who need to be able to:

- View monitoring dashboards in the portal and create their own private monitoring dashboards.
- Query for metrics using the Azure Monitor REST API, PowerShell cmdlets, or cross-platform CLI.
- Query the Activity Log using the portal, Azure Monitor REST API, PowerShell cmdlets, or cross-platform CLI.
- View the diagnostic settings for a resource.
- View the log profile for a subscription.
- View autoscale settings.
- View alert activity and settings.
- Access Application Insights data and view data in AI Analytics.
- Search Log Analytics (OMS) workspace data including usage data for the workspace.
- View Log Analytics (OMS) management groups.
- Retrieve the Log Analytics (OMS) search schema.
- List Log Analytics (OMS) intelligence packs.
- Retrieve and execute Log Analytics (OMS) saved searches.
- Retrieve the Log Analytics (OMS) storage configuration.

> **NOTE**
>
> This role does not give read access to log data that has been streamed to an event hub or stored in a storage account. See below for information on configuring access to these resources.

Monitoring Contributor

People assigned the Monitoring Contributor role can view all monitoring data in a subscription and create or modify monitoring settings, but cannot modify any other resources. This role is a superset of the Monitoring Reader role, and is appropriate for members of an organization's monitoring team or managed service providers

who, in addition to the permissions above, also need to be able to:

- Publish monitoring dashboards as a shared dashboard.
- Set diagnostic settings for a resource.*
- Set the log profile for a subscription.*
- Set alert activity and settings.
- Create Application Insights web tests and components.
- List Log Analytics (OMS) workspace shared keys.
- Enable or disable Log Analytics (OMS) intelligence packs.
- Create and delete and execute Log Analytics (OMS) saved searches.
- Create and delete the Log Analytics (OMS) storage configuration.

*user must also separately be granted ListKeys permission on the target resource (storage account or event hub namespace) to set a log profile or diagnostic setting.

> **NOTE**
> This role does not give read access to log data that has been streamed to an event hub or stored in a storage account. See below for information on configuring access to these resources.

Monitoring permissions and custom RBAC roles

If the above built-in roles don't meet the exact needs of your team, you can create a custom RBAC role with more granular permissions. Below are the common Azure Monitor RBAC operations with their descriptions.

OPERATION	DESCRIPTION
Microsoft.Insights/AlertRules/[Read, Write, Delete]	Read/write/delete alert rules.
Microsoft.Insights/AlertRules/Incidents/Read	List incidents (history of the alert rule being triggered) for alert rules. This only applies to the portal.
Microsoft.Insights/AutoscaleSettings/[Read, Write, Delete]	Read/write/delete autoscale settings.
Microsoft.Insights/DiagnosticSettings/[Read, Write, Delete]	Read/write/delete diagnostic settings.
Microsoft.Insights/eventtypes/digestevents/Read	This permission is necessary for users who need access to Activity Logs via the portal.
Microsoft.Insights/eventtypes/values/Read	List Activity Log events (management events) in a subscription. This permission is applicable to both programmatic and portal access to the Activity Log.
Microsoft.Insights/LogDefinitions/Read	This permission is necessary for users who need access to Activity Logs via the portal.
Microsoft.Insights/MetricDefinitions/Read	Read metric definitions (list of available metric types for a resource).
Microsoft.Insights/Metrics/Read	Read metrics for a resource.

> **NOTE**
> Access to alerts, diagnostic settings, and metrics for a resource requires that the user has Read access to the resource type and scope of that resource. Creating ("write") a diagnostic setting or log profile that archives to a storage account or streams to event hubs requires the user to also have ListKeys permission on the target resource.

For example, using the above table you could create a custom RBAC role for an "Activity Log Reader" like this:

```
$role = Get-AzureRmRoleDefinition "Reader"
$role.Id = $null
$role.Name = "Activity Log Reader"
$role.Description = "Can view activity logs."
$role.Actions.Clear()
$role.Actions.Add("Microsoft.Insights/eventtypes/*")
$role.AssignableScopes.Clear()
$role.AssignableScopes.Add("/subscriptions/mySubscription")
New-AzureRmRoleDefinition -Role $role
```

Security considerations for monitoring data

Monitoring data—particularly log files—can contain sensitive information, such as IP addresses or user names. Monitoring data from Azure comes in three basic forms:

1. The Activity Log, which describes all control-plane actions on your Azure subscription.
2. Diagnostic Logs, which are logs emitted by a resource.
3. Metrics, which are emitted by resources.

All three of these data types can be stored in a storage account or streamed to Event Hub, both of which are general-purpose Azure resources. Because these are general-purpose resources, creating, deleting, and accessing them is a privileged operation usually reserved for an administrator. We suggest that you use the following practices for monitoring-related resources to prevent misuse:

- Use a single, dedicated storage account for monitoring data. If you need to separate monitoring data into multiple storage accounts, never share usage of a storage account between monitoring and non-monitoring data, as this may inadvertently give those who only need access to monitoring data (eg. a third-party SIEM) access to non-monitoring data.
- Use a single, dedicated Service Bus or Event Hub namespace across all diagnostic settings for the same reason as above.
- Limit access to monitoring-related storage accounts or event hubs by keeping them in a separate resource group, and use scope on your monitoring roles to limit access to only that resource group.
- Never grant the ListKeys permission for either storage accounts or event hubs at subscription scope when a user only needs access to monitoring data. Instead, give these permissions to the user at a resource or resource group (if you have a dedicated monitoring resource group) scope.

Limiting access to monitoring-related storage accounts

When a user or application needs access to monitoring data in a storage account, you should generate an Account SAS on the storage account that contains monitoring data with service-level read-only access to blob storage. In PowerShell, this might look like:

```
$context = New-AzureStorageContext -ConnectionString "[connection string for your monitoring Storage Account]"
$token = New-AzureStorageAccountSASToken -ResourceType Service -Service Blob -Permission "rl" -Context $context
```

You can then give the token to the entity that needs to read from that storage account, and it can list and read from all blobs in that storage account.

Alternatively, if you need to control this permission with RBAC, you can grant that entity the Microsoft.Storage/storageAccounts/listkeys/action permission on that particular storage account. This is necessary for users who need to be able to set a diagnostic setting or log profile to archive to a storage account. For example, you could create the following custom RBAC role for a user or application that only needs to read from one storage account:

```
$role = Get-AzureRmRoleDefinition "Reader"
$role.Id = $null
$role.Name = "Monitoring Storage Account Reader"
$role.Description = "Can get the storage account keys for a monitoring storage account."
$role.Actions.Clear()
$role.Actions.Add("Microsoft.Storage/storageAccounts/listkeys/action")
$role.Actions.Add("Microsoft.Storage/storageAccounts/Read")
$role.AssignableScopes.Clear()
$role.AssignableScopes.Add("/subscriptions/mySubscription/resourceGroups/myResourceGroup/providers/Microsoft.Stor
age/storageAccounts/myMonitoringStorageAccount")
New-AzureRmRoleDefinition -Role $role
```

WARNING

The ListKeys permission enables the user to list the primary and secondary storage account keys. These keys grant the user all signed permissions (read, write, create blobs, delete blobs, etc.) across all signed services (blob, queue, table, file) in that storage account. We recommend using an Account SAS described above when possible.

Limiting access to monitoring-related event hubs

A similar pattern can be followed with event hubs, but first you need to create a dedicated Listen authorization rule. If you want to grant access to an application that only needs to listen to monitoring-related event hubs, do the following:

1. Create a shared access policy on the event hub(s) that were created for streaming monitoring data with only Listen claims. This can be done in the portal. For example, you might call it "monitoringReadOnly." If possible, you will want to give that key directly to the consumer and skip the next step.
2. If the consumer needs to be able to get the key ad-hoc, grant the user the ListKeys action for that event hub. This is also necessary for users who need to be able to set a diagnostic setting or log profile to stream to event hubs. For example, you might create an RBAC rule:

   ```
   $role = Get-AzureRmRoleDefinition "Reader"
   $role.Id = $null
   $role.Name = "Monitoring Event Hub Listener"
   $role.Description = "Can get the key to listen to an event hub streaming monitoring data."
   $role.Actions.Clear()
   $role.Actions.Add("Microsoft.ServiceBus/namespaces/authorizationrules/listkeys/action")
   $role.Actions.Add("Microsoft.ServiceBus/namespaces/Read")
   $role.AssignableScopes.Clear()
   $role.AssignableScopes.Add("/subscriptions/mySubscription/resourceGroups/myResourceGroup/providers/Microso
   ft.ServiceBus/namespaces/mySBNameSpace")
   New-AzureRmRoleDefinition -Role $role
   ```

Next steps

- Read about RBAC and permissions in Resource Manager
- Read the overview of monitoring in Azure

Use Azure portal to create alerts for Azure services

1/17/2017 • 2 min to read • Edit on GitHub

Overview

This article shows you how to set up Azure alerts using the Azure portal.

You can receive an alert based on monitoring metrics for, or events on, your Azure services.

- **Metric values** - The alert triggers when the value of a specified metric crosses a threshold you assign in either direction. That is, it triggers both when the condition is first met and then afterwards when that condition is no longer being met.
- **Activity log events** - An alert can trigger on *every* event, or, only when a certain number of events occur.

You can configure an alert to do the following when it triggers:

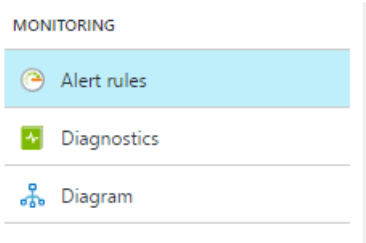
- send email notifications to the service administrator and co-administrators
- send email to additional emails that you specify.
- call a webhook
- start execution of an Azure runbook (only from the Azure portal)

You can configure and get information about alert rules using

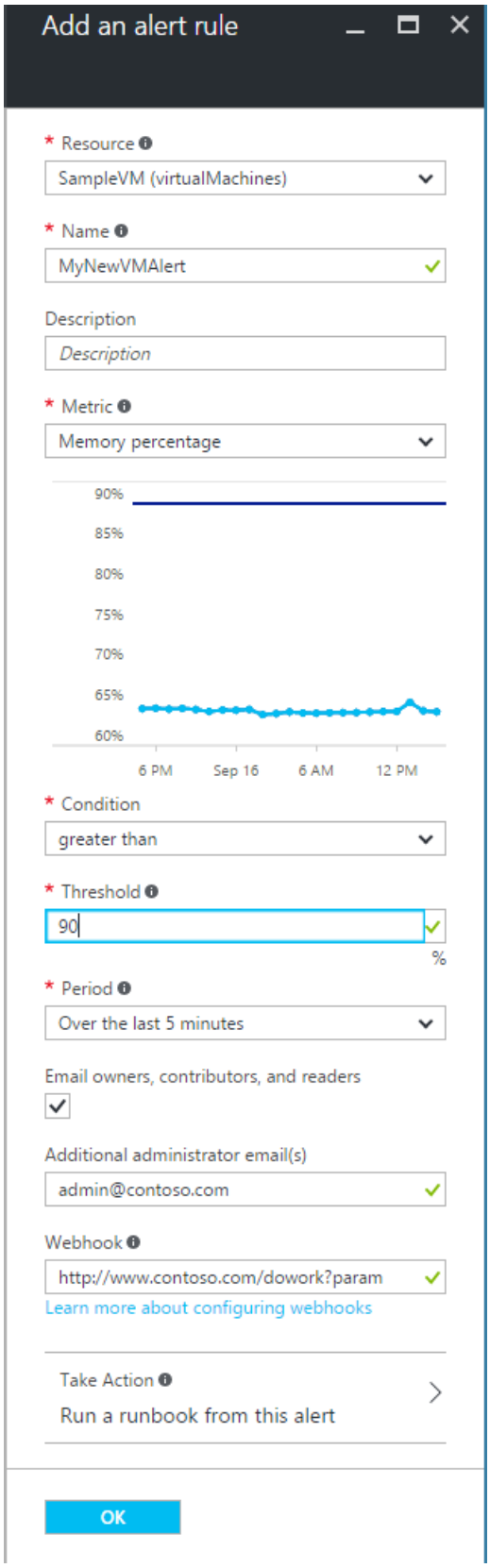
- Azure portal
- PowerShell
- command-line interface (CLI)
- Azure Monitor REST API

Create an alert rule on a metric with the Azure portal

1. In the portal, locate the resource you are interested in monitoring and select it.
2. Select **Alerts** or **Alert rules** under the MONITORING section. The text and icon may vary slightly for different resources.



3. Select the **Add alert** command and fill in the fields.



4. **Name** your alert rule, and choose a **Description**, which also shows in notification emails.
5. Select the **Metric** you want to monitor, then choose a **Condition** and **Threshold** value for the metric. Also chose the **Period** of time that the metric rule must be satisfied before the alert triggers. So for example, if you use the period "PT5M" and your alert looks for CPU above 80%, the alert triggers when the CPU has been consistently above 80% for 5 minutes. Once the first trigger occurs, it again triggers when the CPU stays below 80% for 5 minutes. The CPU measurement occurs every 1 minute.
6. Check **Email owners...** if you want administrators and co-administrators to be emailed when the alert fires.
7. If you want additional emails to receive a notification when the alert fires, add them in the **Additional Administrator email(s)** field. Separate multiple emails with semi-colons - *email@contoso.com;email2@contoso.com*

8. Put in a valid URI in the **Webhook** field if you want it called when the alert fires.
9. If you use Azure Automation, you can select a Runbook to be run when the alert fires.
10. Select **OK** when done to create the alert.

Within a few minutes, the alert is active and triggers as previously described.

Managing your alerts

Once you have created an alert, you can select it and:

- View a graph showing the metric threshold and the actual values from the previous day.
- Edit or delete it.
- **Disable** or **Enable** it if you want to temporarily stop or resume receiving notifications for that alert.

Next steps

- Get an overview of Azure monitoring including the types of information you can collect and monitor.
- Learn more about configuring webhooks in alerts.
- Learn more about Azure Automation Runbooks.
- Get an overview of diagnostic logs and collect detailed high-frequency metrics on your service.
- Get an overview of metrics collection to make sure your service is available and responsive.

Use the cross-platform Command Line Interface (CLI) to create alerts for Azure services

1/17/2017 • 4 min to read • Edit on GitHub

Overview

This article shows you how to set up Azure alerts using the Command Line Interface (CLI).

> **NOTE**
> Azure Monitor is the new name for what was called "Azure Insights" until Sept 25th, 2016. However, the namespaces and thus the commands below still contain the "insights".

You can receive an alert based on monitoring metrics for, or events on, your Azure services.

- **Metric values** - The alert triggers when the value of a specified metric crosses a threshold you assign in either direction. That is, it triggers both when the condition is first met and then afterwards when that condition is no longer being met.
- **Activity log events** - An alert can trigger on *every* event, or, only when a certain number of events occur.

You can configure an alert to do the following when it triggers:

- send email notifications to the service administrator and co-administrators
- send email to additional emails that you specify.
- call a webhook
- start execution of an Azure runbook (only from the Azure portal at this time)

You can configure and get information about alert rules using

- Azure portal
- PowerShell
- command-line interface (CLI)
- Azure Monitor REST API

You can always receive help for commands by typing a command and putting -help at the end. For example:

```console
azure insights alerts -help
azure insights alerts actions email create -help
```

Create alert rules using the CLI

1. Perform the Prerequisites and login to Azure. See Azure Monitor CLI samples. In short, install the CLI and run these commands. They get you logged in, show what subscription you are using, and prepare you to run Azure Monitor commands.

   ```
   azure login
   azure account show
   azure config mode arm
   ```

2. To list existing rules on a resource group, use the following form **azure insights alerts rule list** *[options] <resourceGroup>*

```
azure insights alerts rule list myresourcegroupname
```

3. To create a rule, you need to have several important pieces of information first.

   - The **Resource ID** for the resource you want to set an alert for
   - The **metric definitions** available for that resource

   One way to get the Resource ID is to use the Azure portal. Assuming the resource is already created, select it in the portal. Then in the next blade, select *Properties* under the *Settings* section. The *RESOURCE ID* is a field in the next blade. Another way is to use the Azure Resource Explorer.

   An example resource id for a web app is

```
/subscriptions/dedededе-7aa0-407d-a6fb-
eb20c8bd1192/resourceGroups/myresourcegroupname/providers/Microsoft.Web/sites/mywebsitename
```

   To get a list of the available metrics and units for those metrics for the previous resource example, use the following CLI command:

```
azure insights metrics list /subscriptions/dedededе-7aa0-407d-a6fb-
eb20c8bd1192/resourceGroups/myresourcegroupname/providers/Microsoft.Web/sites/mywebsitename PT1M
```

   PT1M is the granularity of the available measurement (1-minute intervals). Using different granularities gives you different metric options.

4. To create a metric-based alert rule, use a command of the following form:

   azure insights alerts rule metric set *[options] <ruleName> <location> <resourceGroup> <windowSize> <operator> <threshold> <targetResourceId> <metricName> <timeAggregationOperator>*

   The following example sets up an alert on a web site resource. The alert triggers whenever it consistently receives any traffic for 5 minutes and again when it receives no traffic for 5 minutes.

```
 azure insights alerts rule metric set myrule eastus myreasourcegroup PT5M GreaterThan 2
/subscriptions/dedededе-7aa0-407d-a6fb-
eb20c8bd1192/resourceGroups/myresourcegroupname/providers/Microsoft.Web/sites/mywebsitename BytesReceived
Total
```

5. To create webhook or send email when an alert fires, first create the email and/or webhooks. Then create the rule immediately afterwards. You cannot associate webhook or emails with already created rules using the CLI.

```
 azure insights alerts actions email create --customEmails myemail@contoso.com

 azure insights alerts actions webhook create https://www.contoso.com

 azure insights alerts rule metric set myrulewithwebhookandemail eastus myreasourcegroup PT5M GreaterThan
2 /subscriptions/dedededе-7aa0-407d-a6fb-
eb20c8bd1192/resourceGroups/myresourcegroupname/providers/Microsoft.Web/sites/mywebsitename BytesReceived
Total
```

6. To create an alert that fires on a specific condition in the activity log, use the form:

   insights alerts rule log set *[options] <ruleName> <location> <resourceGroup> <operationName>*

For example

```
 azure insights alerts rule log set myActivityLogRule eastus myresourceGroupName
Microsoft.Storage/storageAccounts/listKeys/action
```

The operationName corresponds to an event type for an entry in the activity log. Examples include *Microsoft.Compute/virtualMachines/delete* and *microsoft.insights/diagnosticSettings/write*.

You can use the PowerShell command Get-AzureRmProviderOperation to obtain a list of possible operationNames. Alternately, you can use the Azure portal to query the Activity log and find specific past operations that you want to create an alert for. The operations shown in the graphic log view of the friendly names. Look in the JSON for the entry and pull out the OperationName value.

7. You can verify that your alerts have been created properly by looking at an individual rule.

```
azure insights alerts rule list myresourcegroup --ruleName myrule
```

8. To delete rules, use a command of the form:

   insights alerts rule delete [options] <resourceGroup> <ruleName>

   These commands delete the rules previously created in this article.

```
azure insights alerts rule delete myresourcegroup myrule
azure insights alerts rule delete myresourcegroup myrulewithwebhookandemail
azure insights alerts rule delete myresourcegroup myActivityLogRule
```

Next steps

- Get an overview of Azure monitoring including the types of information you can collect and monitor.
- Learn more about configuring webhooks in alerts.
- Learn more about Azure Automation Runbooks.
- Get an overview of collecting diagnostic logs to collect detailed high-frequency metrics on your service.
- Get an overview of metrics collection to make sure your service is available and responsive.

Use PowerShell to create alerts for Azure services

1/17/2017 • 3 min to read • Edit on GitHub

Overview

This article shows you how to set up Azure alerts using PowerShell.

You can receive an alert based on monitoring metrics for, or events on, your Azure services.

- **Metric values** - The alert triggers when the value of a specified metric crosses a threshold you assign in either direction. That is, it triggers both when the condition is first met and then afterwards when that condition is no longer being met.
- **Activity log events** - An alert can trigger on *every* event, or, only when a certain number of events occur.

You can configure an alert to do the following when it triggers:

- send email notifications to the service administrator and co-administrators
- send email to additional emails that you specify.
- call a webhook
- start execution of an Azure runbook (only from the Azure portal)

You can configure and get information about alert rules using

- Azure portal
- PowerShell
- command-line interface (CLI)
- Azure Monitor REST API

For additional information, you can always type `get-help` and then the PowerShell command you want help on.

Create alert rules in PowerShell

1. Log in to Azure.

```
Login-AzureRmAccount
```

2. Get a list of the subscriptions you have available. Verify that you are working with the right subscription. If not, set it to the right one using the output from `Get-AzureRmSubscription`.

```
Get-AzureRmSubscription
Get-AzureRmContext
Set-AzureRmContext -SubscriptionId <subscriptionid>
```

3. To list existing rules on a resource group, use the following command:

```
Get-AzureRmAlertRule -ResourceGroup <myresourcegroup> -DetailedOutput
```

4. To create a rule, you need to have several important pieces of information first.

   - The **Resource ID** for the resource you want to set an alert for
   - The **metric definitions** available for that resource

One way to get the Resource ID is to use the Azure portal. Assuming the resource is already created, select it in the portal. Then in the next blade, select *Properties* under the *Settings* section. The RESOURCE ID is a field in the next blade. Another way is to use the Azure Resource Explorer.

An example resource id for a web app is

```
/subscriptions/dedededе-7aa0-407d-a6fb-
eb20c8bd1192/resourceGroups/myresourcegroupname/providers/Microsoft.Web/sites/mywebsitename
```

You can use `Get-AzureRmMetricDefinition` to view the list of all metric definitions for a specific resource.

```
Get-AzureRmMetricDefinition -ResourceId <resource_id>
```

The following example generates a table with the metric Name and the Unit for that metric.

```
Get-AzureRmMetricDefinition -ResourceId <resource_id> | Format-Table -Property Name,Unit
```

A full list of available options for Get-AzureRmMetricDefinition is available by running Get-MetricDefinitions.

5. The following example sets up an alert on a web site resource. The alert triggers whenever it consistently receives any traffic for 5 minutes and again when it receives no traffic for 5 minutes.

```
 Add-AzureRmMetricAlertRule -Name myMetricRuleWithWebhookAndEmail -Location "East US" -ResourceGroup
myresourcegroup -TargetResourceId /subscriptions/dedededе-7aa0-407d-a6fb-
eb20c8bd1192/resourceGroups/myresourcegroupname/providers/Microsoft.Web/sites/mywebsitename -MetricName
"BytesReceived" -Operator GreaterThan -Threshold 2 -WindowSize 00:05:00 -TimeAggregationOperator Total -
Description "alert on any website activity"
```

6. To create webhook or send email when an alert triggers, first create the email and/or webhooks. Then immediately create the rule afterwards with the -Actions tag and as shown in the following example. You cannot associate webhook or emails with already created rules via PowerShell.

```
 $actionEmail = New-AzureRmAlertRuleEmail -CustomEmail myname@company.com
 $actionWebhook = New-AzureRmAlertRuleWebhook -ServiceUri https://www.contoso.com?token=mytoken

 Add-AzureRmMetricAlertRule -Name myMetricRuleWithWebhookAndEmail -Location "East US" -ResourceGroup
myresourcegroup -TargetResourceId /subscriptions/dedededе-7aa0-407d-a6fb-
eb20c8bd1192/resourceGroups/myresourcegroupname/providers/Microsoft.Web/sites/mywebsitename -MetricName
"BytesReceived" -Operator GreaterThan -Threshold 2 -WindowSize 00:05:00 -TimeAggregationOperator Total -
Actions $actionEmail, $actionWebhook -Description "alert on any website activity"
```

7. To create an alert that triggers on a specific condition in the activity log, use commands of the following form

```
 $actionEmail = New-AzureRmAlertRuleEmail -CustomEmail myname@company.com
 $actionWebhook = New-AzureRmAlertRuleWebhook -ServiceUri https://www.contoso.com?token=mytoken

 Add-AzureRmLogAlertRule -Name myLogAlertRule -Location "East US" -ResourceGroup myresourcegroup -
OperationName microsoft.web/sites/start/action -Status Succeeded -TargetResourceGroup
resourcegroupbeingmonitored -Actions $actionEmail, $actionWebhook
```

The -OperationName corresponds to a type of event in the activity log. Examples include *Microsoft.Compute/virtualMachines/delete* and *microsoft.insights/diagnosticSettings/write*.

You can use the PowerShell command Get-AzureRmProviderOperation to obtain a list of possible

operationNames. Alternately, you can use the Azure portal to query the Activity log and find specific past operations that you want to create an alert for. The operations shown in the graphic log view of the friendly names. Look in the JSON for the entry and pull out the OperationName value.

8. Verify that your alerts have been created properly by looking at the individual rules.

```
 Get-AzureRmAlertRule -Name myMetricRuleWithWebhookAndEmail -ResourceGroup myresourcegroup -
DetailedOutput

 Get-AzureRmAlertRule -Name myLogAlertRule -ResourceGroup myresourcegroup -DetailedOutput
```

9. Delete your alerts. These commands delete the rules created previously in this article.

```
 Remove-AzureRmAlertRule -ResourceGroup myresourcegroup -Name myrule
 Remove-AzureRmAlertRule -ResourceGroup myresourcegroup -Name myMetricRuleWithWebhookAndEmail
 Remove-AzureRmAlertRule -ResourceGroup myresourcegroup -Name myLogAlertRule
```

Next steps

- Get an overview of Azure monitoring including the types of information you can collect and monitor.
- Learn more about configuring webhooks in alerts.
- Learn more about Azure Automation Runbooks.
- Get an overview of collecting diagnostic logs to collect detailed high-frequency metrics on your service.
- Get an overview of metrics collection to make sure your service is available and responsive.

Configure a webhook on an Azure metric alert

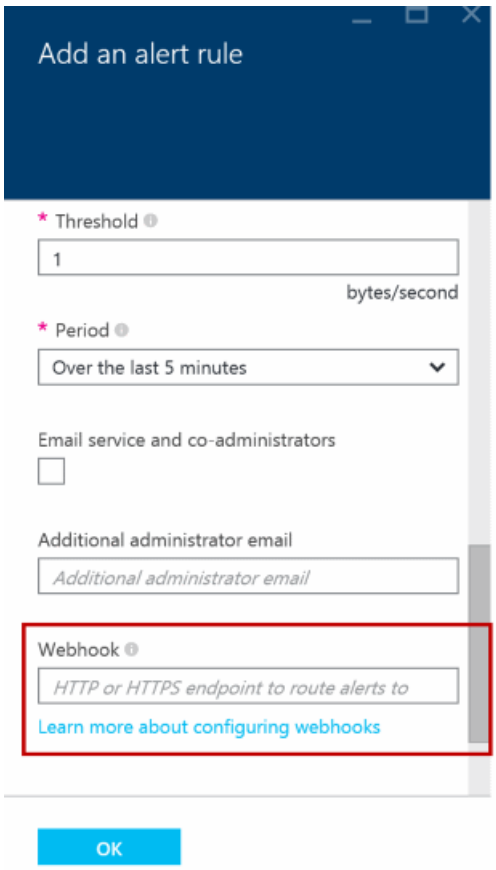
1/17/2017 • 3 min to read • Edit on GitHub

Webhooks allow you to route an Azure alert notification to other systems for post-processing or custom actions. You can use a webhook on an alert to route it to services that send SMS, log bugs, notify a team via chat/messaging services, or do any number of other actions. This article describes how to set a webhook on an Azure metric alert and what the payload for the HTTP POST to a webhook looks like. For information on the setup and schema for an Azure Activity Log alert (alert on events), see this page instead.

Azure alerts HTTP POST the alert contents in JSON format, schema defined below, to a webhook URI that you provide when creating the alert. This URI must be a valid HTTP or HTTPS endpoint. Azure posts one entry per request when an alert is activated.

Configuring webhooks via the portal

You can add or update the webhook URI in the Create/Update Alerts screen in the portal.



You can also configure an alert to post to a webhook URI using the Azure PowerShell Cmdlets, Cross-Platform CLI, or Azure Monitor REST API.

Authenticating the webhook

The webhook can authenticate using either of these methods:

1. **Token-based authorization** - The webhook URI is saved with a token ID, eg.
   ```
   https://mysamplealert/webcallback?tokenid=sometokenid&someparameter=somevalue
   ```
2. **Basic authorization** - The webhook URI is saved with a username and password, eg.

```
https://userid:password@mysamplealert/webcallback?someparamater=somevalue&foo=bar
```

Payload schema

The POST operation contains the following JSON payload and schema for all metric-based alerts.

```
{
"status": "Activated",
"context": {
            "timestamp": "2015-08-14T22:26:41.9975398Z",
            "id": "/subscriptions/s1/resourceGroups/useast/providers/microsoft.insights/alertrules/ruleName1",
            "name": "ruleName1",
            "description": "some description",
            "conditionType": "Metric",
            "condition": {
                        "metricName": "Requests",
                        "metricUnit": "Count",
                        "metricValue": "10",
                        "threshold": "10",
                        "windowSize": "15",
                        "timeAggregation": "Average",
                        "operator": "GreaterThanOrEqual"
                }
            "subscriptionId": "s1",
            "resourceGroupName": "useast",
            "resourceName": "mysite1",
            "resourceType": "microsoft.foo/sites",
            "resourceId": "/subscriptions/s1/resourceGroups/useast/providers/microsoft.foo/sites/mysite1",
            "resourceRegion": "centralus",
            "portalLink":
"https://portal.azure.com/#resource/subscriptions/s1/resourceGroups/useast/providers/microsoft.foo/sites/mysite1
"
},
"properties": {
              "key1": "value1",
              "key2": "value2"
              }
}
```

FIELD	MANDATORY	FIXED SET OF VALUES	NOTES
status	Y	“Activated”, “Resolved”	Status for the alert based off of the conditions you have set.
context	Y		The alert context.
timestamp	Y		The time at which the alert was triggered.
id	Y		Every alert rule has a unique id.
name	Y		The alert name.
description	Y		Description of the alert.

FIELD	MANDATORY	FIXED SET OF VALUES	NOTES
conditionType	Y	“Metric”, “Event”	Two types of alerts are supported. One based on a metric condition and the other based on an event in the Activity Log. Use this value to check if the alert is based on metric or event.
condition	Y		The specific fields to check for based on the conditionType.
metricName	for Metric alerts		The name of the metric that defines what the rule monitors.
metricUnit	for Metric alerts	"Bytes", "BytesPerSecond", "Count", "CountPerSecond", "Percent", "Seconds"	The unit allowed in the metric. Allowed values are listed here.
metricValue	for Metric alerts		The actual value of the metric that caused the alert.
threshold	for Metric alerts		The threshold value at which the alert is activated.
windowSize	for Metric alerts		The period of time that is used to monitor alert activity based on the threshold. Must be between 5 minutes and 1 day. ISO 8601 duration format.
timeAggregation	for Metric alerts	"Average", "Last", "Maximum", "Minimum", "None", "Total"	How the data that is collected should be combined over time. The default value is Average. Allowed values are listed here.
operator	for Metric alerts		The operator used to compare the current metric data to the set threshold.
subscriptionId	Y		Azure subscription ID.
resourceGroupName	Y		Name of the resource group for the impacted resource.
resourceName	Y		Resource name of the impacted resource.
resourceType	Y		Resource type of the impacted resource.

FIELD	MANDATORY	FIXED SET OF VALUES	NOTES
resourceId	Y		Resource ID of the impacted resource.
resourceRegion	Y		Region or location of the impacted resource.
portalLink	Y		Direct link to the portal resource summary page.
properties	N	Optional	Set of `<Key, Value>` pairs (i.e. `Dictionary<String, String>`) that includes details about the event. The properties field is optional. In a custom UI or Logic app-based workflow, users can enter key/values that can be passed via the payload. The alternate way to pass custom properties back to the webhook is via the webhook uri itself (as query parameters)

> **NOTE**
> The properties field can only be set using the Azure Monitor REST API.

Next steps

- Learn more about Azure alerts and webhooks in the video Integrate Azure Alerts with PagerDuty
- Execute Azure Automation scripts (Runbooks) on Azure alerts
- Use Logic App to send an SMS via Twilio from an Azure alert
- Use Logic App to send a Slack message from an Azure alert
- Use Logic App to send a message to an Azure Queue from an Azure alert

Create a metric alert with a Resource Manager template

1/17/2017 • 4 min to read • Edit on GitHub

This article shows how you can use an Azure Resource Manager template to configure Azure metric alerts. This enables you to automatically set up alerts on your resources when they are created to ensure that all resources are monitored correctly.

The basic steps are as follows:

1. Create a template as a JSON file that describes how to create the alert.
2. Deploy the template using any deployment method.

Below we describe how to create a Resource Manager template first for an alert alone, then for an alert during the creation of another resource.

Resource Manager template for a metric alert

To create an alert using a Resource Manager template, you create a resource of type `Microsoft.Insights/alertRules` and fill in all related properties. Below is a template that creates an alert rule.

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "alertName": {
            "type": "string",
            "metadata": {
                "description": "Name of alert"
            }
        },
        "alertDescription": {
            "type": "string",
            "defaultValue": "",
            "metadata": {
                "description": "Description of alert"
            }
        },
        "isEnabled": {
            "type": "bool",
            "defaultValue": true,
            "metadata": {
                "description": "Specifies whether alerts are enabled"
            }
        },
        "resourceId": {
            "type": "string",
            "defaultValue": "",
            "metadata": {
                "description": "Resource ID of the resource emitting the metric that will be used for the
comparison."
            }
        },
        "metricName": {
            "type": "string",
            "defaultValue": "",
            "metadata": {
                "description": "Name of the metric used in the comparison to activate the alert."
```

```
            }
        },
        "operator": {
            "type": "string",
            "defaultValue": "GreaterThan",
            "allowedValues": [
                "GreaterThan",
                "GreaterThanOrEqual",
                "LessThan",
                "LessThanOrEqual"
            ],
            "metadata": {
                "description": "Operator comparing the current value with the threshold value."
            }
        },
        "threshold": {
            "type": "string",
            "defaultValue": "",
            "metadata": {
                "description": "The threshold value at which the alert is activated."
            }
        },
        "aggregation": {
            "type": "string",
            "defaultValue": "Average",
            "allowedValues": [
                "Average",
                "Last",
                "Maximum",
                "Minimum",
                "Total"
            ],
            "metadata": {
                "description": "How the data that is collected should be combined over time."
            }
        },
        "windowSize": {
            "type": "string",
            "defaultValue": "00:05:00",
            "metadata": {
                "description": "Period of time used to monitor alert activity based on the threshold. Must be
between 00:05:00 and 24:00:00. ISO 8601 duration format."
            }
        },
        "sendToServiceOwners": {
            "type": "bool",
            "defaultValue": true,
            "metadata": {
                "description": "Specifies whether alerts are sent to service owners"
            }
        },
        "customEmailAddresses": {
            "type": "string",
            "defaultValue": "",
            "metadata": {
                "description": "Comma-delimited email addresses where the alerts are also sent"
            }
        },
        "webhookUrl": {
            "type": "string",
            "defaultValue": "",
            "metadata": {
                "description": "URL of a webhook that will receive an HTTP POST when the alert activates."
            }
        }
    },
    "variables": {
        "customEmails": "[split(parameters('customEmailAddresses'), ',')]"
    },
```

```
    "resources": [
        {
            "type": "Microsoft.Insights/alertRules",
            "name": "[parameters('alertName')]",
            "location": "[resourceGroup().location]",
            "apiVersion": "2016-03-01",
            "properties": {
                "name": "[parameters('alertName')]",
                "description": "[parameters('alertDescription')]",
                "isEnabled": "[parameters('isEnabled')]",
                "condition": {
                    "odata.type": "Microsoft.Azure.Management.Insights.Models.ThresholdRuleCondition",
                    "dataSource": {
                        "odata.type": "Microsoft.Azure.Management.Insights.Models.RuleMetricDataSource",
                        "resourceUri": "[parameters('resourceId')]",
                        "metricName": "[parameters('metricName')]",
                        "operator": "[parameters('operator')]"
                    },
                    "threshold": "[parameters('threshold')]",
                    "windowSize": "[parameters('windowSize')]",
                    "timeAggregation": "[parameters('aggregation')]"
                },
                "actions": [
                    {
                        "odata.type": "Microsoft.Azure.Management.Insights.Models.RuleEmailAction",
                        "sendToServiceOwners": "[parameters('sendToServiceOwners')]",
                        "customEmails": "[variables('customEmails')]"
                    },
                    {
                        "odata.type": "Microsoft.Azure.Management.Insights.Models.RuleWebhookAction",
                        "serviceUri": "[parameters('webhookUrl')]",
                        "properties": {}
                    }
                ]
            }
        }
    ]
}
```

An explanation of the schema and properties for an alert rule is available here.

Resource Manager template for a resource with an alert

An alert on a Resource Manager template is most often useful when creating an alert while creating a resource. For example, you may want to ensure that a "CPU % > 80" rule is set up every time you deploy a Virtual Machine. To do this, you add the alert rule as a resource in the resource array for your VM template and add a dependency using the `dependsOn` property to the VM resource ID. Here's a full example that creates a Windows VM and adds an alert that notifies subscription admins when the CPU utilization goes above 80%.

```
{
    "$schema": "http://schema.management.azure.com/schemas/2014-04-01-preview/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "newStorageAccountName": {
            "type": "string",
            "metadata": {
                "Description": "The name of the storage account where the VM disk is stored."
            }
        },
        "adminUsername": {
            "type": "string",
            "metadata": {
                "Description": "The name of the administrator account on the VM."
            }
        },
```

```
        "adminPassword": {
            "type": "securestring",
            "metadata": {
                "Description": "The administrator account password on the VM."
            }
        },
        "dnsNameForPublicIP": {
            "type": "string",
            "metadata": {
                "Description": "The name of the public IP address used to access the VM."
            }
        }
    },
    "variables": {
        "location": "Central US",
        "imagePublisher": "MicrosoftWindowsServer",
        "imageOffer": "WindowsServer",
        "windowsOSVersion": "2012-R2-Datacenter",
        "OSDiskName": "osdisk1",
        "nicName": "nc1",
        "addressPrefix": "10.0.0.0/16",
        "subnetName": "sn1",
        "subnetPrefix": "10.0.0.0/24",
        "storageAccountType": "Standard_LRS",
        "publicIPAddressName": "ip1",
        "publicIPAddressType": "Dynamic",
        "vmStorageAccountContainerName": "vhds",
        "vmName": "vm1",
        "vmSize": "Standard_A0",
        "virtualNetworkName": "vn1",
        "vnetID": "[resourceId('Microsoft.Network/virtualNetworks',variables('virtualNetworkName'))]",
        "subnetRef": "[concat(variables('vnetID'),'/subnets/',variables('subnetName'))]",
        "vmID":"[resourceId('Microsoft.Compute/virtualMachines',variables('vmName'))]",
        "alertName": "highCPUOnVM",
        "alertDescription":"CPU is over 80%",
        "alertIsEnabled": true,
        "resourceId": "",
        "metricName": "Percentage CPU",
        "operator": "GreaterThan",
        "threshold": "80",
        "windowSize": "00:10:00",
        "aggregation": "Average",
        "customEmails": "",
        "sendToServiceOwners": true,
        "webhookUrl": "http://testwebhook.test"
    },
    "resources": [
        {
            "type": "Microsoft.Storage/storageAccounts",
            "name": "[parameters('newStorageAccountName')]",
            "apiVersion": "2015-06-15",
            "location": "[variables('location')]",
            "properties": {
                "accountType": "[variables('storageAccountType')]"
            }
        },
        {
            "apiVersion": "2016-03-30",
            "type": "Microsoft.Network/publicIPAddresses",
            "name": "[variables('publicIPAddressName')]",
            "location": "[variables('location')]",
            "properties": {
                "publicIPAllocationMethod": "[variables('publicIPAddressType')]",
                "dnsSettings": {
                    "domainNameLabel": "[parameters('dnsNameForPublicIP')]"
                }
            }
        },
        {
```

```
            "apiVersion": "2016-03-30",
            "type": "Microsoft.Network/virtualNetworks",
            "name": "[variables('virtualNetworkName')]",
            "location": "[variables('location')]",
            "properties": {
                "addressSpace": {
                    "addressPrefixes": [
                        "[variables('addressPrefix')]"
                    ]
                },
                "subnets": [
                    {
                        "name": "[variables('subnetName')]",
                        "properties": {
                            "addressPrefix": "[variables('subnetPrefix')]"
                        }
                    }
                ]
            }
        },
        {
            "apiVersion": "2016-03-30",
            "type": "Microsoft.Network/networkInterfaces",
            "name": "[variables('nicName')]",
            "location": "[variables('location')]",
            "dependsOn": [
                "[concat('Microsoft.Network/publicIPAddresses/', variables('publicIPAddressName'))]",
                "[concat('Microsoft.Network/virtualNetworks/', variables('virtualNetworkName'))]"
            ],
            "properties": {
                "ipConfigurations": [
                    {
                        "name": "ipconfig1",
                        "properties": {
                            "privateIPAllocationMethod": "Dynamic",
                            "publicIPAddress": {
                                "id": "
[resourceId('Microsoft.Network/publicIPAddresses',variables('publicIPAddressName'))]"
                            },
                            "subnet": {
                                "id": "[variables('subnetRef')]"
                            }
                        }
                    }
                ]
            }
        },
        {
            "apiVersion": "2016-03-30",
            "type": "Microsoft.Compute/virtualMachines",
            "name": "[variables('vmName')]",
            "location": "[variables('location')]",
            "dependsOn": [
                "[concat('Microsoft.Storage/storageAccounts/', parameters('newStorageAccountName'))]",
                "[concat('Microsoft.Network/networkInterfaces/', variables('nicName'))]"
            ],
            "properties": {
                "hardwareProfile": {
                    "vmSize": "[variables('vmSize')]"
                },
                "osProfile": {
                    "computername": "[variables('vmName')]",
                    "adminUsername": "[parameters('adminUsername')]",
                    "adminPassword": "[parameters('adminPassword')]"
                },
                "storageProfile": {
                    "imageReference": {
                        "publisher": "[variables('imagePublisher')]",
                        "offer": "[variables('imageOffer')]",
```

```
                        "sku": "[variables('windowsOSVersion')]",
                        "version": "latest"
                    },
                    "osDisk": {
                        "name": "osdisk",
                        "vhd": {
                            "uri": "
[concat('http://',parameters('newStorageAccountName'),'.blob.core.windows.net/',variables('vmStorageAccountContai
nerName'),'/',variables('OSDiskName'),'.vhd')]"
                        },
                        "caching": "ReadWrite",
                        "createOption": "FromImage"
                    }
                },
                "networkProfile": {
                    "networkInterfaces": [
                        {
                            "id": "[resourceId('Microsoft.Network/networkInterfaces',variables('nicName'))]"
                        }
                    ]
                }
            }
        },
        {
            "type": "Microsoft.Insights/alertRules",
            "name": "[variables('alertName')]",
            "dependsOn": [
                "[variables('vmID')]"
            ],
            "location": "[variables('location')]",
            "apiVersion": "2016-03-01",
            "properties": {
                "name": "[variables('alertName')]",
                "description": "variables('alertDescription')",
                "isEnabled": "[variables('alertIsEnabled')]",
                "condition": {
                    "odata.type": "Microsoft.Azure.Management.Insights.Models.ThresholdRuleCondition",
                    "dataSource": {
                        "odata.type": "Microsoft.Azure.Management.Insights.Models.RuleMetricDataSource",
                        "resourceUri": "[variables('vmID')]",
                        "metricName": "[variables('metricName')]",
                        "operator": "[variables('operator')]"
                    },
                    "threshold": "[variables('threshold')]",
                    "windowSize": "[variables('windowSize')]",
                    "timeAggregation": "[variables('aggregation')]"
                },
                "actions": [
                    {
                        "odata.type": "Microsoft.Azure.Management.Insights.Models.RuleEmailAction",
                        "sendToServiceOwners": "[variables('sendToServiceOwners')]",
                        "customEmails": "[variables('customEmails')]"
                    },
                    {
                        "odata.type": "Microsoft.Azure.Management.Insights.Models.RuleWebhookAction",
                        "serviceUri": "[variables('webhookUrl')]",
                        "properties": {}
                    }
                ]
            }
        }
    ]
}
```

Next Steps

- Read more about Alerts
- Add Diagnostic Settings to your Resource Manager template

Best practices for Azure Monitor autoscaling

1/17/2017 • 9 min to read • Edit on GitHub

The following sections in this document help you understand the best practices for autoscale-in Azure. After reviewing this information, you'll be better able to effectively use autoscale in your Azure infrastructure.

Autoscale concepts

- A resource can have only *one* autoscale setting
- An autoscale setting can have one or more profiles and each profile can have one or more autoscale rules.
- An autoscale setting scales instances horizontally, which is *out* by increasing the instances and *in* by decreasing the number of instances. An autoscale setting has a maximum, minimum, and default value of instances.
- An autoscale job always reads the associated metric to scale by, checking if it has crossed the configured threshold for scale-out or scale-in. You can view a list of metrics that autoscale can scale by at Azure Monitor autoscaling common metrics.
- All thresholds are calculated at an instance level. For example, "scale out by 1 instance when average CPU > 80% when instance count is 2", means scale-out when the average CPU across all instances is greater than 80%.
- You always receive failure notifications via email. Specifically, the owner, contributor, and readers of the target resource receive email. You also always receive a *recovery* email when autoscale recovers from a failure and starts functioning normally.
- You can opt-in to receive a successful scale action notification via email and webhooks.

Autoscale best practices

Use the following best practices as you use autoscale.

Ensure the maximum and minimum values are different and have an adequate margin between them

If you have a setting that has minimum=2, maximum=2 and the current instance count is 2, no scale action can occur. Keep an adequate margin between the maximum and minimum instance counts, which are inclusive. Autoscale always scales between these limits.

Manual scaling is reset by autoscale min and max

If you manually update the instance count to a value above or below the maximum, the autoscale engine automatically scales back to the minimum (if below) or the maximum (if above). For example, you set the range between 3 and 6. If you have one running instance, the autoscale engine scales to 3 instances on its next run. Likewise, it would scale-in 8 instances back to 6 on its next run. Manual scaling is very temporary unless you reset the autoscale rules as well.

Always use a scale-out and scale-in rule combination that performs an increase and decrease

If you use only one part of the combination, autoscale scale-in that single out, or in, until the maximum, or minimum, is reached.

Do not switch between the Azure portal and the Azure classic portal when managing Autoscale

For Cloud Services and App Services (Web Apps), use the Azure portal (portal.azure.com) to create and manage autoscale settings. For Virtual Machine Scale Sets use PoSH, CLI or REST API to create and manage autoscale setting. Do not switch between the Azure classic portal (manage.windowsazure.com) and the Azure portal (portal.azure.com) when managing autoscale configurations. The Azure classic portal and its underlying backend has limitations. Move to the Azure portal to manage autoscale using a graphical user interface. The options are to use the autoscale PowerShell, CLI or REST API (via Azure Resource Explorer).

Choose the appropriate statistic for your diagnostics metric

For diagnostics metrics, you can choose among *Average*, *Minimum*, *Maximum* and *Total* as a metric to scale by. The most common statistic is *Average*.

Choose the thresholds carefully for all metric types

We recommend carefully choosing different thresholds for scale-out and scale-in based on practical situations.

We *do not recommend* autoscale settings like the examples below with the same or very similar threshold values for out and in conditions:

- Increase instances by 1 count when Thread Count <= 600
- Decrease instances by 1 count when Thread Count >= 600

Let's look at an example of what can lead to a behavior that may seem confusing. Cosider the following sequence.

1. Assume there are 2 instances to begin with and then the average number of threads per instance grows to 625.
2. Autoscale scales out adding a 3rd instance.
3. Next, assume that the average thread count across instance falls to 575.
4. Before scaling down, autoscale tries to estimate what the final state will be if it scaled in. For example, 575 x 3 (current instance count) = 1,725 / 2 (final number of instances when scaled down) = 862.5 threads. This means autoscale would have to immediately scale-out again even after it scaled in, if the average thread count remains the same or even falls only a small amount. However, if it scaled up again, the whole process would repeat, leading to an infinite loop.
5. To avoid this situation (termed "flapping"), autoscale does not scale down at all. Instead, it skips and reevaluates the condition again the next time the service's job executes. This could confuse many people because autoscale wouldn't appear to work when the average thread count was 575.

Estimation during a scale-in is intended to avoid "flappy" situations. You should keep this behavior in mind when you choose the same thresholds for scale-out and in.

We recommend choosing an adequate margin between the scale-out and in thresholds. As an example, consider the following better rule combination.

- Increase instances by 1 count when CPU% >= 80
- Decrease instances by 1 count when CPU% <= 60

In this case

1. Assume there are 2 instances to start with.
2. If the average CPU% across instances goes to 80, autoscale scales out adding a third instance.
3. Now assume that over time the CPU% falls to 60.
4. Autoscale's scale-in rule estimates the final state if it were to scale-in. For example, 60 x 3 (current instance count) = 180 / 2 (final number of instances when scaled down) = 90. So autoscale does not scale-in because it would have to scale-out again immediately. Instead, it skips scaling down.
5. The next time autoscale checks, the CPU continues to fall to 50. It estimates again - 50 x 3 instance = 150 / 2 instances = 75, which is below the scale-out threshold of 80, so it scales in successfully to 2 instances.

Considerations for scaling threshold values for special metrics

For special metrics such as Storage or Service Bus Queue length metric, the threshold is the average number of messages available per current number of instances. Carefully choose the choose the threshold value for this metric.

Let's illustrate it with an example to ensure you understand the behavior better.

- Increase instances by 1 count when Storage Queue message count >= 50
- Decrease instances by 1 count when Storage Queue message count <= 10

Consider the following sequence:

1. There are 2 storage queue instances.
2. Messages keep coming and when you review the storage queue, the total count reads 50. You might assume that autoscale should start a scale-out action. However, note that it is still 50/2 = 25 messages per instance. So, scale-out does not occur. For the first scale-out to happen, the total message count in the storage queue should be 100.
3. Next, assume that the total message count reaches 100.
4. A 3rd storage queue instance is added due to a scale-out action. The next scale-out action will not happen until the total message count in the queue reaches 150 because 150/3 = 50.
5. Now the number of messages in the queue gets smaller. With 3 instances, the first scale-in action happens when the total messages in all queues add up to 30 because 30/3 = 10 messages per instance, which is the scale-in threshold.

Considerations for scaling when multiple profiles are configured in an autoscale setting

In an autoscale setting, you can choose a default profile, which is always applied without any dependency on schedule or time, or you can choose a recurring profile or a profile for a fixed period with a date and time range.

When autoscale service processes them, it always checks in the following order:

1. Fixed Date profile
2. Recurring profile
3. Default ("Always") profile

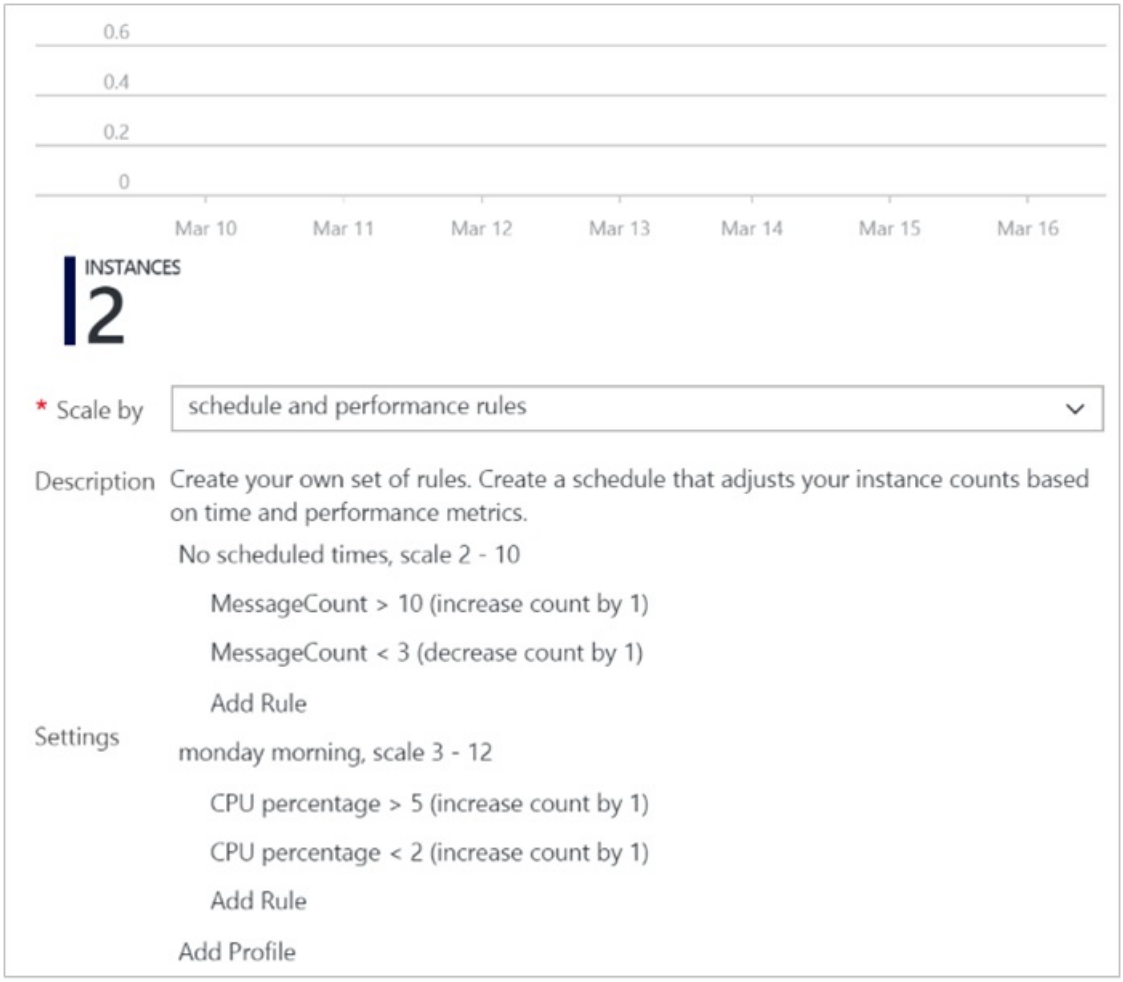
If a profile condition is met, autoscale does not check the next profile condition below it. Autoscale only processes one profile at a time. This means if you want to also include a processing condition from a lower-tier profile, you must include those rules as well in the current profile.

Let's review this using an example:

The image below shows an autoscale setting with a default profile of minimum instances = 2 and maximum instances = 10. In this example, rules are configured to scale-out when the message count in the queue is greater than 10 and scale-in when the message count in the queue is less than 3. So now the resource can scale between 2 and 10 instances.

In addition, there is a recurring profile set for Monday. It is set for minimum instances = 2 and maximum instances = 12. This means on Monday, the first time autoscale checks for this condition, if the instance count is 2, it scales to the new minimum of 3. As long as autoscale continues to find this profile condition matched (Monday), it only processes the CPU-based scale-out/in rules configured for this profile. At this time, it does not check for the queue length. However, if you also want the queue length condition to be checked, you should include those rules from the default profile as well in your Monday profile.

Similarly, when autoscale switches back to the default profile, it first checks if the minimum and maximum conditions are met. If the number of instances at the time is 12, it scales in to 10, the maximum allowed for the default profile.



Considerations for scaling when multiple rules are configured in a profile

There are cases where you may have to set multiple rules in a profile. The following set of autoscale rules are used by services use when multiple rules are set.

On *scale out*, autoscale runs if any rule is met. On *scale-in*, autoscale require all rules to be met.

To illustrate, assume that you have the following 4 autoscale rules:

- If CPU < 30 %, scale-in by 1
- If Memory < 50%, scale-in by 1
- If CPU > 75%, scale-out by 1
- If Memory > 75%, scale-out by 1

Then the follow occurs:

- If CPU is 76% and Memory is 50%, we scale-out.
- If CPU is 50% and Memory is 76% we scale-out.

On the other hand, if CPU is 25% and memory is 51% autoscale does **not** scale-in. In order to scale-in, CPU must be 29% and Memory 49%.

Always select a safe default instance count

The default instance count is important autoscale scales your service to that count when metrics are not available. Therefore, select a default instance count that's safe for your workloads.

Configure autoscale notifications

Autoscale notifies the administrators and contributors of the resource by email if any of the following conditions occur:

- autoscale service fails to take an action.
- Metrics are not available for autoscale service to make a scale decision.
- Metrics are available (recovery) again to make a scale decision. In addition to the conditions above, you can configure email or webhook notifications to get notified for successful scale actions.

Azure Monitor autoscaling common metrics

1/17/2017 • 5 min to read • Edit on GitHub

Azure Monitor autoscaling allows you to scale the number of running instances up or down, based on telemetry data (metrics). This document describes common metrics that you might want to use. In the Azure portal for Cloud Services and Server Farms, you can choose the metric of the resource to scale by. However, you can also choose any metric from a different resource to scale by.

The following information also applies to scaling Virtual Machine Scale Sets.

> **NOTE**
> This information applies to Resource Manager based VMs and VM Scale Sets only.

Compute metrics for Resource Manager-based VMs

By default, Resource Manager-based Virtual Machines and Virtual Machine Scale Sets emit basic (host-level) metrics. In addition, when you configure diagnostics data collection for an Azure VM and VMSS, the Azure diagnostic extension also emits guest-OS performance counters (commonly known as "guest-OS metrics"). You use all these metrics in autoscale rules.

You can use the `Get MetricDefinitions` API/PoSH/CLI to view the metrics available for your VMSS resource.

If you're using VM scale sets and you don't see a particular metric listed, then it is likely *disabled* in your diagnostics extension.

If a particular metric is not being sampled or transferred at the frequency you want, you can update the diagnostics configuration.

If either preceding case is true, then review Use PowerShell to enable Azure Diagnostics in a virtual machine running Windows about PowerShell to configure and update your Azure VM Diagnostics extension to enable the metric. That article also includes a sample diagnostics configuration file.

Host metrics for Resource Manager-based Windows and Linux VMs

The following host-level metrics are emitted by default for Azure VM and VMSS in both Windows and Linux instances. These metrics describe your Azure VM, but are collected from the Azure VM host rather than via agent installed on the guest VM. You may use these metrics in autoscaling rules.

- Host metrics for Resource Manager-based Windows and Linux VMs
- Host metrics for Resource Manager-based Windows and Linux VM Scale Sets

Guest OS metrics Resource Manager-based Windows VMs

When you create a VM in Azure, diagnostics is enabled by using the Diagnostics extension. The diagnostics extension emits a set of metrics taken from inside of the VM. This means you can autoscale off of metrics that are not emitted by default.

You can generate a list of the metrics by using the following command in PowerShell.

```
Get-AzureRmMetricDefinition -ResourceId <resource_id> | Format-Table -Property Name,Unit
```

You can create an alert for the following metrics:

METRIC NAME	UNIT
\Processor(_Total)\% Processor Time	Percent
\Processor(_Total)\% Privileged Time	Percent
\Processor(_Total)\% User Time	Percent
\Processor Information(_Total)\Processor Frequency	Count
\System\Processes	Count
\Process(_Total)\Thread Count	Count
\Process(_Total)\Handle Count	Count
\Memory\% Committed Bytes In Use	Percent
\Memory\Available Bytes	Bytes
\Memory\Committed Bytes	Bytes
\Memory\Commit Limit	Bytes
\Memory\Pool Paged Bytes	Bytes
\Memory\Pool Nonpaged Bytes	Bytes
\PhysicalDisk(_Total)\% Disk Time	Percent
\PhysicalDisk(_Total)\% Disk Read Time	Percent
\PhysicalDisk(_Total)\% Disk Write Time	Percent
\PhysicalDisk(_Total)\Disk Transfers/sec	CountPerSecond
\PhysicalDisk(_Total)\Disk Reads/sec	CountPerSecond
\PhysicalDisk(_Total)\Disk Writes/sec	CountPerSecond
\PhysicalDisk(_Total)\Disk Bytes/sec	BytesPerSecond
\PhysicalDisk(_Total)\Disk Read Bytes/sec	BytesPerSecond
\PhysicalDisk(_Total)\Disk Write Bytes/sec	BytesPerSecond
\PhysicalDisk(_Total)\Avg. Disk Queue Length	Count
\PhysicalDisk(_Total)\Avg. Disk Read Queue Length	Count
\PhysicalDisk(_Total)\Avg. Disk Write Queue Length	Count

METRIC NAME	UNIT
\LogicalDisk(_Total)\% Free Space	Percent
\LogicalDisk(_Total)\Free Megabytes	Count

Guest OS metrics Linux VMs

When you create a VM in Azure, diagnostics is enabled by default by using Diagnostics extension.

You can generate a list of the metrics by using the following command in PowerShell.

```
Get-AzureRmMetricDefinition -ResourceId <resource_id> | Format-Table -Property Name,Unit
```

You can create an alert for the following metrics:

METRIC NAME	UNIT
\Memory\AvailableMemory	Bytes
\Memory\PercentAvailableMemory	Percent
\Memory\UsedMemory	Bytes
\Memory\PercentUsedMemory	Percent
\Memory\PercentUsedByCache	Percent
\Memory\PagesPerSec	CountPerSecond
\Memory\PagesReadPerSec	CountPerSecond
\Memory\PagesWrittenPerSec	CountPerSecond
\Memory\AvailableSwap	Bytes
\Memory\PercentAvailableSwap	Percent
\Memory\UsedSwap	Bytes
\Memory\PercentUsedSwap	Percent
\Processor\PercentIdleTime	Percent
\Processor\PercentUserTime	Percent
\Processor\PercentNiceTime	Percent
\Processor\PercentPrivilegedTime	Percent
\Processor\PercentInterruptTime	Percent
\Processor\PercentDPCTime	Percent

METRIC NAME	UNIT
\Processor\PercentProcessorTime	Percent
\Processor\PercentIOWaitTime	Percent
\PhysicalDisk\BytesPerSecond	BytesPerSecond
\PhysicalDisk\ReadBytesPerSecond	BytesPerSecond
\PhysicalDisk\WriteBytesPerSecond	BytesPerSecond
\PhysicalDisk\TransfersPerSecond	CountPerSecond
\PhysicalDisk\ReadsPerSecond	CountPerSecond
\PhysicalDisk\WritesPerSecond	CountPerSecond
\PhysicalDisk\AverageReadTime	Seconds
\PhysicalDisk\AverageWriteTime	Seconds
\PhysicalDisk\AverageTransferTime	Seconds
\PhysicalDisk\AverageDiskQueueLength	Count
\NetworkInterface\BytesTransmitted	Bytes
\NetworkInterface\BytesReceived	Bytes
\NetworkInterface\PacketsTransmitted	Count
\NetworkInterface\PacketsReceived	Count
\NetworkInterface\BytesTotal	Bytes
\NetworkInterface\TotalRxErrors	Count
\NetworkInterface\TotalTxErrors	Count
\NetworkInterface\TotalCollisions	Count

Commonly used Web (Server Farm) metrics

You can also perform autoscale based on common web server metrics such as the Http queue length. It's metric name is **HttpQueueLength**. The following section lists available server farm (Web Apps) metrics.

Web Apps metrics

You can generate a list of the Web Apps metrics by using the following command in PowerShell.

```
Get-AzureRmMetricDefinition -ResourceId <resource_id> | Format-Table -Property Name,Unit
```

You can alert on or scale by these metrics.

METRIC NAME	UNIT
CpuPercentage	Percent
MemoryPercentage	Percent
DiskQueueLength	Count
HttpQueueLength	Count
BytesReceived	Bytes
BytesSent	Bytes

Commonly used Storage metrics

You can scale by Storage queue length, which is the number of messages in the storage queue. Storage queue length is a special metric and the threshold is the number of messages per instance. For example, if there are two instances and if the threshold is set to 100, scaling occurs when the total number of messages in the queue is 200. That can be 100 messages per instance, 120 and 80, or any other combination that adds up to 200 or more.

Configure this setting in the Azure portal in the **Settings** blade. For VM scale sets, you can update the Autoscale setting in the Resource Manager template to use *metricName* as *ApproximateMessageCount* and pass the ID of the storage queue as *metricResourceUri*.

For example, with a Classic Storage Account the autoscale setting metricTrigger would include:

```
"metricName": "ApproximateMessageCount",
 "metricNamespace": "",
 "metricResourceUri":
"/subscriptions/s1/resourceGroups/rg1/providers/Microsoft.ClassicStorage/storageAccounts/mystorage/services/queue/queues/mystoragequeue"
```

For a (non-classic) storage account, the metricTrigger would include:

```
"metricName": "ApproximateMessageCount",
"metricNamespace": "",
"metricResourceUri":
"/subscriptions/s1/resourceGroups/rg1/providers/Microsoft.Storage/storageAccounts/mystorage/services/queue/queues/mystoragequeue"
```

Commonly used Service Bus metrics

You can scale by Service Bus queue length, which is the number of messages in the Service Bus queue. Service Bus queue length is a special metric and the threshold is the number of messages per instance. For example, if there are two instances and if the threshold is set to 100, scaling occurs when the total number of messages in the queue is 200. That can be 100 messages per instance, 120 and 80, or any other combination that adds up to 200 or more.

For VM scale sets, you can update the Autoscale setting in the Resource Manager template to use *metricName* as *ApproximateMessageCount* and pass the ID of the storage queue as *metricResourceUri*.

```
"metricName": "MessageCount",
 "metricNamespace": "",
"metricResourceUri":
"/subscriptions/s1/resourceGroups/rg1/providers/Microsoft.ServiceBus/namespaces/mySB/queues/myqueue"
```

NOTE

For Service Bus, the resource group concept does not exist but Azure Resource Manager creates a default resource group per region. The resource group is usually in the 'Default-ServiceBus-[region]' format. For example, 'Default-ServiceBus-EastUS', 'Default-ServiceBus-WestUS', 'Default-ServiceBus-AustraliaEast' etc.

Advanced Autoscale configuration using Resource Manager templates for VM Scale Sets

1/17/2017 • 5 min to read • Edit on GitHub

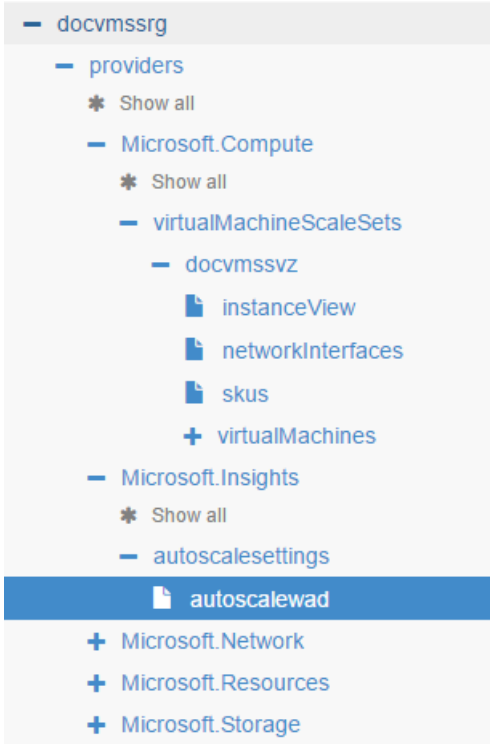
You can scale out and in Virtual Machine Scale Sets based on performance metric thresholds, by a recurring schedule, or by a particular date. You can also configure email and webhook notifications for scale actions. This walkthrough shows an example of configuring all these objects using a Resource Manager template on a VM Scale Set.

> **NOTE**
>
> While this walkthrough explains the steps for VM Scale Sets, the same information applies to autoscaling Cloud Services and Web Apps. For a simple scale in/out setting on a VM Scale Set based on a simple performance metric such as CPU, refer to the Linux and Windows documents

Walkthrough

In this walkthrough, we use Azure Resource Explorer to configure and update the autoscale setting for a scale set. Azure Resource Explorer is an easy way to manage Azure resources via Resource Manager templates. If you are new to Azure Resource Explorer tool, read this introduction.

1. Deploy a new scale set with a basic autoscale setting. This article uses the one from the Azure QuickStart Gallery, which has a Windows scale set with a basic autoscale template. Linux scale sets work the same way.
2. After the scale set is created, navigate to the scale set resource from Azure Resource Explorer. You see the following under Microsoft.Insights node.



The template execution has created a default autoscale setting with the name **'autoscalewad'**. On the right-hand side, you can view the full definition of this autoscale setting. In this case, the default autoscale setting comes with a CPU% based scale-out and scale-in rule.

3. You can now add more profiles and rules based on the schedule or specific requirements. We create an autoscale setting with three profiles. To understand profiles and rules in autoscale, review Autoscale Best Practices.

PROFILES & RULES	DESCRIPTION
Profile	**Performance/metric based**
Rule	Service Bus Queue Message Count > x
Rule	Service Bus Queue Message Count < y
Rule	CPU% > n
Rule	CPU% < p
Profile	**Weekday morning hours (no rules)**
Profile	**Product Launch day (no rules)**

4. Here is a hypothetical scaling scenario that we use for this walk-through.

   - **Load based** - I'd like to scale out or in based on the load on my application hosted on my scale set.*
   - **Message Queue size** - I use a Service Bus Queue for the incoming messages to my application. I use the queue's message count and CPU% and configure a default profile to trigger a scale action if either of message count or CPU hits the threshold.*
   - **Time of week and day** - I want a weekly recurring 'time of the day' based profile called 'Weekday Morning Hours'. Based on historical data, I know it is better to have certain number of VM instances to handle my application's load during this time.*
   - **Special Dates** - I added a 'Product Launch Day' profile. I plan ahead for specific dates so my application is ready to handle the load due marketing announcements and when we put a new product in the application.*
   - *The last two profiles can also have other performance metric based rules within them. In this case, I decided not to have one and instead to rely on the default performance metric based rules. Rules are optional for the recurring and date-based profiles.*

     Autoscale engine's prioritization of the profiles and rules is also captured in the autoscaling best practices article. For a list of common metrics for autoscale, refer Common metrics for Autoscale

5. Make sure you are on the **Read/Write** mode in Resource Explorer

6. Click Edit. **Replace** the 'profiles' element in autoscale setting with the following configuration:

```
  "profiles": [
    {
```

```
 {
        "name": "Perf_Based_Scale",
        "capacity": {
          "minimum": "2",
          "maximum": "12",
          "default": "2"
        },
        "rules": [
          {
            "metricTrigger": {
              "metricName": "MessageCount",
              "metricNamespace": "",
              "metricResourceUri":
"/subscriptions/s1/resourceGroups/rg1/providers/Microsoft.ServiceBus/namespaces/mySB/queues/myqueue",
              "timeGrain": "PT5M",
              "statistic": "Average",
              "timeWindow": "PT5M",
              "timeAggregation": "Average",
              "operator": "GreaterThan",
              "threshold": 10
            },
            "scaleAction": {
              "direction": "Increase",
              "type": "ChangeCount",
              "value": "1",
              "cooldown": "PT5M"
            }
          },
          {
            "metricTrigger": {
              "metricName": "MessageCount",
              "metricNamespace": "",
              "metricResourceUri":
"/subscriptions/s1/resourceGroups/rg1/providers/Microsoft.ServiceBus/namespaces/mySB/queues/myqueue",
              "timeGrain": "PT5M",
              "statistic": "Average",
              "timeWindow": "PT5M",
              "timeAggregation": "Average",
              "operator": "LessThan",
              "threshold": 3
            },
            "scaleAction": {
              "direction": "Decrease",
              "type": "ChangeCount",
              "value": "1",
              "cooldown": "PT5M"
            }
          },
          {
            "metricTrigger": {
              "metricName": "Percentage CPU",
              "metricNamespace": "",
              "metricResourceUri":
"/subscriptions/s1/resourceGroups/rg1/providers/Microsoft.Compute/virtualMachineScaleSets/<this_vmss_name
>",
              "timeGrain": "PT5M",
              "statistic": "Average",
              "timeWindow": "PT30M",
              "timeAggregation": "Average",
              "operator": "GreaterThan",
              "threshold": 85
            },
            "scaleAction": {
              "direction": "Increase",
              "type": "ChangeCount",
```

```
              "value": "1",
              "cooldown": "PT5M"
            }
          },
          {
            "metricTrigger": {
              "metricName": "Percentage CPU",
              "metricNamespace": "",
              "metricResourceUri":
"/subscriptions/s1/resourceGroups/rg1/providers/Microsoft.Compute/virtualMachineScaleSets/<this_vmss_name
>",
              "timeGrain": "PT5M",
              "statistic": "Average",
              "timeWindow": "PT30M",
              "timeAggregation": "Average",
              "operator": "LessThan",
              "threshold": 60
            },
            "scaleAction": {
              "direction": "Decrease",
              "type": "ChangeCount",
              "value": "1",
              "cooldown": "PT5M"
            }
          }
        ]
      },
      {
        "name": "Weekday_Morning_Hours_Scale",
        "capacity": {
          "minimum": "4",
          "maximum": "12",
          "default": "4"
        },
        "rules": [],
        "recurrence": {
          "frequency": "Week",
          "schedule": {
            "timeZone": "Pacific Standard Time",
            "days": [
              "Monday",
              "Tuesday",
              "Wednesday",
              "Thursday",
              "Friday"
            ],
            "hours": [
              6
            ],
            "minutes": [
              0
            ]
          }
        }
      },
      {
        "name": "Product_Launch_Day",
        "capacity": {
          "minimum": "6",
          "maximum": "20",
          "default": "6"
        },
        "rules": [],
        "fixedDate": {
          "timeZone": "Pacific Standard Time",
          "start": "2016-06-20T00:06:00Z",
          "end": "2016-06-21T23:59:00Z"
        }
      }
```

For supported fields and their values, see Autoscale REST API documentation. Now your autoscale setting contains the three profiles explained previously.

7. Finally, look at the Autoscale **notification** section. Autoscale notifications allow you to do three things when a scale-out or in action is successfully triggered.

   - Notify the admin and co-admins of your subscription
   - Email a set of users
   - Trigger a webhook call. When fired, this webhook sends metadata about the autoscaling condition and the scale set resource. To learn more about the payload of autoscale webhook, see Configure Webhook & Email Notifications for Autoscale.

   Add the following to the Autoscale setting replacing your **notification** element whose value is null

```
"notifications": [
   {
     "operation": "Scale",
     "email": {
       "sendToSubscriptionAdministrator": true,
       "sendToSubscriptionCoAdministrators": false,
       "customEmails": [
           "user1@mycompany.com",
           "user2@mycompany.com"
           ]
     },
     "webhooks": [
       {
         "serviceUri": "https://foo.webhook.example.com?token=abcd1234",
         "properties": {
           "optional_key1": "optional_value1",
           "optional_key2": "optional_value2"
         }
       }
     ]
   }
 ]
```

   Hit **Put** button in Resource Explorer to update the autoscale setting.

You have updated an autoscale setting on a VM Scale set to include multiple scale profiles and scale notifications.

Next Steps

Use these links to learn more about autoscaling.

Common Metrics for Autoscale

Best Practices for Azure Autoscale

Manage Autoscale using PowerShell

Manage Autoscale using CLI

Configure Webhook & Email Notifications for Autoscale

Automatically scale machines in a virtual machine scale set

1/17/2017 • 12 min to read • Edit on GitHub

Virtual machine scale sets make it easy for you to deploy and manage identical virtual machines as a set. Scale sets provide a highly scalable and customizable compute layer for hyperscale applications, and they support Windows platform images, Linux platform images, custom images, and extensions. For more information about scale sets, see Virtual Machine Scale Sets.

This tutorial shows you how to create a scale set of Windows virtual machines and automatically scale the machines in the set. You create the scale set and set up scaling by creating an Azure Resource Manager template and deploying it using Azure PowerShell. For more information about templates, see Authoring Azure Resource Manager templates. To learn more about automatic scaling of scale sets, see Automatic scaling and Virtual Machine Scale Sets.

In this article, you deploy the following resources and extensions:

- Microsoft.Storage/storageAccounts
- Microsoft.Network/virtualNetworks
- Microsoft.Network/publicIPAddresses
- Microsoft.Network/loadBalancers
- Microsoft.Network/networkInterfaces
- Microsoft.Compute/virtualMachines
- Microsoft.Compute/virtualMachineScaleSets
- Microsoft.Insights.VMDiagnosticsSettings
- Microsoft.Insights/autoscaleSettings

For more information about Resource Manager resources, see Azure Resource Manager vs. classic deployment.

Step 1: Install Azure PowerShell

See How to install and configure Azure PowerShell for information about installing the latest version of Azure PowerShell, selecting your subscription, and signing in to Azure.

Step 2: Create a resource group and a storage account

1. **Create a resource group** – All resources must be deployed to a resource group. Use New-AzureRmResourceGroup to create a resource group named **vmsstestrg1**.
2. **Create a storage account** – This storage account is where the template is stored. Use New-AzureRmStorageAccount to create a storage account named **vmsstestsa**.

Step 3: Create the template

An Azure Resource Manager template makes it possible for you to deploy and manage Azure resources together by using a JSON description of the resources and associated deployment parameters.

1. In your favorite editor, create the file C:\VMSSTemplate.json and add the initial JSON structure to support the template.

```
{
  "$schema":"http://schema.management.azure.com/schemas/2014-04-01-preview/VM.json",
  "contentVersion": "1.0.0.0",
  "parameters": {
  },
  "variables": {
  },
  "resources": [
  ]
}
```

2. Parameters are not always required, but they provide a way to input values when the template is deployed. Add these parameters under the parameters parent element that you added to the template.

```
"vmName": { "type": "string" },
"vmSSName": { "type": "string" },
"instanceCount": { "type": "string" },
"adminUsername": { "type": "string" },
"adminPassword": { "type": "securestring" },
"resourcePrefix": { "type": "string" }
```

- A name for the separate virtual machine that is used to access the machines in the scale set.
- The name of the storage account where the template is stored.
- The number of virtual machines to initially create in the scale set.
- The name and password of the administrator account on the virtual machines.
- A name prefix for the resources that are created to support the scale set.

3. Variables can be used in a template to specify values that may change frequently or values that need to be created from a combination of parameter values. Add these variables under the variables parent element that you added to the template.

```
"dnsName1": "[concat(parameters('resourcePrefix'),'dn1')]",
"dnsName2": "[concat(parameters('resourcePrefix'),'dn2')]",
"publicIP1": "[concat(parameters('resourcePrefix'),'ip1')]",
"publicIP2": "[concat(parameters('resourcePrefix'),'ip2')]",
"loadBalancerName": "[concat(parameters('resourcePrefix'),'lb1')]",
"virtualNetworkName": "[concat(parameters('resourcePrefix'),'vn1')]",
"nicName": "[concat(parameters('resourcePrefix'),'nc1')]",
"lbID": "[resourceId('Microsoft.Network/loadBalancers',variables('loadBalancerName'))]",
"frontEndIPConfigID": "[concat(variables('lbID'),'/frontendIPConfigurations/loadBalancerFrontEnd')]",
"storageAccountSuffix": [ "a", "g", "m", "s", "y" ],
"diagnosticsStorageAccountName": "[concat(parameters('resourcePrefix'), 'a')]",
"accountid": "[concat('/subscriptions/',subscription().subscriptionId,'/resourceGroups/',
resourceGroup().name,'/providers/','Microsoft.Storage/storageAccounts/',
variables('diagnosticsStorageAccountName'))]",
   "wadlogs": "<WadCfg> <DiagnosticMonitorConfiguration overallQuotaInMB=\"4096\"
xmlns=\"http://schemas.microsoft.com/ServiceHosting/2010/10/DiagnosticsConfiguration\">
<DiagnosticInfrastructureLogs scheduledTransferLogLevelFilter=\"Error\"/> <WindowsEventLog
scheduledTransferPeriod=\"PT1M\" > <DataSource name=\"Application!*[System[(Level = 1 or Level = 2)]]\"
/> <DataSource name=\"Security!*[System[(Level = 1 or Level = 2)]]\" /> <DataSource name=\"System!*
[System[(Level = 1 or Level = 2)]]\" /></WindowsEventLog>",
"wadperfcounter": "<PerformanceCounters scheduledTransferPeriod=\"PT1M\">
<PerformanceCounterConfiguration counterSpecifier=\"\\Processor(_Total)\\% Processor Time\"
sampleRate=\"PT15S\" unit=\"Percent\"><annotation displayName=\"CPU utilization\" locale=\"en-us\"/>
</PerformanceCounterConfiguration></PerformanceCounters>",
"wadcfgxstart": "[concat(variables('wadlogs'),variables('wadperfcounter'),'<Metrics resourceId=\"')]",
"wadmetricsresourceid": "
[concat('/subscriptions/',subscription().subscriptionId,'/resourceGroups/',resourceGroup().name
,'/providers/','Microsoft.Compute/virtualMachineScaleSets/',parameters('vmssName'))]",
"wadcfgxend": "[concat('\"><MetricAggregation scheduledTransferPeriod=\"PT1H\"/><MetricAggregation
scheduledTransferPeriod=\"PT1M\"/></Metrics></DiagnosticMonitorConfiguration></WadCfg>')]"
```

- DNS names that are used by the network interfaces.
  - The IP address names and prefixes for the virtual network and subnets.
  - The names and identifiers of the virtual network, load balancer, and network interfaces.
  - Storage account names for the accounts associated with the machines in the scale set.
  - Settings for the Diagnostics extension that is installed on the virtual machines. For more information about the Diagnostics extension, see Create a Windows Virtual machine with monitoring and diagnostics using Azure Resource Manager Template.

4. Add the storage account resource under the resources parent element that you added to the template. This template uses a loop to create the recommended five storage accounts where the operating system disks and diagnostic data are stored. This set of accounts can support up to 100 virtual machines in a scale set, which is the current maximum. Each storage account is named with a letter designator that was defined in the variables combined with the prefix that you provide in the parameters for the template.

```
{
  "type": "Microsoft.Storage/storageAccounts",
  "name": "[concat(parameters('resourcePrefix'), variables('storageAccountSuffix')[copyIndex()])]",
  "apiVersion": "2015-06-15",
  "copy": {
    "name": "storageLoop",
    "count": 5
  },
  "location": "[resourceGroup().location]",
  "properties": { "accountType": "Standard_LRS" }
},
```

5. Add the virtual network resource. For more information, see Network Resource Provider.

```
{
  "apiVersion": "2015-06-15",
  "type": "Microsoft.Network/virtualNetworks",
  "name": "[variables('virtualNetworkName')]",
  "location": "[resourceGroup().location]",
  "properties": {
    "addressSpace": { "addressPrefixes": [ "10.0.0.0/16" ] },
    "subnets": [
      {
        "name": "subnet1",
        "properties": { "addressPrefix": "10.0.0.0/24" }
      }
    ]
  }
},
```

6. Add the public IP address resources that are used by the load balancer and network interface.

```
{
  "apiVersion": "2016-03-30",
  "type": "Microsoft.Network/publicIPAddresses",
  "name": "[variables('publicIP1')]",
  "location": "[resourceGroup().location]",
  "properties": {
    "publicIPAllocationMethod": "Dynamic",
    "dnsSettings": {
      "domainNameLabel": "[variables('dnsName1')]"
    }
  }
},
{
  "apiVersion": "2016-03-30",
  "type": "Microsoft.Network/publicIPAddresses",
  "name": "[variables('publicIP2')]",
  "location": "[resourceGroup().location]",
  "properties": {
    "publicIPAllocationMethod": "Dynamic",
    "dnsSettings": {
      "domainNameLabel": "[variables('dnsName2')]"
    }
  }
},
```

7. Add the load balancer resource that is used by the scale set. For more information, see Azure Resource Manager Support for Load Balancer.

```
{
  "apiVersion": "2015-06-15",
  "name": "[variables('loadBalancerName')]",
  "type": "Microsoft.Network/loadBalancers",
  "location": "[resourceGroup().location]",
  "dependsOn": [
    "[concat('Microsoft.Network/publicIPAddresses/', variables('publicIP1'))]"
  ],
  "properties": {
    "frontendIPConfigurations": [
      {
        "name": "loadBalancerFrontEnd",
        "properties": {
          "publicIPAddress": {
            "id": "[concat('Microsoft.Network/publicIPAddresses/', variables('publicIP1'))]"
          }
        }
      }
    ],
    "backendAddressPools": [ { "name": "bepool1" } ],
    "inboundNatPools": [
      {
        "name": "natpool1",
        "properties": {
          "frontendIPConfiguration": {
            "id": "[variables('frontEndIPConfigID')]"
          },
          "protocol": "tcp",
          "frontendPortRangeStart": 50000,
          "frontendPortRangeEnd": 50500,
          "backendPort": 3389
        }
      }
    ]
  }
},
```

8. Add the network interface resource that is used by the separate virtual machine. Because machines in a scale set aren't accessible through a public IP address, a separate virtual machine is created in the same virtual network to remotely access the machines.

```
 {
   "apiVersion": "2016-03-30",
   "type": "Microsoft.Network/networkInterfaces",
   "name": "[variables('nicName')]",
   "location": "[resourceGroup().location]",
   "dependsOn": [
     "[concat('Microsoft.Network/publicIPAddresses/', variables('publicIP2'))]",
     "[concat('Microsoft.Network/virtualNetworks/', variables('virtualNetworkName'))]"
   ],
   "properties": {
     "ipConfigurations": [
       {
         "name": "ipconfig1",
         "properties": {
           "privateIPAllocationMethod": "Dynamic",
           "publicIPAddress": {
             "id": "[resourceId('Microsoft.Network/publicIPAddresses', variables('publicIP2'))]"
           },
           "subnet": {
             "id": "
[concat('/subscriptions/',subscription().subscriptionId,'/resourceGroups/',resourceGroup().name,'/provide
rs/Microsoft.Network/virtualNetworks/',variables('virtualNetworkName'),'/subnets/subnet1')]"
           }
         }
       }
     ]
   }
 },
```

9. Add the separate virtual machine in the same network as the scale set.

```
{
  "apiVersion": "2016-03-30",
  "type": "Microsoft.Compute/virtualMachines",
  "name": "[parameters('vmName')]",
  "location": "[resourceGroup().location]",
  "dependsOn": [
    "storageLoop",
    "[concat('Microsoft.Network/networkInterfaces/', variables('nicName'))]"
  ],
  "properties": {
    "hardwareProfile": { "vmSize": "Standard_A1" },
    "osProfile": {
      "computername": "[parameters('vmName')]",
      "adminUsername": "[parameters('adminUsername')]",
      "adminPassword": "[parameters('adminPassword')]"
    },
    "storageProfile": {
      "imageReference": {
        "publisher": "MicrosoftWindowsServer",
        "offer": "WindowsServer",
        "sku": "2012-R2-Datacenter",
        "version": "latest"
      },
      "osDisk": {
        "name": "[concat(parameters('resourcePrefix'), 'os1')]",
        "vhd": {
          "uri":  "
[concat('https://',parameters('resourcePrefix'),'a.blob.core.windows.net/vhds/',parameters('resourcePrefi
x'),'os1.vhd')]"
        },
        "caching": "ReadWrite",
        "createOption": "FromImage"
      }
    },
    "networkProfile": {
      "networkInterfaces": [
        {
          "id": "[resourceId('Microsoft.Network/networkInterfaces',variables('nicName'))]"
        }
      ]
    }
  }
},
```

10. Add the virtual machine scale set resource and specify the diagnostics extension that is installed on all virtual machines in the scale set. Many of the settings for this resource are similar with the virtual machine resource. The main differences are the capacity element that specifies the number of virtual machines in the scale set, and upgradePolicy that specifies how updates are made to virtual machines. The scale set is not created until all the storage accounts are created as specified with the dependsOn element.

```
{
  "type": "Microsoft.Compute/virtualMachineScaleSets",
  "apiVersion": "2016-03-30",
  "name": "[parameters('vmSSName')]",
  "location": "[resourceGroup().location]",
  "dependsOn": [
    "storageLoop",
    "[concat('Microsoft.Network/virtualNetworks/', variables('virtualNetworkName'))]",
    "[concat('Microsoft.Network/loadBalancers/', variables('loadBalancerName'))]"
  ],
  "sku": {
    "name": "Standard_A1",
    "tier": "Standard",
    "capacity": "[parameters('instanceCount')]"
  },
  "properties": {
```

```
    properties": {
      "upgradePolicy": {
        "mode": "Manual"
      },
      "virtualMachineProfile": {
        "storageProfile": {
          "osDisk": {
            "vhdContainers": [
              "[concat('https://', parameters('resourcePrefix'), variables('storageAccountSuffix')[0],
'.blob.core.windows.net/vhds')]",
              "[concat('https://', parameters('resourcePrefix'), variables('storageAccountSuffix')[1],
'.blob.core.windows.net/vhds')]",
              "[concat('https://', parameters('resourcePrefix'), variables('storageAccountSuffix')[2],
'.blob.core.windows.net/vhds')]",
              "[concat('https://', parameters('resourcePrefix'), variables('storageAccountSuffix')[3],
'.blob.core.windows.net/vhds')]",
              "[concat('https://', parameters('resourcePrefix'), variables('storageAccountSuffix')[4],
'.blob.core.windows.net/vhds')]"
            ],
            "name": "vmssosdisk",
            "caching": "ReadOnly",
            "createOption": "FromImage"
          },
          "imageReference": {
            "publisher": "MicrosoftWindowsServer",
            "offer": "WindowsServer",
            "sku": "2012-R2-Datacenter",
            "version": "latest"
          }
        },
        "osProfile": {
          "computerNamePrefix": "[parameters('vmSSName')]",
          "adminUsername": "[parameters('adminUsername')]",
          "adminPassword": "[parameters('adminPassword')]"
        },
        "networkProfile": {
          "networkInterfaceConfigurations": [
            {
              "name": "networkconfig1",
              "properties": {
                "primary": "true",
                "ipConfigurations": [
                  {
                    "name": "ip1",
                    "properties": {
                      "subnet": {
                        "id": "
[concat('/subscriptions/',subscription().subscriptionId,'/resourceGroups/',resourceGroup().name,'/provide
rs/Microsoft.Network/virtualNetworks/',variables('virtualNetworkName'),'/subnets/subnet1')]"
                      },
                      "loadBalancerBackendAddressPools": [
                        {
                          "id": "
[concat('/subscriptions/',subscription().subscriptionId,'/resourceGroups/',resourceGroup().name,'/provide
rs/Microsoft.Network/loadBalancers/',variables('loadBalancerName'),'/backendAddressPools/bepool1')]"
                        }
                      ],
                      "loadBalancerInboundNatPools": [
                        {
                          "id": "
[concat('/subscriptions/',subscription().subscriptionId,'/resourceGroups/',resourceGroup().name,'/provide
rs/Microsoft.Network/loadBalancers/',variables('loadBalancerName'),'/inboundNatPools/natpool1')]"
                        }
                      ]
                    }
                  }
                ]
              }
            }
```

```
          ]
        },
        "extensionProfile": {
          "extensions": [
            {
              "name": "Microsoft.Insights.VMDiagnosticsSettings",
              "properties": {
                "publisher": "Microsoft.Azure.Diagnostics",
                "type": "IaaSDiagnostics",
                "typeHandlerVersion": "1.5",
                "autoUpgradeMinorVersion": true,
                "settings": {
                  "xmlCfg": "
[base64(concat(variables('wadcfgxstart'),variables('wadmetricsresourceid'),variables('wadcfgxend')))]",
                  "storageAccount": "[variables('diagnosticsStorageAccountName')]"
                },
                "protectedSettings": {
                  "storageAccountName": "[variables('diagnosticsStorageAccountName')]",
                  "storageAccountKey": "[listkeys(variables('accountid'), '2015-06-15').key1]",
                  "storageAccountEndPoint": "https://core.windows.net"
                }
              }
            }
          ]
        }
      }
    }
  },
```

11. Add the autoscaleSettings resource that defines how the scale set adjusts based on the processor usage on the machines in the scale set.

```
{
  "type": "Microsoft.Insights/autoscaleSettings",
  "apiVersion": "2015-04-01",
  "name": "[concat(parameters('resourcePrefix'),'as1')]",
  "location": "[resourceGroup().location]",
  "dependsOn": [
    "[concat('Microsoft.Compute/virtualMachineScaleSets/',parameters('vmSSName'))]"
  ],
  "properties": {
    "enabled": true,
    "name": "[concat(parameters('resourcePrefix'),'as1')]",
    "profiles": [
      {
        "name": "Profile1",
        "capacity": {
          "minimum": "1",
          "maximum": "10",
          "default": "1"
        },
        "rules": [
          {
            "metricTrigger": {
              "metricName": "\\Processor(_Total)\\% Processor Time",
              "metricNamespace": "",
              "metricResourceUri": "
[concat('/subscriptions/',subscription().subscriptionId,'/resourceGroups/',resourceGroup().name,'/provide
rs/Microsoft.Compute/virtualMachineScaleSets/',parameters('vmSSName'))]",
              "timeGrain": "PT1M",
              "statistic": "Average",
              "timeWindow": "PT5M",
              "timeAggregation": "Average",
              "operator": "GreaterThan",
              "threshold": 50.0
            },
            "scaleAction": {
              "direction": "Increase",
              "type": "ChangeCount",
              "value": "1",
              "cooldown": "PT5M"
            }
          }
        ]
      }
    ],
    "targetResourceUri": "[concat('/subscriptions/',subscription().subscriptionId,'/resourceGroups/',
resourceGroup().name,'/providers/Microsoft.Compute/virtualMachineScaleSets/',parameters('vmSSName'))]"
  }
}
```

For this tutorial, these values are important:

- **metricName** - This value is the same as the performance counter that we defined in the wadperfcounter variable. Using that variable, the Diagnostics extension collects the **Processor(_Total)\% Processor Time** counter.
- **metricResourceUri** - This value is the resource identifier of the virtual machine scale set.
- **timeGrain** – This value is the granularity of the metrics that are collected. In this template, it is set to one minute.
- **statistic** – This value determines how the metrics are combined to accommodate the automatic scaling action. The possible values are: Average, Min, Max. In this template, the average total CPU usage of the virtual machines is collected.
- **timeWindow** – This value is the range of time in which instance data is collected. It must be between 5 minutes and 12 hours.
- **timeAggregation** –This value determines how the data that is collected should be combined over time.

The default value is Average. The possible values are: Average, Minimum, Maximum, Last, Total, Count.

- **operator** – This value is the operator that is used to compare the metric data and the threshold. The possible values are: Equals, NotEquals, GreaterThan, GreaterThanOrEqual, LessThan, LessThanOrEqual.
- **threshold** – This value is the value that triggers the scale action. In this template, machines are added to the scale set when the average CPU usage among machines in the set is over 50%.
- **direction** – This value determines the action that is taken when the threshold value is achieved. The possible values are Increase or Decrease. In this template, the number of virtual machines in the scale set is increased if the threshold is over 50% in the defined time window.
- **type** – This value is the type of action that should occur and must be set to ChangeCount.
- **value** – This value is the number of virtual machines that are added or removed from the scale set. This value must be 1 or greater. The default value is 1. In this template, the number of machines in the scale set increases by 1 when the threshold is met.
- **cooldown** – This value is the amount of time to wait since the last scaling action before the next action occurs. This value must be between one minute and one week.

12. Save the template file.

Step 4: Upload the template to storage

The template can be uploaded as long as you know the name and primary key of the storage account that you created in step 1.

1. In the Microsoft Azure PowerShell window, set a variable that specifies the name of the storage account that you created in step 1.

```
$storageAccountName = "vmstestsa"
```

2. Set a variable that specifies the primary key of the storage account.

```
$storageAccountKey = "<primary-account-key>"
```

   You can get this key by clicking the key icon when viewing the storage account resource in the Azure portal.

3. Create the storage account context object that is used to validate operations with the storage account.

```
$ctx = New-AzureStorageContext -StorageAccountName $storageAccountName -StorageAccountKey
$storageAccountKey
```

4. Create the container for storing the template.

```
$containerName = "templates"
New-AzureStorageContainer -Name $containerName -Context $ctx  -Permission Blob
```

5. Upload the template file to the new container.

```
$blobName = "VMSSTemplate.json"
$fileName = "C:\" + $BlobName
Set-AzureStorageBlobContent -File $fileName -Container $containerName -Blob  $blobName -Context $ctx
```

Step 5: Deploy the template

Now that you created the template, you can start deploying the resources. Use this command to start the process:

```
New-AzureRmResourceGroupDeployment -Name "vmsstestdp1" -ResourceGroupName "vmsstestrg1" -TemplateUri
"https://vmsstestsa.blob.core.windows.net/templates/VMSSTemplate.json"
```

When you press enter, you are prompted to provide values for the variables you assigned. Provide these values:

```
vmName: vmsstestvm1
  vmSSName: vmsstest1
  instanceCount: 5
  adminUserName: vmadmin1
  adminPassword: VMpass1
  resourcePrefix: vmsstest
```

It should take about 15 minutes for all the resources to successfully be deployed.

> **NOTE**
> You can also use the portal's ability to deploy the resources. Use this link: "https://portal.azure.com/#create/Microsoft.Template/uri/"

Step 6: Monitor resources

You can get some information about virtual machine scale sets using these methods:

- The Azure portal - You can currently get a limited amount of information using the portal.
- The Azure Resource Explorer - This tool is the best for exploring the current state of your scale set. Follow this path and you should see the instance view of the scale set that you created:

  ```
  subscriptions > {your subscription} > resourceGroups > vmsstestrg1 > providers > Microsoft.Compute >
  virtualMachineScaleSets > vmsstest1 > virtualMachines
  ```

- Azure PowerShell - Use this command to get some information:

  ```
  Get-AzureRmVmss -ResourceGroupName "resource group name" -VMScaleSetName "scale set name"

  Or

  Get-AzureRmVmss -ResourceGroupName "resource group name" -VMScaleSetName "scale set name" -InstanceView
  ```

- Connect to the separate virtual machine just like you would any other machine and then you can remotely access the virtual machines in the scale set to monitor individual processes.

> **NOTE**
> A complete REST API for obtaining information about scale sets can be found in Virtual Machine Scale Sets

Step 7: Remove the resources

Because you are charged for resources used in Azure, it is always a good practice to delete resources that are no longer needed. You don't need to delete each resource separately from a resource group. You can delete the resource group and all its resources are automatically deleted.

```
Remove-AzureRmResourceGroup -Name vmsstestrg1
```

If you want to keep your resource group, you can delete the scale set only.

```
Remove-AzureRmVmss -ResourceGroupName "resource group name" –VMScaleSetName "scale set name"
```

Next steps

- Manage the scale set that you just created using the information in Manage virtual machines in a Virtual Machine Scale Set.
- Learn more about vertical scaling by reviewing Vertical autoscale with Virtual Machine Scale sets
- Find examples of Azure Monitor monitoring features in Azure Monitor PowerShell quick start samples
- Learn about notification features in Use autoscale actions to send email and webhook alert notifications in Azure Monitor
- Learn how to Use audit logs to send email and webhook alert notifications in Azure Monitor

Use autoscale actions to send email and webhook alert notifications in Azure Monitor

1/17/2017 • 3 min to read • Edit on GitHub

This article shows you how set up triggers so that you can call specific web URLs or send emails based on autoscale actions in Azure.

Webhooks

Webhooks allow you to route the Azure alert notifications to other systems for post-processing or custom notifications. For example, routing the alert to services that can handle an incoming web request to send SMS, log bugs, notify a team using chat or messaging services, etc. The webhook URI must be a valid HTTP or HTTPS endpoint.

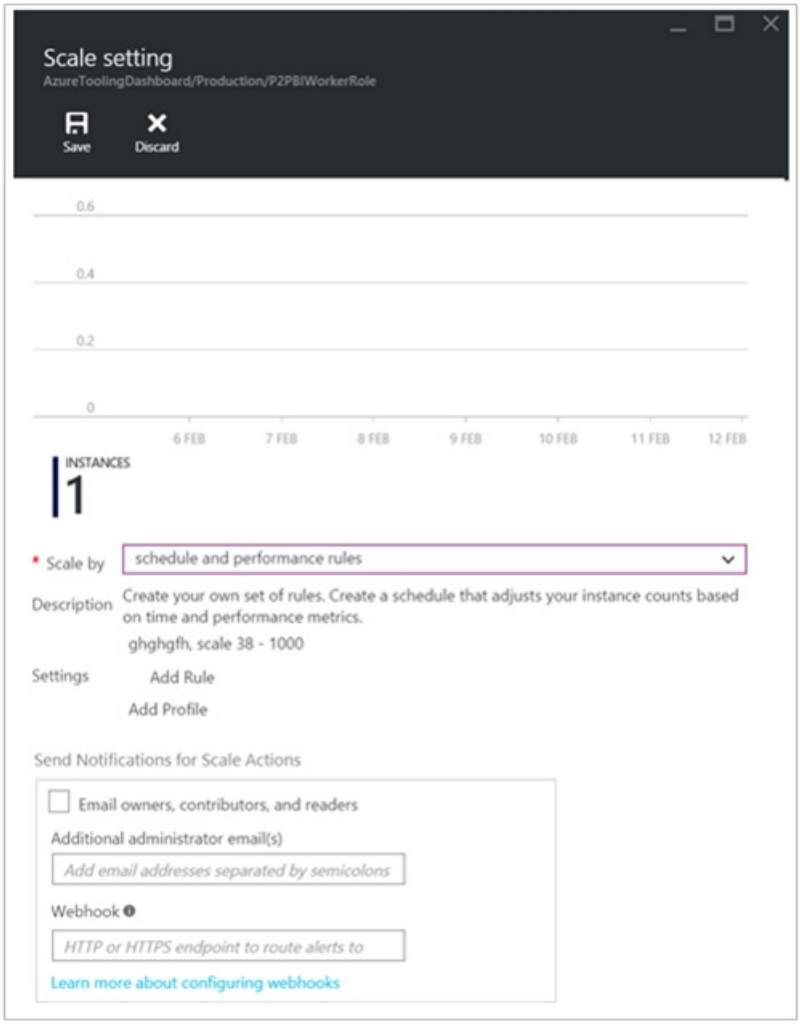
Email

Email can be sent to any valid email address. Administrators and co-administrators of the subscription where the rule is running will also be notified.

Cloud Services and Web Apps

You can opt-in from the Azure portal for Cloud Services and Server Farms (Web Apps).

- Choose the **scale by** metric.



Virtual Machine scale sets

For newer Virtual Machines created with Resource Manager (Virtual Machine scale sets), you can configure this using REST API, Resource Manager templates, PowerShell, and CLI. A portal interface is not yet available. When using the REST API or Resource Manager template, include the notifications element with the following options.

```
"notifications": [
      {
        "operation": "Scale",
        "email": {
          "sendToSubscriptionAdministrator": false,
          "sendToSubscriptionCoAdministrators": false,
          "customEmails": [
              "user1@mycompany.com",
              "user2@mycompany.com"
              ]
        },
        "webhooks": [
          {
            "serviceUri": "https://foo.webhook.example.com?token=abcd1234",
            "properties": {
              "optional_key1": "optional_value1",
              "optional_key2": "optional_value2"
            }
          }
        ]
      }
    ]
```

FIELD	MANDATORY?	DESCRIPTION
operation	yes	value must be "Scale"
sendToSubscriptionAdministrator	yes	value must be "true" or "false"
sendToSubscriptionCoAdministrators	yes	value must be "true" or "false"
customEmails	yes	value can be null [] or string array of emails
webhooks	yes	value can be null or valid Uri
serviceUri	yes	a valid https Uri
properties	yes	value must be empty {} or can contain key-value pairs

Authentication in webhooks

There are two authentication URI forms:

1. Token-base authentication, where you save the webhook URI with a token ID as a query parameter. For example, https://mysamplealert/webcallback?tokenid=sometokenid&someparameter=somevalue
2. Basic authentication, where you use a user ID and password. For example, https://userid:password@mysamplealert/webcallback?someparamater=somevalue¶meter=value

Autoscale notification webhook payload schema

When the autoscale notification is generated, the following metadata is included in the webhook payload:

```
{
        "version": "1.0",
        "status": "Activated",
        "operation": "Scale In",
        "context": {
                "timestamp": "2016-03-11T07:31:04.5834118Z",
                "id":
"/subscriptions/s1/resourceGroups/rg1/providers/microsoft.insights/autoscalesettings/myautoscaleSetting",
                "name": "myautoscaleSetting",
                "details": "Autoscale successfully started scale operation for resource 'MyCSRole' from capacity
'3' to capacity '2'",
                "subscriptionId": "s1",
                "resourceGroupName": "rg1",
                "resourceName": "MyCSRole",
                "resourceType": "microsoft.classiccompute/domainnames/slots/roles",
                "resourceId":
"/subscriptions/s1/resourceGroups/rg1/providers/microsoft.classicCompute/domainNames/myCloudService/slots/Produc
tion/roles/MyCSRole",
                "portalLink":
"https://portal.azure.com/#resource/subscriptions/s1/resourceGroups/rg1/providers/microsoft.classicCompute/domai
nNames/myCloudService",
                "oldCapacity": "3",
                "newCapacity": "2"
        },
        "properties": {
                "key1": "value1",
                "key2": "value2"
        }
}
```


FIELD	MANDATORY?	DESCRIPTION
status	yes	The status that indicates that an autoscale action was generated
operation	yes	For an increase of instances, it will be "Scale Out" and for a decrease in instances, it will be "Scale In"
context	yes	The autoscale action context
timestamp	yes	Time stamp when the autoscale action was triggered
id	Yes	Resource Manager ID of the autoscale setting
name	Yes	The name of the autoscale setting
details	Yes	Explanation of the action that the autoscale service took and the change in the instance count
subscriptionId	Yes	Subscription ID of the target resource that is being scaled
resourceGroupName	Yes	Resource Group name of the target resource that is being scaled

FIELD	MANDATORY?	DESCRIPTION
resourceName	Yes	Name of the target resource that is being scaled
resourceType	Yes	The three supported values: "microsoft.classiccompute/domainnames/slots/roles" - Cloud Service roles, "microsoft.compute/virtualmachinescalesets" - Virtual Machine Scale Sets, and "Microsoft.Web/serverfarms" - Web App
resourceId	Yes	Resource Manager ID of the target resource that is being scaled
portalLink	Yes	Azure portal link to the summary page of the target resource
oldCapacity	Yes	The current (old) instance count when Autoscale took a scale action
newCapacity	Yes	The new instance count that Autoscale scaled the resource to
Properties	No	Optional. Set of pairs (for example, Dictionary). The properties field is optional. In a custom user interface or Logic app based workflow, you can enter keys and values that can be passed using the payload. An alternate way to pass custom properties back to the outgoing webhook call is to use the webhook URI itself (as query parameters)

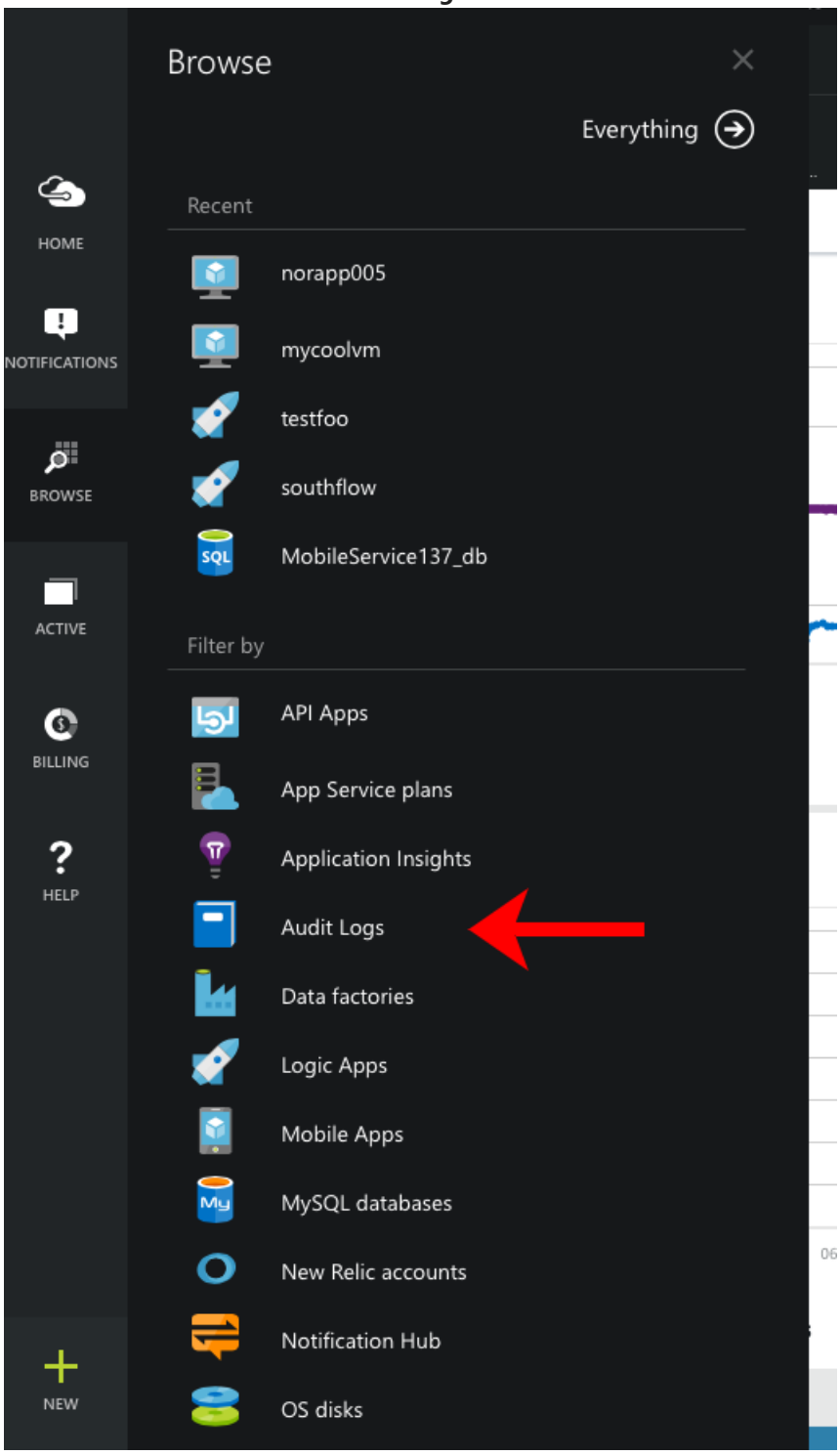
View events and activity logs

1/17/2017 • 2 min to read • Edit on GitHub

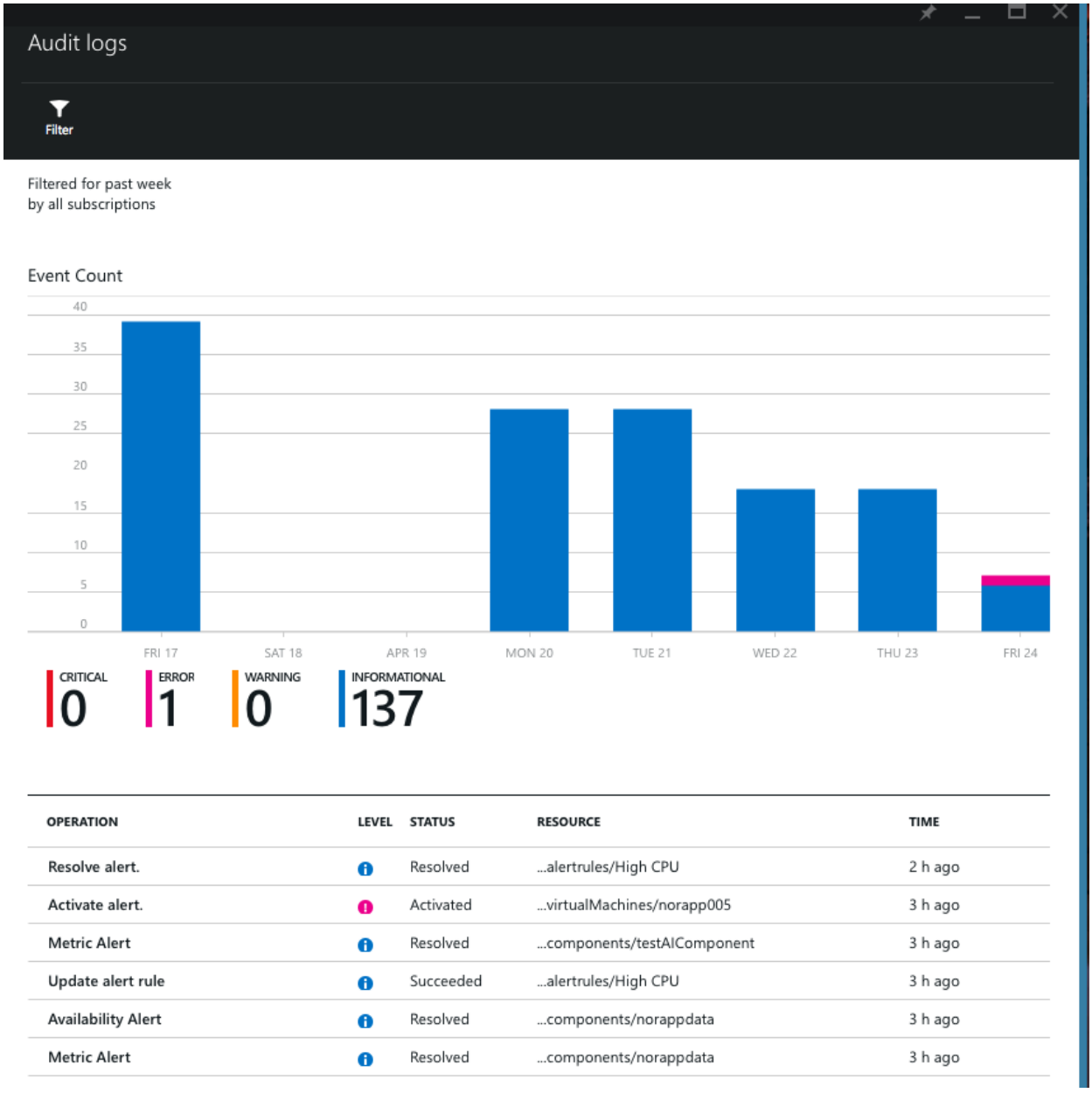
All operations performed on Azure resources are fully audited by the Azure Resource Manager, from creation and deletions to granting or revoking access. You can browse these logs in the Azure portal, and you can also use the REST API or .NET SDK to access the full set of events programmatically.

Browse the events impacting your Azure subscription

1. Sign in to the Azure Portal.
2. Click on the **Browse** and select **Audit logs**.

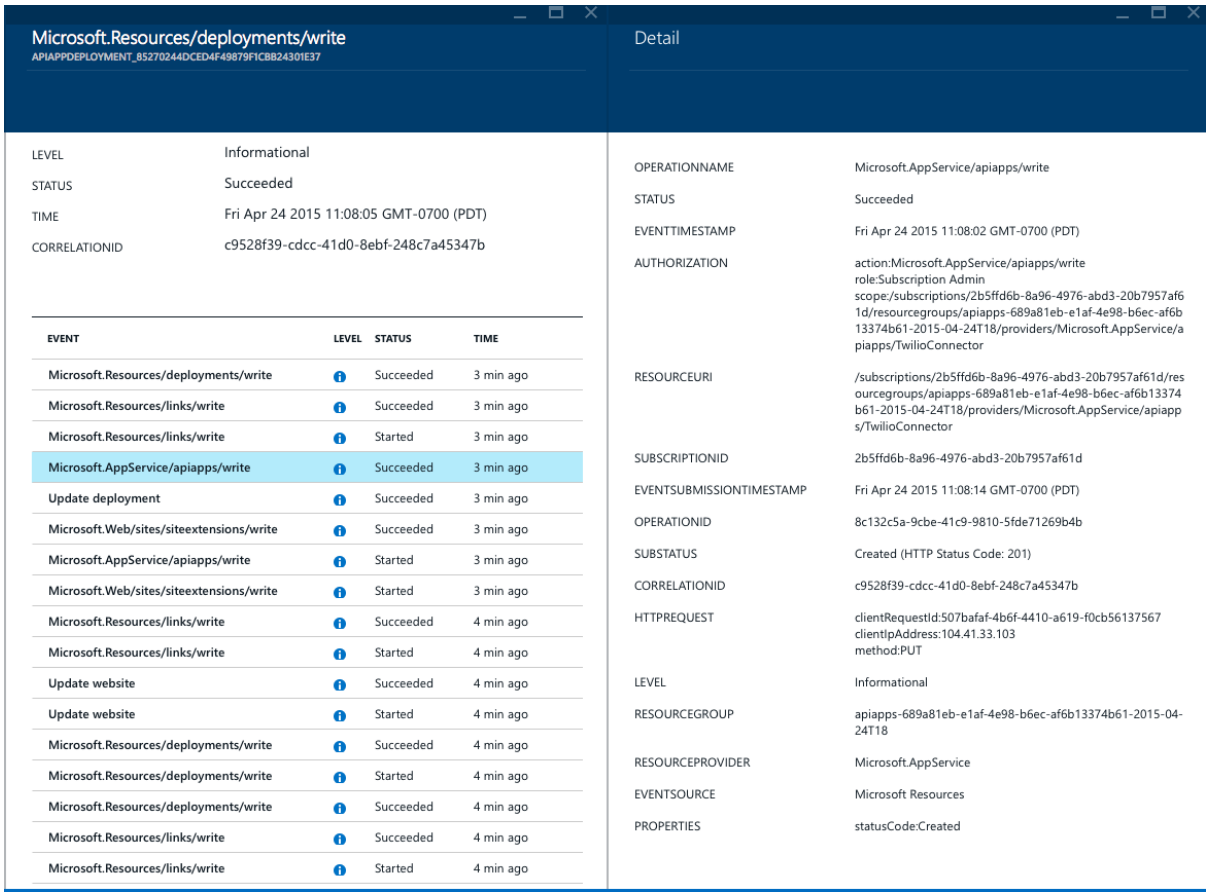


3. This will open up a blade showing all of the events that have impacted any of your subscriptions for the past 7 days. At the top is a chart showing data by level, and below that is the full list of logs:



> **NOTE**
>
> You can only view the 500 most recent events for a given subscription in the Azure portal.

1. You can click on any log entry to see the events that made it up. For example, when you deploy something to a resource group, many different resources may be created or modified. For each entry you can see:
   - The **Level** of the event - for example, it could be just something to track (**Informational**), or when something has gone wrong that you need to know about (**Error**).
   - The **Status** - the final status will generally be **Succeeded** or **Failed**, but it may also be **Accepted** for long-running operations.
   - *When* the event occurred.
   - *Who* performed the operation, if anyone. Not all operations are performed by users, some are performed by backend services so they would not have a **Caller**.
   - The **Correlation ID** of the event - this is the unique identifier for this set of operations.
2. From there you can go to the details blade to see the specifics of the event.

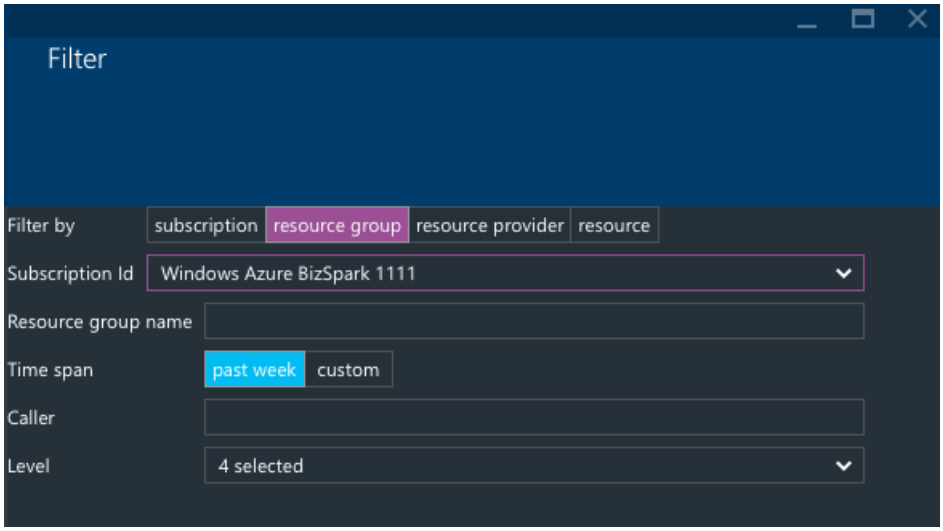


For **Failed** events, this page usually includes a **Substatus** and a **Properties** section that include useful details for debugging purposes.

Filter to specific logs

In order to see events that apply to a specific entity, or of a specific type, you can filter the Audit logs blade by clicking the **Filter** command. You also use the Filter blade to change the **Time span** of the Audit logs blade.

Once you click this command, a new blade will open:



There are four types of filters:

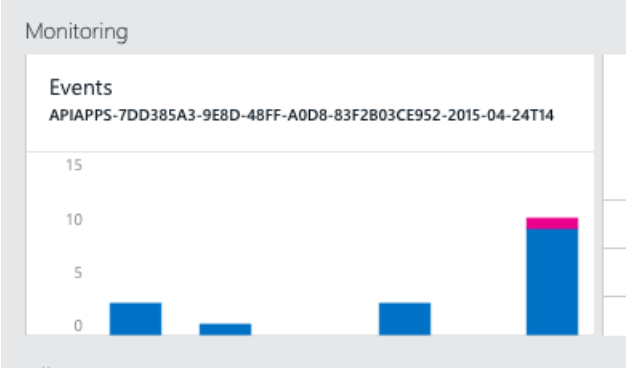
1. By subscription
2. By a **Resource group**
3. By a **Resource type**
4. By a particular **Resource** - for this you must past in the full *Resource ID* of the resource you are interested in

In addition, you can also filter events by who performed the event, or, by the level of the event.

Once you have finished choosing what you want to see, click the **Update** button at the bottom of the blade.

Monitor events impacting specific resources

1. Click on **Browse** to find the resource you are interested in. You can also see all of the logs for an entire **Resource group**.
2. On the resource's blade, scroll down until you find the **Events** tile.



3. Click on that tile to see events filtered to just the resource that you selected. You can use the **Filter** command to change the time range or apply more specific filters.

Next steps

- Receive alert notifications whenever an event happens.
- Monitor service metrics to make sure your service is available and responsive.
- Track service health to find out when Azure has experienced performance degradation or service interruptions.

Configure a webhook on an Azure Activity Log alert

1/17/2017 • 3 min to read • Edit on GitHub

Webhooks allow you to route an Azure alert notification to other systems for post-processing or custom actions. You can use a webhook on an alert to route it to services that send SMS, log bugs, notify a team via chat/messaging services, or do any number of other actions. This article describes how to set a webhook on an Azure Activity Log alert and what the payload for the HTTP POST to a webhook looks like. For information on the setup and schema for an Azure metric alert, see this page instead. You can also set up an Activity Log alert to send email when activated.

> **NOTE**
>
> This feature is currently in preview and will be removed at some point in the future.

You can set up an Activity Log alert using the Azure PowerShell Cmdlets, Cross-Platform CLI, or Azure Monitor REST API.

Authenticating the webhook

TThe webhook can authenticate using either of these methods:

1. **Token-based authorization** - The webhook URI is saved with a token ID, eg.
   `https://mysamplealert/webcallback?tokenid=sometokenid&someparameter=somevalue`
2. **Basic authorization** - The webhook URI is saved with a username and password, eg.
   `https://userid:password@mysamplealert/webcallback?someparamater=somevalue&foo=bar`

Payload schema

The POST operation contains the following JSON payload and schema for all Activity Log-based alerts. This schema is similar to the one used by metric-based alerts.

```
{
        "status": "Activated",
        "context": {
                "resourceProviderName": "Microsoft.Web",
                "event": {
                        "$type":
"Microsoft.WindowsAzure.Management.Monitoring.Automation.Notifications.GenericNotifications.Datacontracts.Instan
ceEventContext, Microsoft.WindowsAzure.Management.Mon.Automation",
                        "authorization": {
                                "action": "Microsoft.Web/sites/start/action",
                                "scope":
"/subscriptions/s1/resourcegroups/rg1/providers/Microsoft.Web/sites/leoalerttest"
                        },
                        "eventDataId": "327caaca-08d7-41b1-86d8-27d0a7adb92d",
                        "category": "Administrative",
                        "caller": "myname@mycompany.com",
                        "httpRequest": {
                                "clientRequestId": "f58cead8-c9ed-43af-8710-55e64def208d",
                                "clientIpAddress": "104.43.166.155",
                                "method": "POST"
                        },
                        "status": "Succeeded",
                        "subStatus": "OK",
                        "level": "Informational",
                        "correlationId": "4a40beaa-6a63-4d92-85c4-923a25abb590",
                        "eventDescription": "",
                        "operationName": "Microsoft.Web/sites/start/action",
                        "operationId": "4a40beaa-6a63-4d92-85c4-923a25abb590",
                        "properties": {
                                "$type":
"Microsoft.WindowsAzure.Management.Common.Storage.CasePreservedDictionary,
Microsoft.WindowsAzure.Management.Common.Storage",
                                "statusCode": "OK",
                                "serviceRequestId": "f7716681-496a-4f5c-8d14-d564bcf54714"
                        }
                },
                "timestamp": "Friday, March 11, 2016 9:13:23 PM",
                "id":
"/subscriptions/s1/resourceGroups/rg1/providers/microsoft.insights/alertrules/alertonevent2",
                "name": "alertonevent2",
                "description": "test alert on event start",
                "conditionType": "Event",
                "subscriptionId": "s1",
                "resourceId": "/subscriptions/s1/resourcegroups/rg1/providers/Microsoft.Web/sites/leoalerttest",
                "resourceGroupName": "rg1"
        },
        "properties": {
                "key1": "value1",
                "key2": "value2"
        }
}
```


ELEMENT NAME	DESCRIPTION
status	Used for metric alerts. Always set to "activated" for Activity Log alerts.
context	Context of the event.
resourceProviderName	The resource provider of the impacted resource.
conditionType	Always "Event."

ELEMENT NAME	DESCRIPTION
name	Name of the alert rule.
id	Resource ID of the alert.
description	Alert description as set during creation of the alert.
subscriptionId	Azure Subscription ID.
timestamp	Time at which the event was generated by the Azure service that processed the request.
resourceId	Resource ID of the impacted resource.
resourceGroupName	Name of the resource group for the impacted resource
properties	Set of `<Key, Value>` pairs (i.e. `Dictionary<String, String>`) that includes details about the event.
event	Element containing metadata about the event.
authorization	The RBAC properties of the event. These usually include the “action”, “role” and the “scope.”
category	Category of the event. Supported values include: Administrative, Alert, Security, ServiceHealth, Recommendation.
caller	Email address of the user who performed the operation, UPN claim, or SPN claim based on availability. Can be null for certain system calls.
correlationId	Usually a GUID in string format. Events with correlationId belong to the same larger action and usually share a correlationId.
eventDescription	Static text description of the event.
eventDataId	Unique identifier for the event.
eventSource	Name of the Azure service or infrastructure that generated the event.
httpRequest	Usually includes the “clientRequestId”, “clientIpAddress” and “method” (HTTP method e.g. PUT).
level	One of the following values: “Critical”, “Error”, “Warning”, “Informational” and “Verbose.”
operationId	Usually a GUID shared among the events corresponding to single operation.
operationName	Name of the operation.

ELEMENT NAME	DESCRIPTION
properties	Properties of the event.
status	String. Status of the operation. Common values include: "Started", "In Progress", "Succeeded", "Failed", "Active", "Resolved".
subStatus	Usually includes the HTTP status code of the corresponding REST call. It might also include other strings describing a substatus. Common substatus values include: OK (HTTP Status Code: 200), Created (HTTP Status Code: 201), Accepted (HTTP Status Code: 202), No Content (HTTP Status Code: 204), Bad Request (HTTP Status Code: 400), Not Found (HTTP Status Code: 404), Conflict (HTTP Status Code: 409), Internal Server Error (HTTP Status Code: 500), Service Unavailable (HTTP Status Code: 503), Gateway Timeout (HTTP Status Code: 504)

Next steps

- Learn more about the Activity Log
- Execute Azure Automation scripts (Runbooks) on Azure alerts
- Use Logic App to send an SMS via Twilio from an Azure alert. This example is for metric alerts, but could be modified to work with an Activity Log alert.
- Use Logic App to send a Slack message from an Azure alert. This example is for metric alerts, but could be modified to work with an Activity Log alert.
- Use Logic App to send a message to an Azure Queue from an Azure alert. This example is for metric alerts, but could be modified to work with an Activity Log alert.

Archive the Azure Activity Log

1/17/2017 • 5 min to read • Edit on GitHub

In this article, we show how you can use the Azure portal, PowerShell Cmdlets, or Cross-Platform CLI to archive your **Azure Activity Log** in a storage account. This option is useful if you would like to retain your Activity Log longer than 90 days (with full control over the retention policy) for audit, static analysis, or backup. If you only need to retain your events for 90 days or less you do not need to set up archival to a storage account, since Activity Log events are retained in the Azure platform for 90 days without enabling archival.

Prerequisites

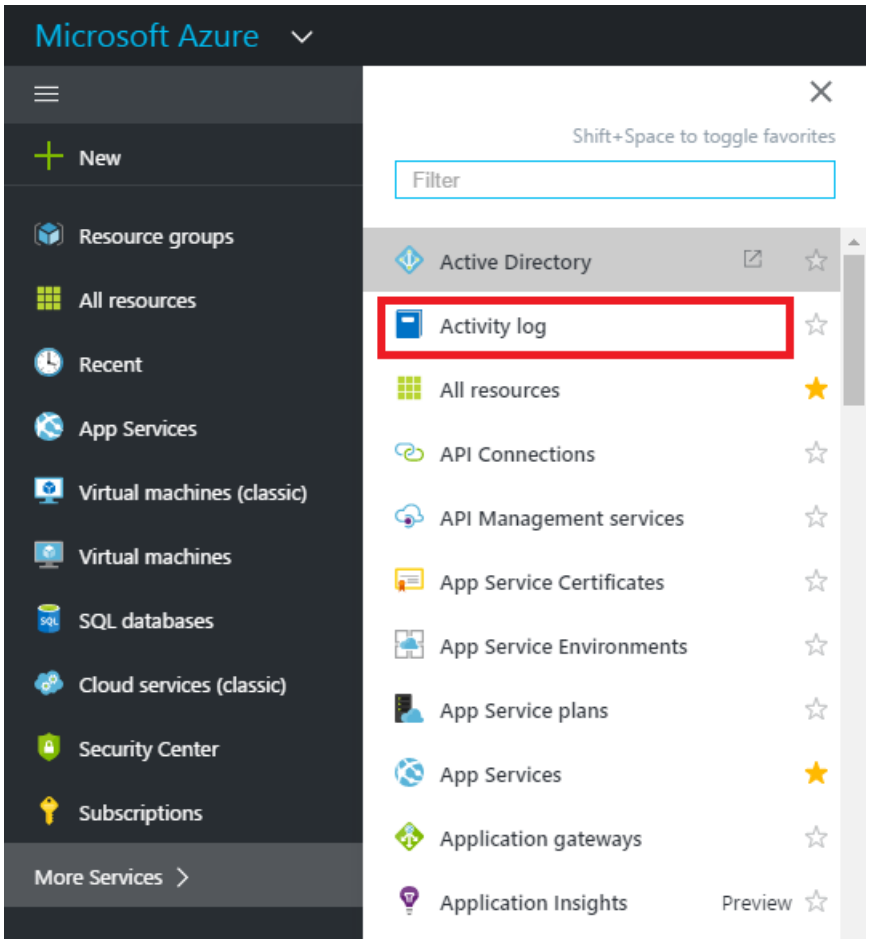
Before you begin, you need to create a storage account to which you can archive your Activity Log. We highly recommend that you do not use an existing storage account that has other, non-monitoring data stored in it so that you can better control access to monitoring data. However, if you are also archiving Diagnostic Logs and metrics to a storage account, it may make sense to use that storage account for your Activity Log as well to keep all monitoring data in a central location. The storage account you use must be a general purpose storage account, not a blob storage account. The storage account does not have to be in the same subscription as the subscription emitting logs as long as the user who configures the setting has appropriate RBAC access to both subscriptions.

Log Profile

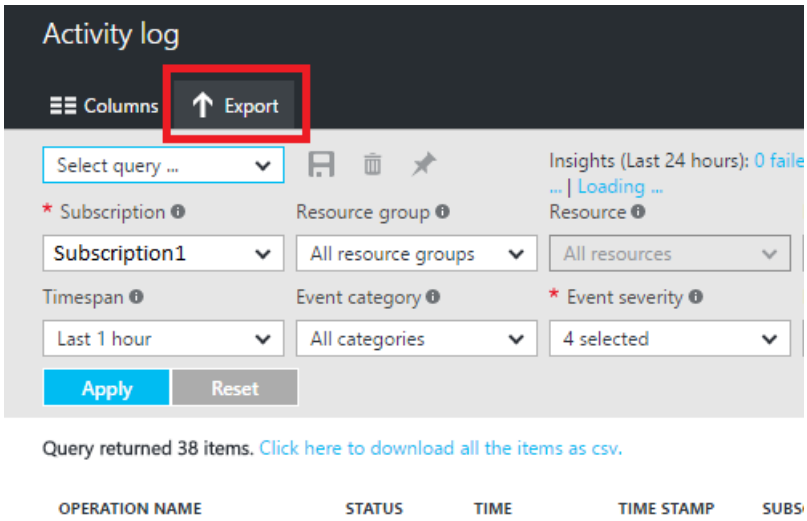
To archive the Activity Log using any of the methods below, you set the **Log Profile** for a subscription. The Log Profile defines the type of events that are stored or streamed and the outputs—storage account and/or event hub. It also defines the retention policy (number of days to retain) for events stored in a storage account. If the retention policy is set to zero, events are stored indefinitely. Otherwise, this can be set to any value between 1 and 2147483647. Retention policies are applied per-day, so at the end of a day (UTC), logs from the day that is now beyond the retention policy will be deleted. For example, if you had a retention policy of one day, at the beginning of the day today the logs from the day before yesterday would be deleted. You can read more about log profiles here.

Archive the Activity Log using the portal

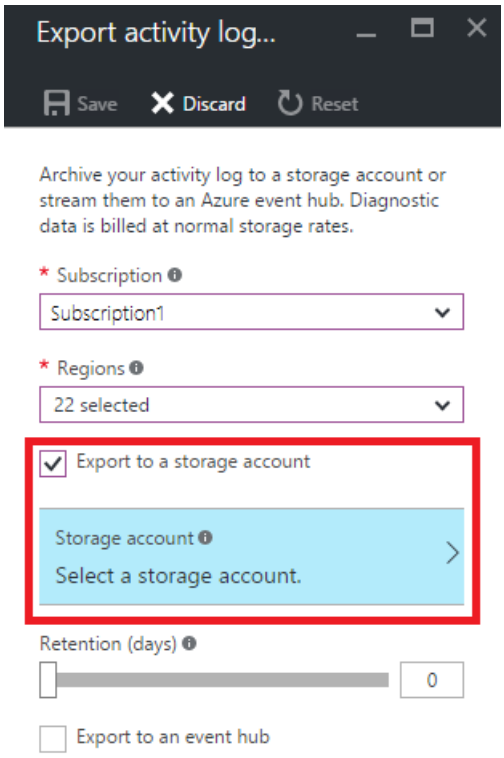
1. In the portal, click the **Activity Log** link on the left-side navigation. If you don't see a link for the Activity Log, click the **More Services** link first.



2. At the top of the blade, click **Export**.



3. In the blade that appears, check the box for **Export to a storage account** and select a storage account.



4. Using the slider or text box, define a number of days for which Activity Log events should be kept in your storage account. If you prefer to have your data persisted in the storage account indefinitely, set this number to zero.
5. Click **Save**.

Archive the Activity Log via PowerShell

```
Add-AzureRmLogProfile -Name my_log_profile -StorageAccountId
/subscriptions/s1/resourceGroups/myrg1/providers/Microsoft.Storage/storageAccounts/my_storage -Locations
global,westus,eastus -RetentionInDays 180 -Categories Write,Delete,Action
```

PROPERTY	REQUIRED	DESCRIPTION
StorageAccountId	No	Resource ID of the Storage Account to which Activity Logs should be saved.
Locations	Yes	Comma-separated list of regions for which you would like to collect Activity Log events. You can view a list of all regions by visiting this page or by using the Azure Management REST API.
RetentionInDays	Yes	Number of days for which events should be retained, between 1 and 2147483647. A value of zero stores the logs indefinitely (forever).
Categories	Yes	Comma-separated list of event categories that should be collected. Possible values are Write, Delete, and Action.

Archive the Activity Log via CLI

```
azure insights logprofile add --name my_log_profile --storageId /subscriptions/s1/resourceGroups/insights-integration/providers/Microsoft.Storage/storageAccounts/my_storage --locations global,westus,eastus,northeurope --retentionInDays 180 –categories Write,Delete,Action
```

PROPERTY	REQUIRED	DESCRIPTION
name	Yes	Name of your log profile.
storageId	No	Resource ID of the Storage Account to which Activity Logs should be saved.
locations	Yes	Comma-separated list of regions for which you would like to collect Activity Log events. You can view a list of all regions by visiting this page or by using the Azure Management REST API.
retentionInDays	Yes	Number of days for which events should be retained, between 1 and 2147483647. A value of zero will store the logs indefinitely (forever).
categories	Yes	Comma-separated list of event categories that should be collected. Possible values are Write, Delete, and Action.

Storage schema of the Activity Log

Once you have set up archival, a storage container will be created in the storage account as soon as an Activity Log event occurs. The blobs within the container follow the same format across the Activity Log and Diagnostic Logs. The structure of these blobs is:

> insights-operational-logs/name=default/resourceId=/SUBSCRIPTIONS/{subscription ID}/y={four-digit numeric year}/m={two-digit numeric month}/d={two-digit numeric day}/h={two-digit 24-hour clock hour}/m=00/PT1H.json

For example, a blob name might be:

> insights-operational-logs/name=default/resourceId=/SUBSCRIPTIONS/s1id1234-5679-0123-4567-890123456789/y=2016/m=08/d=22/h=18/m=00/PT1H.json

Each PT1H.json blob contains a JSON blob of events that occurred within the hour specified in the blob URL (e.g. h=12). During the present hour, events are appended to the PT1H.json file as they occur. The minute value (m=00) is always 00, since Activity Log events are broken into individual blobs per hour.

Within the PT1H.json file, each event is stored in the “records” array, following this format:

```
{
    "records": [
        {
            "time": "2015-01-21T22:14:26.9792776Z",
            "resourceId":
"/subscriptions/s1/resourceGroups/MSSupportGroup/providers/microsoft.support/supporttickets/115012112305841",
            "operationName": "microsoft.support/supporttickets/write",
            "category": "Write",
            "resultType": "Success",
            "resultSignature": "Succeeded.Created",
            "durationMs": 2826,
            "callerIpAddress": "111.111.111.11",
            "correlationId": "c776f9f4-36e5-4e0e-809b-c9b3c3fb62a8",
            "identity": {
                "authorization": {
                    "scope":
"/subscriptions/s1/resourceGroups/MSSupportGroup/providers/microsoft.support/supporttickets/115012112305841",
                    "action": "microsoft.support/supporttickets/write",
                    "evidence": {
                        "role": "Subscription Admin"
                    }
                },
                "claims": {
                    "aud": "https://management.core.windows.net/",
                    "iss": "https://sts.windows.net/72f988bf-86f1-41af-91ab-2d7cd011db47/",
                    "iat": "1421876371",
                    "nbf": "1421876371",
                    "exp": "1421880271",
                    "ver": "1.0",
                    "http://schemas.microsoft.com/identity/claims/tenantid": "1e8d8218-c5e7-4578-9acc-
9abbd5d23315 ",
                    "http://schemas.microsoft.com/claims/authnmethodsreferences": "pwd",
                    "http://schemas.microsoft.com/identity/claims/objectidentifier": "2468adf0-8211-44e3-95xq-
85137af64708",
                    "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn": "admin@contoso.com",
                    "puid": "20030000801A118C",
                    "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier":
"9vckmEGF7zDKk1YzIY8k0t1_EAPaXoeHyPRn6f413zM",
                    "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname": "John",
                    "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname": "Smith",
                    "name": "John Smith",
                    "groups": "cacfe77c-e058-4712-83qw-f9b08849fd60,7f71d11d-4c41-4b23-99d2-
d32ce7aa621c,31522864-0578-4ea0-9gdc-e66cc564d18c",
                    "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name": " admin@contoso.com",
                    "appid": "c44b4083-3bq0-49c1-b47d-974e53cbdf3c",
                    "appidacr": "2",
                    "http://schemas.microsoft.com/identity/claims/scope": "user_impersonation",
                    "http://schemas.microsoft.com/claims/authnclassreference": "1"
                }
            },
            "level": "Information",
            "location": "global",
            "properties": {
                "statusCode": "Created",
                "serviceRequestId": "50d5cddb-8ca0-47ad-9b80-6cde2207f97c"
            }
        }
    ]
}
```


ELEMENT NAME	DESCRIPTION
time	Timestamp when the event was generated by the Azure service processing the request corresponding the event.

ELEMENT NAME	DESCRIPTION
resourceId	Resource ID of the impacted resource.
operationName	Name of the operation.
category	Category of the action, eg. Write, Read, Action.
resultType	The type of the result, eg. Success, Failure, Start
resultSignature	Depends on the resource type.
durationMs	Duration of the operation in milliseconds
callerIpAddress	IP address of the user who has performed the operation, UPN claim, or SPN claim based on availability.
correlationId	Usually a GUID in the string format. Events that share a correlationId belong to the same uber action.
identity	JSON blob describing the authorization and claims.
authorization	Blob of RBAC properties of the event. Usually includes the "action", "role" and "scope" properties.
level	Level of the event. One of the following values: "Critical", "Error", "Warning", "Informational" and "Verbose"
location	Region in which the location occurred (or global).
properties	Set of `<Key, Value>` pairs (i.e. Dictionary) describing the details of the event.

> **NOTE**
> The properties and usage of those properties can vary depending on the resource.

Next steps

- Download blobs for analysis
- Stream the Activity Log to Event Hubs
- Read more about the Activity Log

Stream the Azure Activity Log to Event Hubs

1/17/2017 • 3 min to read • Edit on GitHub

The **Azure Activity Log** can be streamed in near real time to any application using the built-in "Export" option in the portal, or by enabling the Service Bus Rule Id in a Log Profile via the Azure PowerShell Cmdlets or Azure CLI.

What you can do with the Activity Log and Event Hubs

Here are just a few ways you might use the streaming capability for the Activity Log:

- **Stream to third-party logging and telemetry systems** – Over time, Event Hubs streaming will become the mechanism to pipe your Activity Log into third-party SIEMs and log analytics solutions.
- **Build a custom telemetry and logging platform** – If you already have a custom-built telemetry platform or are just thinking about building one, the highly scalable publish-subscribe nature of Event Hubs allows you to flexibly ingest the activity log. See Dan Rosanova's guide to using Event Hubs in a global scale telemetry platform here.

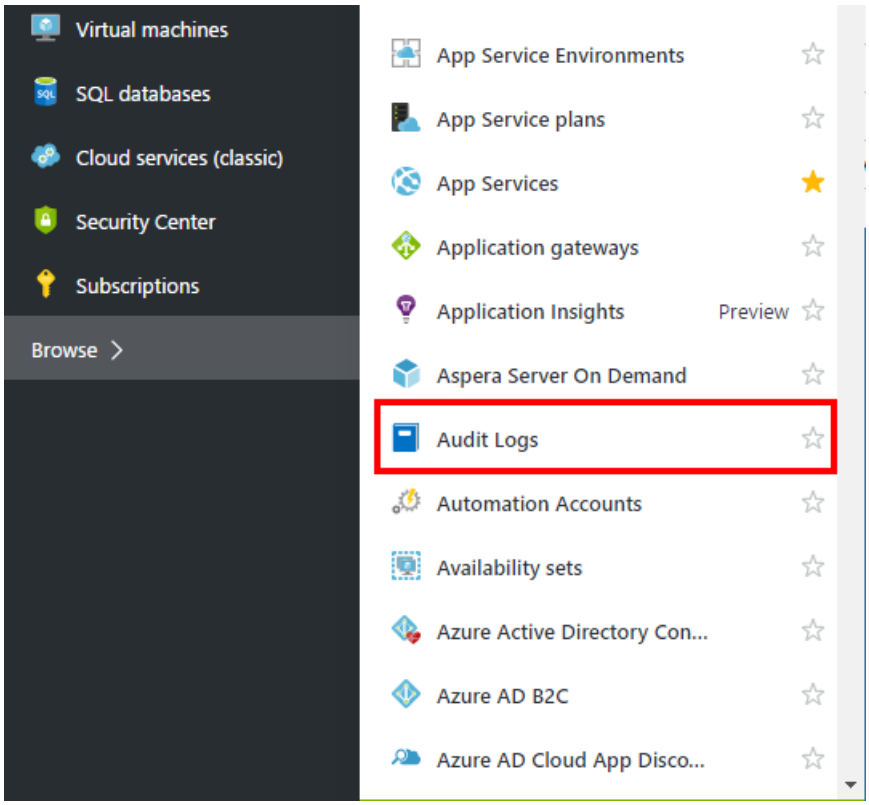
Enable streaming of the Activity Log

You can enable streaming of the Activity Log either programmatically or via the portal. Either way, you pick a Service Bus Namespace and a shared access policy for that namespace, and an Event Hub is created in that namespace when the first new Activity Log event occurs. If you do not have a Service Bus Namespace, you first need to create one. If you have previously streamed Activity Log events to this Service Bus Namespace, the Event Hub that was previously created will be reused. The shared access policy defines the permissions that the streaming mechanism has. Today, streaming to an Event Hubs requires **Manage**, **Send**, and **Listen** permissions. You can create or modify Service Bus Namespace shared access policies in the classic portal under the "Configure" tab for your Service Bus Namespace. To update the Activity Log log profile to include streaming, the user making the change must have the ListKey permission on that Service Bus Authorization Rule.

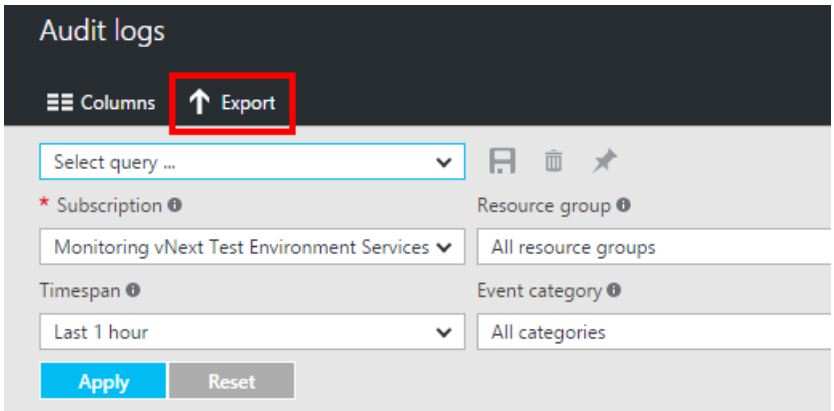
The service bus or event hub namespace does not have to be in the same subscription as the subscription emitting logs as long as the user who configures the setting has appropriate RBAC access to both subscriptions.

Via Azure portal

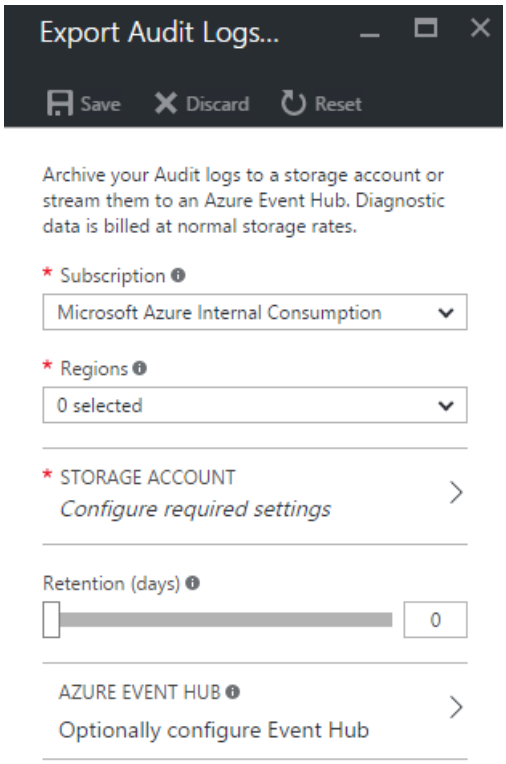
1. Navigate to the **Activity Log** blade using the menu on the left side of the portal.



2. Click the **Export** button at the top of the blade.



3. In the blade that appears, you can select the regions for which you would like to stream events and the Service Bus Namespace in which you would like an Event Hub to be created for streaming these events.



4. Click **Save** to save these settings. The settings are immediately be applied to your subscription.

Via PowerShell Cmdlets

If a log profile already exists, you first need to remove that profile.

1. Use `Get-AzureRmLogProfile` to identify if a log profile exists
2. If so, use `Remove-AzureRmLogProfile` to remove it.
3. Use `Set-AzureRmLogProfile` to create a profile:

```
Add-AzureRmLogProfile -Name my_log_profile -StorageAccountId
/subscriptions/s1/resourceGroups/myrg1/providers/Microsoft.Storage/storageAccounts/my_storage -serviceBusRuleId
/subscriptions/s1/resourceGroups/Default-ServiceBus-
EastUS/providers/Microsoft.ServiceBus/namespaces/mytestSB/authorizationrules/RootManageSharedAccessKey -
Locations global,westus,eastus -RetentionInDays 90 -Categories Write,Delete,Action
```

The Service Bus Rule ID is a string with this format: {service bus resource ID}/authorizationrules/{key name}, for example

Via Azure CLI

If a log profile already exists, you first need to remove that profile.

1. Use `azure insights logprofile list` to identify if a log profile exists
2. If so, use `azure insights logprofile delete` to remove it.
3. Use `azure insights logprofile add` to create a profile:

```
azure insights logprofile add --name my_log_profile --storageId /subscriptions/s1/resourceGroups/insights-
integration/providers/Microsoft.Storage/storageAccounts/my_storage --serviceBusRuleId
/subscriptions/s1/resourceGroups/Default-ServiceBus-
EastUS/providers/Microsoft.ServiceBus/namespaces/mytestSB/authorizationrules/RootManageSharedAccessKey --
locations global,westus,eastus,northeurope --retentionInDays 90 –categories Write,Delete,Action
```

The Service Bus Rule ID is a string with this format: `{service bus resource ID}/authorizationrules/{key name}`.

How do I consume the log data from Event Hubs?

The schema for the Activity Log is available here. Each event is in an array of JSON blobs called "records."

Next Steps

- Archive the Activity Log to a storage account
- Read the overview of the Azure Activity Log
- Set up an alert based on an Activity Log event

View activity logs to manage Azure resources

1/17/2017 • 3 min to read • Edit on GitHub

Through activity logs, you can determine:

- what operations were taken on the resources in your subscription
- who initiated the operation (although operations initiated by a backend service do not return a user as the caller)
- when the operation occurred
- the status of the operation
- the values of other properties that might help you research the operation

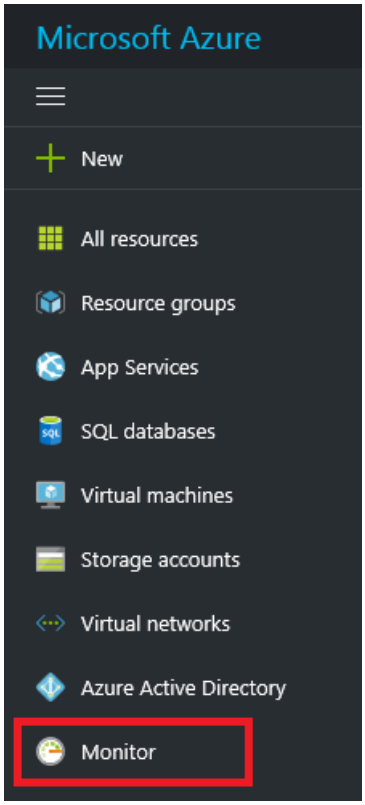
The activity log contains all write operations (PUT, POST, DELETE) performed on your resources. It does not include read operations (GET). You can use the audit logs to find an error when troubleshooting or to monitor how a user in your organization modified a resource.

Activity logs are retained for 90 days. You can query for any range of dates, as long as the starting date is not more than 90 days in the past.

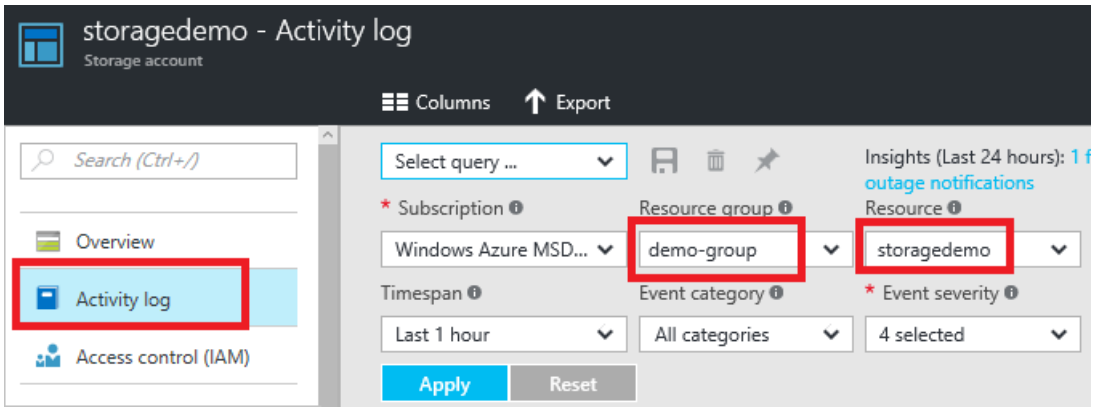
You can retrieve information from the activity logs through the portal, PowerShell, Azure CLI, Insights REST API, or Insights .NET Library.

Portal

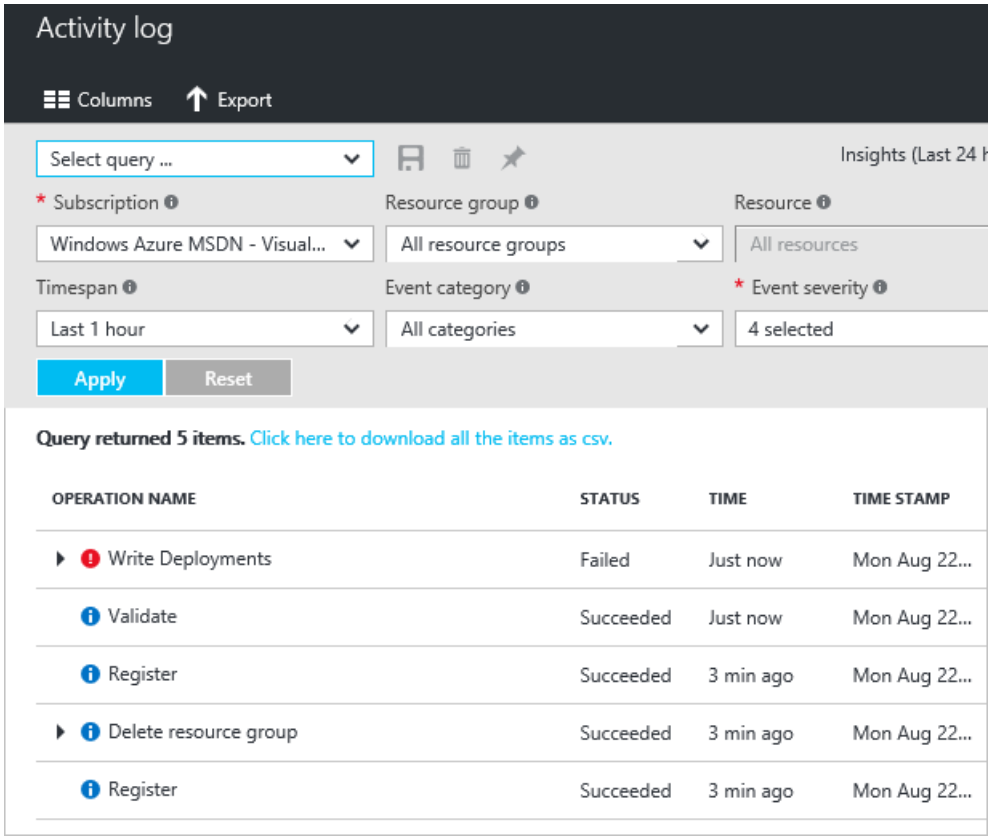
1. To view the activity logs through the portal, select **Monitor**.



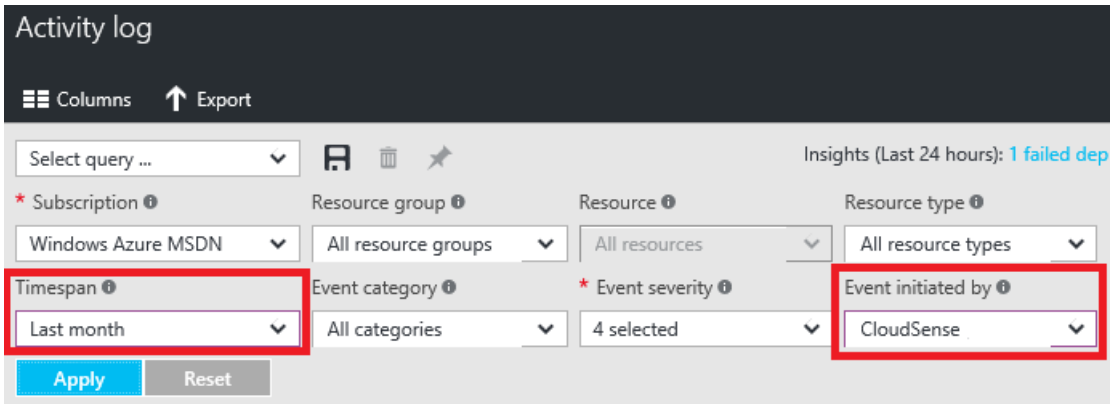
Or, to automatically filter the activity log for a particular resource or resource group, select **Activity log** from that resource blade. Notice that the activity log is automatically filtered by the selected resource.



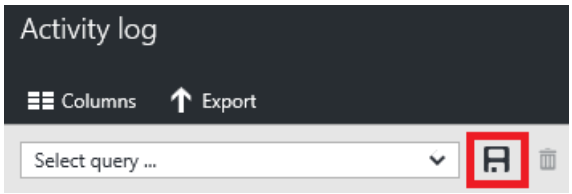
2. In the **Activity Log** blade, you see a summary of recent operations.



3. To restrict the number of operations displayed, select different conditions. For example, the following image shows the **Timespan** and **Event initiated by** fields changed to view the actions taken by a particular user or application for the past month. Select **Apply** to view the results of your query.

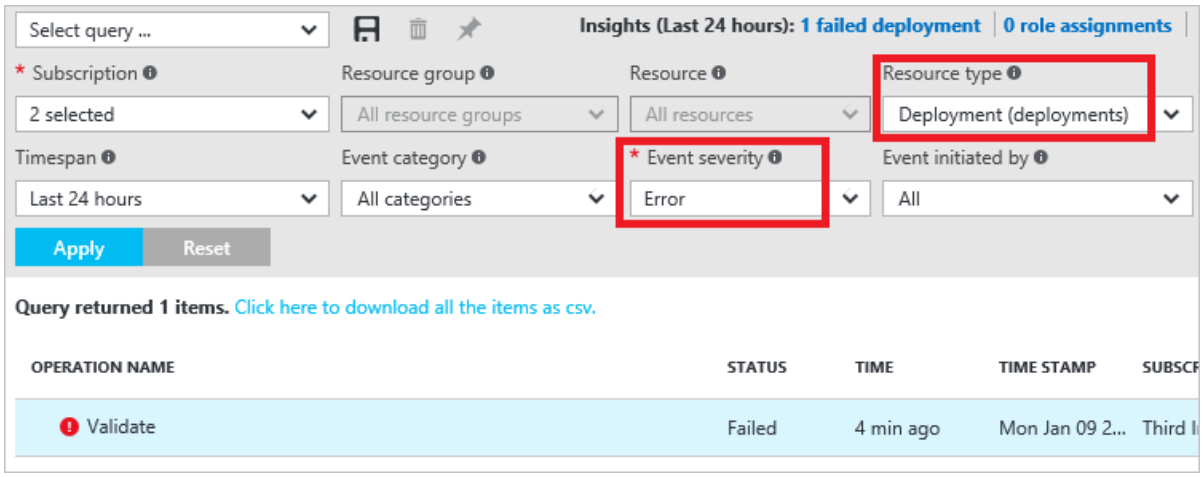


4. If you need to run the query again later, select **Save** and give the query a name.

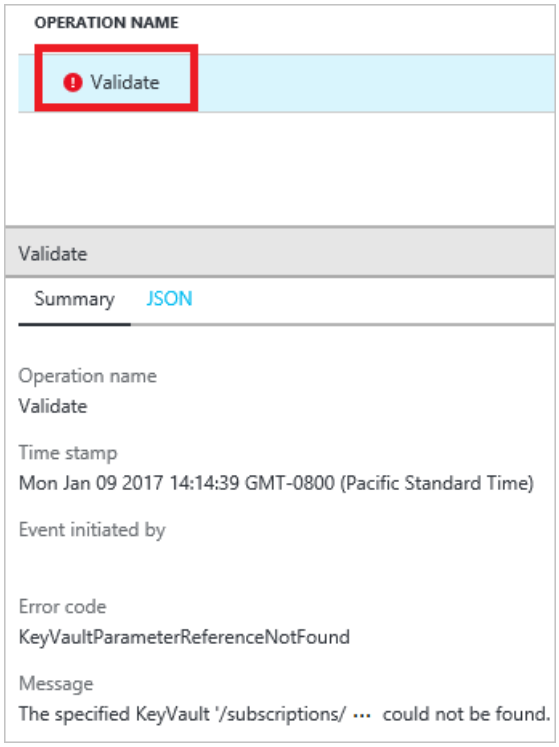


5. To quickly run a query, you can select one of the built-in queries, such as failed deployments.

The selected query automatically sets the required filter values.



6. Select one of the operations to see a summary of the event.



PowerShell

1. To retrieve log entries, run the **Get-AzureRmLog** command. You provide additional parameters to filter the list of entries. If you do not specify a start and end time, entries for the last hour are returned. For example, to retrieve the operations for a resource group during the past hour run:

```
Get-AzureRmLog -ResourceGroup ExampleGroup
```

The following example shows how to use the activity log to research operations taken during a specified time. The start and end dates are specified in a date format.

```
Get-AzureRmLog -ResourceGroup ExampleGroup -StartTime 2015-08-28T06:00 -EndTime 2015-09-10T06:00
```

Or, you can use date functions to specify the date range, such as the last 14 days.

```
Get-AzureRmLog -ResourceGroup ExampleGroup -StartTime (Get-Date).AddDays(-14)
```

2. Depending on the start time you specify, the previous commands can return a long list of operations for the resource group. You can filter the results for what you are looking for by providing search criteria. For example, if you are trying to research how a web app was stopped, you could run the following command:

```
Get-AzureRmLog -ResourceGroup ExampleGroup -StartTime (Get-Date).AddDays(-14) | Where-Object
OperationName -eq Microsoft.Web/sites/stop/action
```

Which for this example shows that a stop action was performed by someone@contoso.com.

```
Authorization     :
Scope     : /subscriptions/xxxxx/resourcegroups/ExampleGroup/providers/Microsoft.Web/sites/ExampleSite
Action    : Microsoft.Web/sites/stop/action
Role      : Subscription Admin
Condition :
Caller            : someone@contoso.com
CorrelationId     : 84beae59-92aa-4662-a6fc-b6fecc0ff8da
EventSource       : Administrative
EventTimestamp    : 8/28/2015 4:08:18 PM
OperationName     : Microsoft.Web/sites/stop/action
ResourceGroupName : ExampleGroup
ResourceId        :
/subscriptions/xxxxx/resourcegroups/ExampleGroup/providers/Microsoft.Web/sites/ExampleSite
Status            : Succeeded
SubscriptionId    : xxxxx
SubStatus         : OK
```

3. You can look up the actions taken by a particular user, even for a resource group that no longer exists.

```
Get-AzureRmLog -ResourceGroup deletedgroup -StartTime (Get-Date).AddDays(-14) -Caller someone@contoso.com
```

4. You can filter for failed operations.

```
Get-AzureRmLog -ResourceGroup ExampleGroup -Status Failed
```

5. You can focus on one error by looking at the status message for that entry.

```
 ((Get-AzureRmLog -Status Failed -ResourceGroup ExampleGroup -
DetailedOutput).Properties[1].Content["statusMessage"] | ConvertFrom-Json).error
```

Which returns:

```
 code           message
 ----           -------
 DnsRecordInUse DNS record dns.westus.cloudapp.azure.com is already used by another public IP.
```

Azure CLI

- To retrieve log entries, you run the **azure group log show** command.

```
azure group log show ExampleGroup --json
```

REST API

The REST operations for working with the activity log are part of the Insights REST API. To retrieve activity log events, see List the management events in a subscription.

Next steps

- Azure Activity logs can be used with Power BI to gain greater insights about the actions in your subscription. See View and analyze Azure Activity Logs in Power BI and more.
- To learn about setting security policies, see Azure Role-based Access Control.
- To learn about the commands for viewing deployment operations, see View deployment operations.
- To learn how to prevent deletions on a resource for all users, see Lock resources with Azure Resource Manager.

Archive Azure Diagnostic Logs

1/17/2017 • 4 min to read • Edit on GitHub

In this article, we show how you can use the Azure portal, PowerShell Cmdlets, CLI, or REST API to archive your Azure Diagnostic Logs in a storage account. This option is useful if you would like to retain your Diagnostic Logs with an optional retention policy for audit, static analysis, or backup. The storage account does not have to be in the same subscription as the resource emitting logs as long as the user who configures the setting has appropriate RBAC access to both subscriptions.

Prerequisites

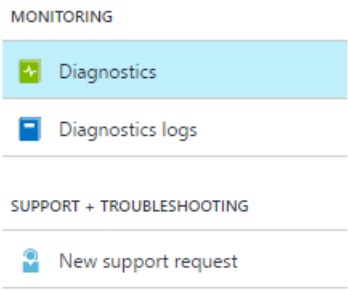
Before you begin, you need to create a storage account to which you can archive your Diagnostic Logs. We highly recommend that you do not use an existing storage account that has other, non-monitoring data stored in it so that you can better control access to monitoring data. However, if you are also archiving your Activity Log and diagnostic metrics to a storage account, it may make sense to use that storage account for your Diagnostic Logs as well to keep all monitoring data in a central location. The storage account you use must be a general purpose storage account, not a blob storage account.

Diagnostic Settings

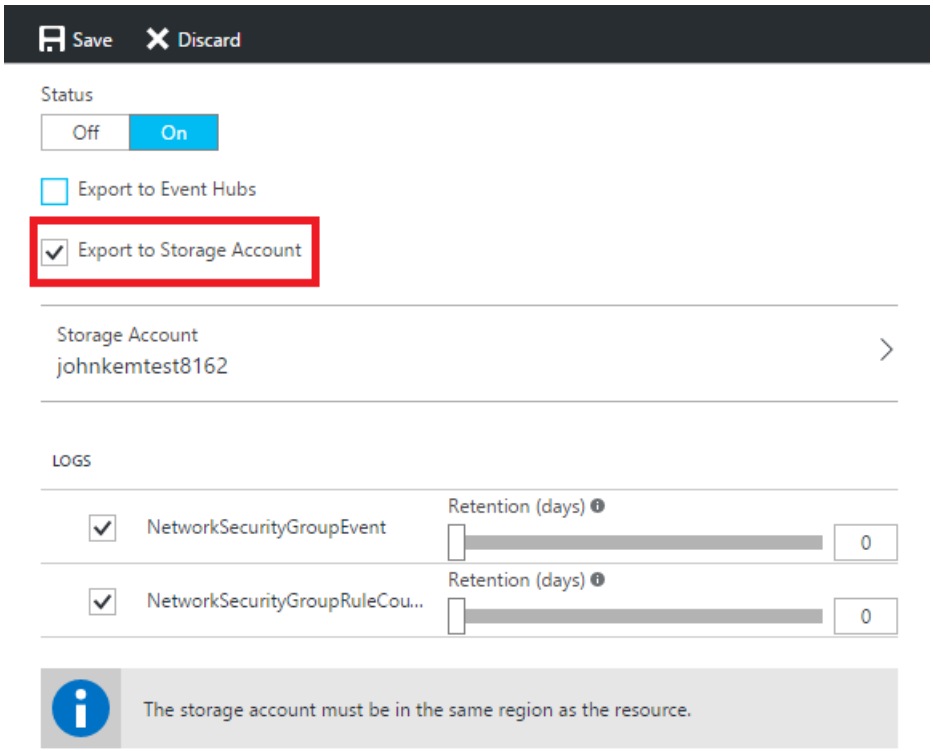
To archive your Diagnostic Logs using any of the methods below, you set a **Diagnostic Setting** for a particular resource. A diagnostic setting for a resource defines the categories of logs that are that are stored or streamed and the outputs—storage account and/or event hub. It also defines the retention policy (number of days to retain) for events of each log category stored in a storage account. If a retention policy is set to zero, events for that log category are stored indefinitely (that is to say, forever). A retention policy can otherwise be any number of days between 1 and 2147483647. You can read more about diagnostic settings here. Retention policies are applied per-day, so at the end of a day (UTC), logs from the day that is now beyond the retention policy will be deleted. For example, if you had a retention policy of one day, at the beginning of the day today the logs from the day before yesterday would be deleted

Archive Diagnostic Logs using the portal

1. In the portal, click into the resource blade for the resource on which you would like to enable archival of Diagnostic Logs.
2. In the **Monitoring** section of the resource settings menu, select **Diagnostics**.



3. Check the box for **Export to Storage Account**, then select a storage account. Optionally, set a number of days to retain these logs by using the **Retention (days)** sliders. A retention of zero days stores the logs indefinitely.



4. Click **Save**.

Diagnostic Logs are archived to that storage account as soon as new event data is generated.

Archive Diagnostic Logs via the PowerShell Cmdlets

```
Set-AzureRmDiagnosticSetting -ResourceId /subscriptions/s1id1234-5679-0123-4567-
890123456789/resourceGroups/testresourcegroup/providers/Microsoft.Network/networkSecurityGroups/testnsg -
StorageAccountId /subscriptions/s1id1234-5679-0123-4567-
890123456789/resourceGroups/myrg1/providers/Microsoft.Storage/storageAccounts/my_storage -Categories
networksecuritygroupevent,networksecuritygrouprulecounter -Enabled $true -RetentionEnabled $true -
RetentionInDays 90
```

PROPERTY	REQUIRED	DESCRIPTION
ResourceId	Yes	Resource ID of the resource on which you want to set a diagnostic setting.
StorageAccountId	No	Resource ID of the Storage Account to which Diagnostic Logs should be saved.
Categories	No	Comma-separated list of log categories to enable.
Enabled	Yes	Boolean indicating whether diagnostics are enabled or disabled on this resource.
RetentionEnabled	No	Boolean indicating if a retention policy are enabled on this resource.

PROPERTY	REQUIRED	DESCRIPTION
RetentionInDays	No	Number of days for which events should be retained between 1 and 2147483647. A value of zero stores the logs indefinitely.

Archive Diagnostic Logs via the Cross-Platform CLI

```
azure insights diagnostic set --resourceId /subscriptions/s1id1234-5679-0123-4567-890123456789/resourceGroups/testresourcegroup/providers/Microsoft.Network/networkSecurityGroups/testnsg --storageId /subscriptions/s1id1234-5679-0123-4567-890123456789/resourceGroups/myrg1/providers/Microsoft.Storage/storageAccounts/my_storage –categories networksecuritygroupevent,networksecuritygrouprulecounter --enabled true
```

PROPERTY	REQUIRED	DESCRIPTION
resourceId	Yes	Resource ID of the resource on which you want to set a diagnostic setting.
storageId	No	Resource ID of the Storage Account to which Diagnostic Logs should be saved.
categories	No	Comma-separated list of log categories to enable.
enabled	Yes	Boolean indicating whether diagnostics are enabled or disabled on this resource.

Archive Diagnostic Logs via the REST API

See this document for information on how you can set up a diagnostic setting using the Azure Monitor REST API.

Schema of Diagnostic Logs in the storage account

Once you have set up archival, a storage container is created in the storage account as soon as an event occurs in one of the log categories you have enabled. The blobs within the container follow the same format across Diagnostic Logs and the Activity Log. The structure of these blobs is:

```
insights-logs-{log category name}/resourceId=/SUBSCRIPTIONS/{subscription ID}/RESOURCEGROUPS/{resource group name}/PROVIDERS/{resource provider name}/{resource type}/{resource name}/y={four-digit numeric year}/m={two-digit numeric month}/d={two-digit numeric day}/h={two-digit 24-hour clock hour}/m=00/PT1H.json
```

Or, more simply,

```
insights-logs-{log category name}/resourceId=/{resource Id}/y={four-digit numeric year}/m={two-digit numeric month}/d={two-digit numeric day}/h={two-digit 24-hour clock hour}/m=00/PT1H.json
```

For example, a blob name might be:

```
insights-logs-networksecuritygrouprulecounter/resourceId=/SUBSCRIPTIONS/s1id1234-5679-0123-4567-
```

890123456789/RESOURCEGROUPS/TESTRESOURCEGROUP/PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUP/TESTNSG/y=2016/m=08/d=22/h=18/m=00/PT1H.json

Each PT1H.json blob contains a JSON blob of events that occurred within the hour specified in the blob URL (for example, h=12). During the present hour, events are appended to the PT1H.json file as they occur. The minute value (m=00) is always 00, since Diagnostic Log events are broken into individual blobs per hour.

Within the PT1H.json file, each event is stored in the "records" array, following this format:

```
{
    "records": [
        {
            "time": "2016-07-01T00:00:37.2040000Z",
            "systemId": "46cdbb41-cb9c-4f3d-a5b4-1d458d827ff1",
            "category": "NetworkSecurityGroupRuleCounter",
            "resourceId": "/SUBSCRIPTIONS/s1id1234-5679-0123-4567-
890123456789/RESOURCEGROUPS/TESTRESOURCEGROUP/PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/TESTNSG",
            "operationName": "NetworkSecurityGroupCounters",
            "properties": {
                "vnetResourceGuid": "{12345678-9012-3456-7890-123456789012}",
                "subnetPrefix": "10.3.0.0/24",
                "macAddress": "000123456789",
                "ruleName": "/subscriptions/ s1id1234-5679-0123-4567-
890123456789/resourceGroups/testresourcegroup/providers/Microsoft.Network/networkSecurityGroups/testnsg/security
Rules/default-allow-rdp",
                "direction": "In",
                "type": "allow",
                "matchedConnections": 1988
            }
        }
    ]
}
```

ELEMENT NAME	DESCRIPTION
time	Timestamp when the event was generated by the Azure service processing the request corresponding the event.
resourceId	Resource ID of the impacted resource.
operationName	Name of the operation.
category	Log category of the event.
properties	Set of `<Key, Value>` pairs (i.e. Dictionary) describing the details of the event.

> **NOTE**
> The properties and usage of those properties can vary depending on the resource.

Next Steps

- Download blobs for analysis
- Stream Diagnostic Logs to Event Hubs
- Read more about Diagnostic Logs

Stream Azure Diagnostic Logs to Event Hubs

1/17/2017 • 4 min to read • Edit on GitHub

Azure Diagnostic Logs can be streamed in near real time to any application using the built-in "Export to Event Hubs" option in the Portal, or by enabling the Service Bus Rule Id in a Diagnostic Setting via the Azure PowerShell Cmdlets or Azure CLI.

What you can do with Diagnostics Logs and Event Hubs

Here are just a few ways you might use the streaming capability for Diagnostic Logs:

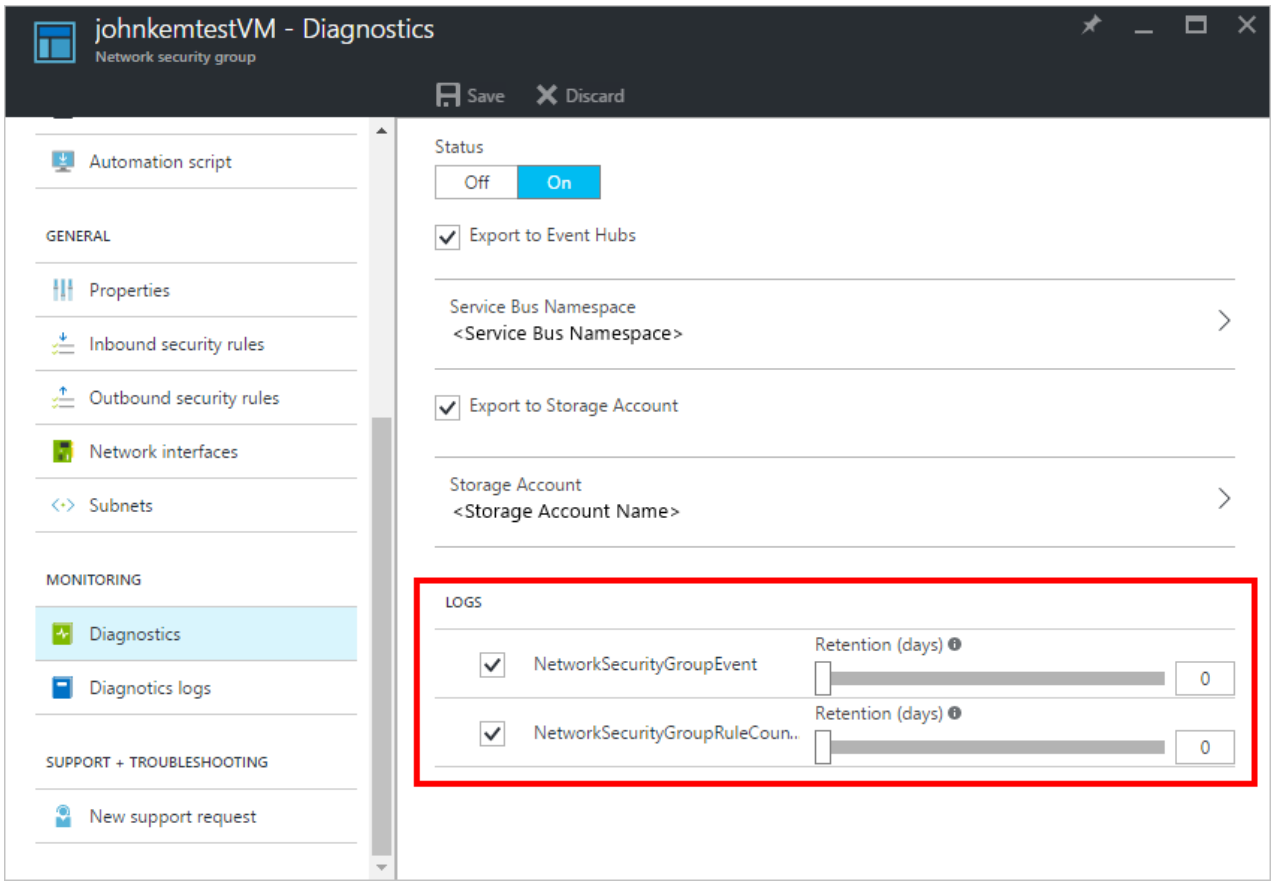
- **Stream logs to 3rd party logging and telemetry systems** – Over time, Event Hubs streaming will become the mechanism to pipe your Diagnostic Logs into third party SIEMs and log analytics solutions.
- **View service health by streaming "hot path" data to PowerBI** – Using Event Hubs, Stream Analytics, and PowerBI, you can easily transform your diagnostics data into near real-time insights on your Azure services. This documentation article gives a great overview of how to set up an Event Hubs, process data with Stream Analytics, and use PowerBI as an output. Here's a few tips for getting set up with Diagnostic Logs:
    - The Event Hubs for a category of Diagnostic Logs is created automatically when you check the option in the portal or enable it through PowerShell, so you want to select the Event Hubs in the Service Bus namespace with the name that starts with "insights-"
    - Here's a sample Stream Analytics query you can use to simply parse all the log data in to a PowerBI table:

```
SELECT
records.ArrayValue.[Properties you want to track]
INTO
[OutputSourceName – the PowerBI source]
FROM
[InputSourceName] AS e
CROSS APPLY GetArrayElements(e.records) AS records
```

- **Build a custom telemetry and logging platform** – If you already have a custom-built telemetry platform or are just thinking about building one, the highly scalable publish-subscribe nature of Event Hubs allows you to flexibly ingest diagnostic logs. See Dan Rosanova's guide to using Event Hubs in a global scale telemetry platform here.

Enable streaming of Diagnostic Logs

You can enable streaming of Diagnostic Logs programmatically, via the portal, or using the Azure Monitor REST API. Either way, you pick a Service Bus Namespace and an Event Hubs is created in the namespace for each log category you enable. A Diagnostic **Log Category** is a type of log that a resource may collect. You can select which log categories you'd like to collect for a particular resource in the Azure Portal under the Diagnostics blade.



WARNING

Enabling and streaming diagnostic logs from Compute resources (for example, VMs or Service Fabric) requires a different set of steps.

The service bus or event hub namespace does not have to be in the same subscription as the resource emitting logs as long as the user who configures the setting has appropriate RBAC access to both subscriptions.

Via PowerShell Cmdlets

To enable streaming via the Azure PowerShell Cmdlets, you can use the `Set-AzureRmDiagnosticSetting` cmdlet with these parameters:

```
Set-AzureRmDiagnosticSetting -ResourceId [your resource Id] -ServiceBusRuleId [your service bus rule id] -
Enabled $true
```

The Service Bus Rule ID is a string with this format: `{service bus resource ID}/authorizationrules/{key name}`, for example,

```
/subscriptions/{subscription ID}/resourceGroups/Default-ServiceBus-
WestUS/providers/Microsoft.ServiceBus/namespaces/{service bus
namespace}/authorizationrules/RootManageSharedAccessKey
```

.

Via Azure CLI

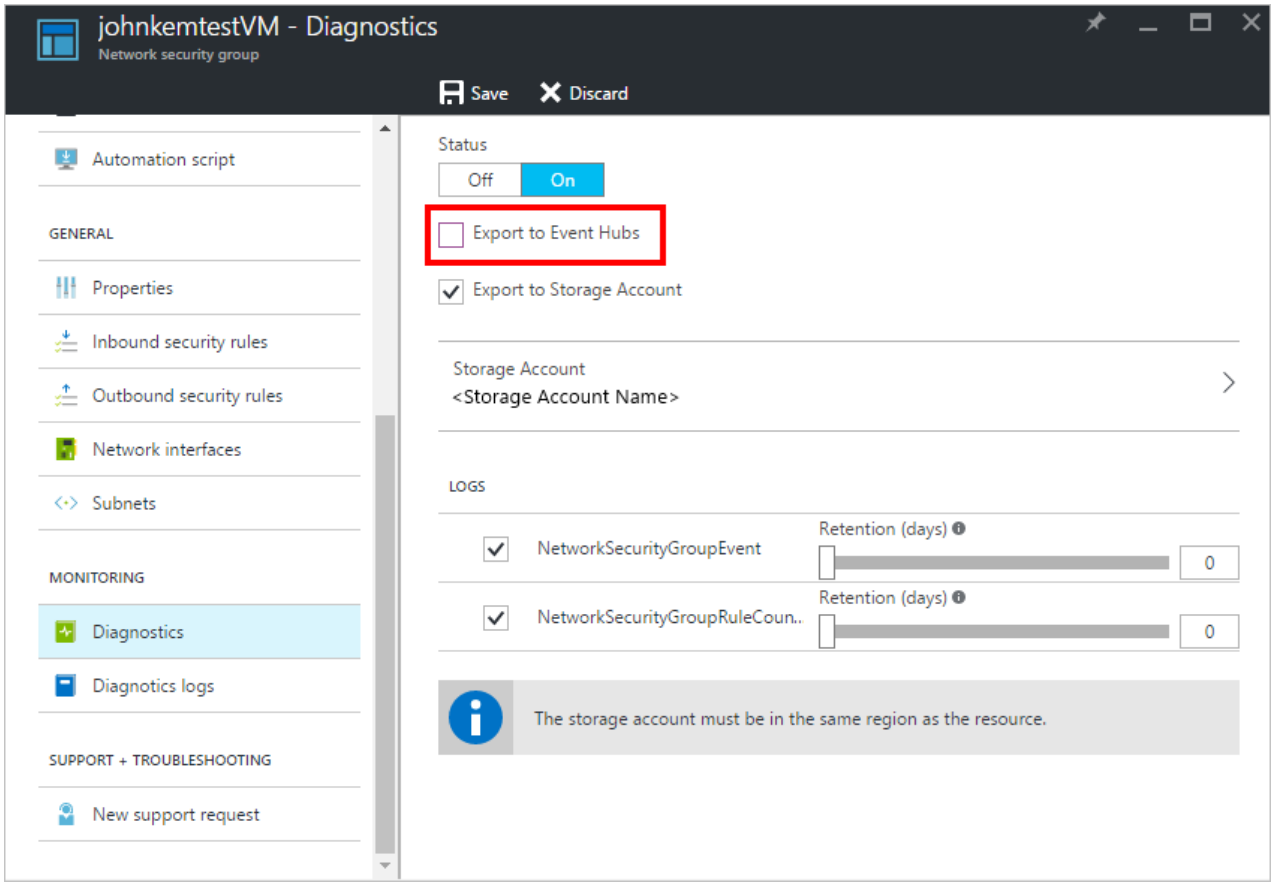
To enable streaming via the Azure CLI, you can use the `insights diagnostic set` command like this:

```
azure insights diagnostic set --resourceId <resourceId> --serviceBusRuleId <serviceBusRuleId> --enabled true
```

Use the same format for Service Bus Rule ID as explained for the PowerShell Cmdlet.

Via Azure Portal

To enable streaming via the Azure Portal, navigate to the diagnostics settings of a resource and select 'Export to Event Hub.'



To configure it, select an existing Service Bus Namespace. The namespace selected will be where the Event Hubs is created (if this is your first time streaming diagnostic logs) or streamed to (if there are already resources that are streaming that log category to this namespace), and the policy defines the permissions that the streaming mechanism has. Today, streaming to an Event Hubs requires Manage, Send, and Listen permissions. You can create or modify Service Bus Namespace shared access policies in the classic portal under the "Configure" tab for your Service Bus Namespace. To update one of these Diagnostic Settings, the client must have the ListKey permission on the Service Bus Authorization Rule.

How do I consume the log data from Event Hubs?

Here is sample output data from the Event Hubs:

```
{
    "records": [
        {
            "time": "2016-07-15T18:00:22.6235064Z",
            "workflowId": "/SUBSCRIPTIONS/DF602C9C-7AA0-407D-A6FB-
EB20C8BD1192/RESOURCEGROUPS/JOHNKEMTEST/PROVIDERS/MICROSOFT.LOGIC/WORKFLOWS/JOHNKEMTESTLA",
            "resourceId": "/SUBSCRIPTIONS/DF602C9C-7AA0-407D-A6FB-
EB20C8BD1192/RESOURCEGROUPS/JOHNKEMTEST/PROVIDERS/MICROSOFT.LOGIC/WORKFLOWS/JOHNKEMTESTLA/RUNS/08587330013509921
957/ACTIONS/SEND_EMAIL",
            "category": "WorkflowRuntime",
            "level": "Error",
            "operationName": "Microsoft.Logic/workflows/workflowActionCompleted",
            "properties": {
                "$schema": "2016-04-01-preview",
                "startTime": "2016-07-15T17:58:55.048482Z",
                "endTime": "2016-07-15T18:00:22.4109204Z",
                "status": "Failed",
                "code": "BadGateway",
                "resource": {
                    "subscriptionId": "df602c9c-7aa0-407d-a6fb-eb20c8bd1192",
                    "resourceGroupName": "JohnKemTest",
                    "workflowId": "243aac67fe904cf195d4a28297803785",
                    "workflowName": "JohnKemTestLA",
                    "runId": "08587330013509921957",
                    "location": "westus",
                    "actionName": "Send_email"
                },
                "correlation": {
                    "actionTrackingId": "29a9862f-969b-4c70-90c4-dfbdc814e413",
                    "clientTrackingId": "08587330013509921958"
                }
            }
        },
        {
            "time": "2016-07-15T18:01:15.7532989Z",
            "workflowId": "/SUBSCRIPTIONS/DF602C9C-7AA0-407D-A6FB-
EB20C8BD1192/RESOURCEGROUPS/JOHNKEMTEST/PROVIDERS/MICROSOFT.LOGIC/WORKFLOWS/JOHNKEMTESTLA",
            "resourceId": "/SUBSCRIPTIONS/DF602C9C-7AA0-407D-A6FB-
EB20C8BD1192/RESOURCEGROUPS/JOHNKEMTEST/PROVIDERS/MICROSOFT.LOGIC/WORKFLOWS/JOHNKEMTESTLA/RUNS/08587330012106702
630/ACTIONS/SEND_EMAIL",
            "category": "WorkflowRuntime",
            "level": "Information",
            "operationName": "Microsoft.Logic/workflows/workflowActionStarted",
            "properties": {
                "$schema": "2016-04-01-preview",
                "startTime": "2016-07-15T18:01:15.5828115Z",
                "status": "Running",
                "resource": {
                    "subscriptionId": "df602c9c-7aa0-407d-a6fb-eb20c8bd1192",
                    "resourceGroupName": "JohnKemTest",
                    "workflowId": "243aac67fe904cf195d4a28297803785",
                    "workflowName": "JohnKemTestLA",
                    "runId": "08587330012106702630",
                    "location": "westus",
                    "actionName": "Send_email"
                },
                "correlation": {
                    "actionTrackingId": "042fb72c-7bd4-439e-89eb-3cf4409d429e",
                    "clientTrackingId": "08587330012106702632"
                }
            }
        }
    ]
}
```

ELEMENT NAME	DESCRIPTION
records	An array of all log events in this payload.
time	Time at which the event occurred.
category	Log category for this event.
resourceId	Resource ID of the resource that generated this event.
operationName	Name of the operation.
level	Optional. Indicates the log event level.
properties	Properties of the event.

You can view a list of all resource providers that support streaming to Event Hub here.

Stream data from Compute resources

You can also stream diagnostic logs from Compute resources using the Windows Azure Diagnositcs agent. See this article for how to set that up.

Next Steps

- Read more about Azure Diagnostic Logs
- Get started with Event Hubs

Automatically enable Diagnostic Settings at resource creation using a Resource Manager template

1/17/2017 • 3 min to read • Edit on GitHub

In this article we show how you can use an Azure Resource Manager template to configure Diagnostic Settings on a resource when it is created. This enables you to automatically start streaming your Diagnostic Logs and metrics to Event Hubs, archiving them in a Storage Account, or sending them to Log Analytics when a resource is created.

The method for enabling Diagnostic Logs using a Resource Manager template depends on the resource type.

- **Non-Compute** resources (for example, Network Security Groups, Logic Apps, Automation) use Diagnostic Settings described in this article.
- **Compute** (WAD/LAD-based) resources use the WAD/LAD configuration file described in this article.

In this article we describe how to configure diagnostics using either method.

The basic steps are as follows:

1. Create a template as a JSON file that describes how to create the resource and enable diagnostics.
2. Deploy the template using any deployment method.

Below we give an example of the template JSON file you need to generate for non-Compute and Compute resources.

Non-Compute resource template

For non-Compute resources, you will need to do two things:

1. Add parameters to the parameters blob for the storage account name, service bus rule ID, and/or OMS Log Analytics workspace ID (enabling archival of Diagnostic Logs in a storage account, streaming of logs to Event Hubs, and/or sending logs to Log Analytics).

```
 "storageAccountName": {
   "type": "string",
   "metadata": {
     "description": "Name of the Storage Account in which Diagnostic Logs should be saved."
   }
 },
 "serviceBusRuleId": {
   "type": "string",
   "metadata": {
     "description": "Service Bus Rule Id for the Service Bus Namespace in which the Event Hub should be
created or streamed to."
   }
 },
 "workspaceId":{
   "type": "string",
   "metadata": {
     "description": "Log Analytics workspace ID for the Log Analytics workspace to which logs will be
sent."
   }
 }
```

2. In the resources array of the resource for which you want to enable Diagnostic Logs, add a resource of type `[resource namespace]/providers/diagnosticSettings`.

```
  "resources": [
    {
      "type": "providers/diagnosticSettings",
      "name": "Microsoft.Insights/service",
      "dependsOn": [
        "[/*resource Id for which Diagnostic Logs will be enabled>*/]"
      ],
      "apiVersion": "2015-07-01",
      "properties": {
        "storageAccountId": "[resourceId('Microsoft.Storage/storageAccounts',
 parameters('storageAccountName'))]",
        "serviceBusRuleId": "[parameters('serviceBusRuleId')]",
        "workspaceId": "[parameters('workspaceId')]",
        "logs": [
          {
            "category": "/* log category name */",
            "enabled": true,
            "retentionPolicy": {
              "days": 0,
              "enabled": false
            }
          }
        ],
        "metrics": [
          {
            "timeGrain": "PT1M",
            "enabled": true,
            "retentionPolicy": {
              "enabled": false,
              "days": 0
            }
          }
        ]
      }
    }
  ]
```

The properties blob for the Diagnostic Setting follows the format described in this article. Adding the `metrics` property will enable you to also send resource metrics to these same outputs.

Here is a full example that creates a Network Security Group and turns on streaming to Event Hubs and storage in a storage account.

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "nsgName": {
      "type": "string",
      "metadata": {
        "description": "Name of the NSG that will be created."
      }
    },
    "storageAccountName": {
      "type": "string",
      "metadata": {
        "description": "Name of the Storage Account in which Diagnostic Logs should be saved."
      }
    },
    "serviceBusRuleId": {
      "type": "string",
      "metadata": {
        "description": "Service Bus Rule Id for the Service Bus Namespace in which the Event Hub should be
created or streamed to."
      }
```

```
    },
    "workspaceId": {
      "type": "string",
      "metadata": {
        "description": "Log Analytics workspace ID for the Log Analytics workspace to which logs will be sent."
      }
    }
  },
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.Network/networkSecurityGroups",
      "name": "[parameters('nsgName')]",
      "apiVersion": "2016-03-30",
      "location": "westus",
      "properties": {
        "securityRules": []
      },
      "resources": [
        {
          "type": "providers/diagnosticSettings",
          "name": "Microsoft.Insights/service",
          "dependsOn": [
            "[resourceId('Microsoft.Network/networkSecurityGroups', parameters('nsgName'))]"
          ],
          "apiVersion": "2015-07-01",
          "properties": {
            "storageAccountId": "[resourceId('Microsoft.Storage/storageAccounts',
parameters('storageAccountName'))]",
            "serviceBusRuleId": "[parameters('serviceBusRuleId')]",
            "workspaceId": "[parameters('workspaceId')]",
            "logs": [
              {
                "category": "NetworkSecurityGroupEvent",
                "enabled": true,
                "retentionPolicy": {
                  "days": 0,
                  "enabled": false
                }
              },
              {
                "category": "NetworkSecurityGroupRuleCounter",
                "enabled": true,
                "retentionPolicy": {
                  "days": 0,
                  "enabled": false
                }
              }
            ],
            "metrics": [
              {
                "timeGrain": "PT1M",
                "enabled": true,
                "retentionPolicy": {
                  "enabled": false,
                  "days": 0
                }
              }
            ]
          }
        }
      ],
      "dependsOn": []
    }
  ]
}
```

Compute resource template

To enable diagnostics on a Compute resource, for example a Virtual Machine or Service Fabric cluster, you need to:

1. Add the Azure Diagnostics extension to the VM resource definition.
2. Specify a storage account and/or event hub as a parameter.
3. Add the contents of your WADCfg XML file into the XMLCfg property, escaping all XML characters properly.

> **WARNING**
>
> This last step can be tricky to get right. See this article for an example that splits the Diagnostics Configuration Schema into variables that are escaped and formatted correctly.

The entire process, including samples, is described in this document.

Next Steps

- Read more about Azure Diagnostic Logs
- Stream Azure Diagnostic Logs to Event Hubs

Azure Monitoring REST API Walkthrough

1/17/2017 • 6 min to read • Edit on GitHub

This article shows you how to perform authentication so your code can use the Microsoft Azure Monitor REST API Reference.

The Azure Monitor API makes it possible to programmatically retrieve the available default metric definitions (the type of metric such as CPU Time, Requests, etc.), granularity, and metric values. Once retrieved, the data can be saved in a separate data store such as Azure SQL Database, DocumentDB, or Azure Data Lake. From there additional analysis can be performed as needed.

Besides working with various metric data points, as this article demonstrates, the Monitor API makes it possible to list alert rules, view activity logs, and much more. For a full list of available operations, see the Microsoft Azure Monitor REST API Reference.

Authenticating Azure Monitor Requests

The first step is to authenticate the request.

All the tasks executed against the Azure Monitor API use the Azure Resource Manager authentication model. Therefore, all requests must be authenticated with Azure Active Directory (Azure AD). One approach to authenticate the client application is to create an Azure AD service principal and retrieve the authentication (JWT) token. The following sample script demonstrates creating an Azure AD service principal via PowerShell. For a more detailed walk-through, refer to the documentation on using Azure PowerShell to create a service principal to access resources. It is also possible to create a service principle via the Azure portal.

```
$subscriptionId = "{azure-subscription-id}"
$resourceGroupName = "{resource-group-name}"
$location = "centralus"

# Authenticate to a specific Azure subscription.
Login-AzureRmAccount -SubscriptionId $subscriptionId

# Password for the service principal
$pwd = "{service-principal-password}"

# Create a new Azure AD application
$azureAdApplication = New-AzureRmADApplication `
                        -DisplayName "My Azure Monitor" `
                        -HomePage "https://localhost/azure-monitor" `
                        -IdentifierUris "https://localhost/azure-monitor" `
                        -Password $pwd

# Create a new service principal associated with the designated application
New-AzureRmADServicePrincipal -ApplicationId $azureAdApplication.ApplicationId

# Assign Reader role to the newly created service principal
New-AzureRmRoleAssignment -RoleDefinitionName Reader `
                           -ServicePrincipalName $azureAdApplication.ApplicationId.Guid
```

To query the Azure Monitor API, the client application should use the previously created service principal to authenticate. The following example PowerShell script shows one approach, using the Active Directory Authentication Library (ADAL) to help get the JWT authentication token. The JWT token is passed as part of an HTTP Authorization parameter in requests to the Azure Monitor REST API.

```
$azureAdApplication = Get-AzureRmADApplication -IdentifierUri "https://localhost/azure-monitor"

$subscription = Get-AzureRmSubscription -SubscriptionId $subscriptionId

$clientId = $azureAdApplication.ApplicationId.Guid
$tenantId = $subscription.TenantId
$authUrl = "https://login.windows.net/${tenantId}"

$AuthContext = [Microsoft.IdentityModel.Clients.ActiveDirectory.AuthenticationContext]$authUrl
$cred = New-Object -TypeName Microsoft.IdentityModel.Clients.ActiveDirectory.ClientCredential -ArgumentList
($clientId, $pwd)

$result = $AuthContext.AcquireToken("https://management.core.windows.net/", $cred)

# Build an array of HTTP header values
$authHeader = @{
'Content-Type'='application/json'
'Accept'='application/json'
'Authorization'=$result.CreateAuthorizationHeader()
}
```

Once the authentication setup step is complete, queries can then be executed against the Azure Monitor REST API. There are two helpful queries:

1. List the metric definitions for a resource
2. Retrieve the metric values

Retrieve Metric Definitions

> **NOTE**
>
> To retrieve metric definitions using the Azure Monitor REST API, use "2016-03-01" as the API version.

```
$apiVersion = "2016-03-01"
$request =
"https://management.azure.com/subscriptions/${subscriptionId}/resourceGroups/${resourceGroupName}/providers/${re
sourceProviderNamespace}/${resourceType}/${resourceName}/providers/microsoft.insights/metricDefinitions?api-
version=${apiVersion}"

Invoke-RestMethod -Uri $request `
                  -Headers $authHeader `
                  -Method Get `
                  -Verbose
```

For an Azure Logic App, the metric definitions would appear similar to the following screenshot:

```
{
    "id": "/subscriptions/[redacted]/resourceGroups/contosoweb001/providers/Microsoft.Logic/workflows/ContosoTweetsLogic/providers/microsoft.insights/metricdefinitions",
    "value": [
        {
            "resourceUri": "/subscriptions/[redacted]/resourceGroups/contosoweb001/providers/Microsoft.Logic/workflows/ContosoTweetsLogic",
            "resourceId": "/subscriptions/[redacted]/resourceGroups/contosoweb001/providers/Microsoft.Logic/workflows/ContosoTweetsLogic",
            "name": {
                "value": "RunsStarted",
                "localizedValue": "RunsStarted"
            },
            "startTime": "0 0 0 1 -0 1 -0 1T00: 0 0: 0 0Z",
            "endTime": "0 0 0 1 -0 1 -0 1T00: 0 0: 0 0Z",
            "unit": "Count",
            "primaryAggregationType": "Total",
            "ResourceUri": "/subscriptions/[redacted]/resourceGroups/contosoweb001/providers/Microsoft.Logic/workflows/ContosoTweetsLogic",
            "ResourceId": "/subscriptions/[redacted]/resourceGroups/contosoweb001/providers/Microsoft.Logic/workflows/ContosoTweetsLogic",
            "metricAvailabilities": [
                {
                    "timeGrain": "PT1M",
                    "retention": "P30D"
                },
                {
                    "timeGrain": "PT1H",
                    "retention": "P30D"
                }
            ],
            "id": "/subscriptions/[redacted]/resourceGroups/contosoweb001/providers/Microsoft.Logic/workflows/ContosoTweetsLogic/providers/microsoft.insights/metricdefinitions/RunsStarted"
        },
        {
            "resourceUri": "/subscriptions/[redacted]/resourceGroups/contosoweb001/providers/Microsoft.Logic/workflows/ContosoTweetsLogic",
            "resourceId": "/subscriptions/[redacted]/resourceGroups/contosoweb001/providers/Microsoft.Logic/workflows/ContosoTweetsLogic",
            "name": {
                "value": "RunsCompleted",
                "localizedValue": "RunsCompleted"
            },
            "startTime": "0 0 0 1 -0 1 -0 1T00: 0 0: 0 0Z",
            "endTime": "0001-01-01T00:00:00Z",
            "unit": "Count",
            "primaryAggregationType": "Total",
            "ResourceUri": "/subscriptions/[redacted]/resourceGroups/contosoweb001/providers/Microsoft.Logic/workflows/ContosoTweetsLogic",
            "ResourceId": "/subscriptions/[redacted]/resourceGroups/contosoweb001/providers/Microsoft.Logic/workflows/ContosoTweetsLogic",
            "metricAvailabilities": [
                {
                    "timeGrain": "PT1M",
                    "retention": "P30D"
                },
                {
                    "timeGrain": "PT1H",
                    "retention": "P30D"
```


For more information, see the List the metric definitions for a resource in Azure Monitor REST API documentation.

Retrieve Metric Values

Once the available metric definitions are known, it is then possible to retrieve the related metric values. Use the metric's name 'value' (not the 'localizedValue') for any filtering requests (for example, retrieve the 'CpuTime' and 'Requests' metric data points). If no filters are specified, the default metric is returned.

> **NOTE**
>
> To retrieve metric values using the Azure Monitor REST API, use "2016-06-01" as the API version.

Method: GET

Request URI: https://management.azure.com/subscriptions/*{subscription-id}*/resourceGroups/*{resource-group-name}*/providers/*{resource-provider-namespace}*/*{resource-type}*/*{resource-name}*/providers/microsoft.insights/metrics?$filter=*{filter}*&api-version=*{apiVersion}*

For example, to retrieve the RunsSucceeded metric data points for the given time range and for a time grain of 1 hour, the request would be as follows:

```
$apiVersion = "2016-06-01"
$filter = "(name.value eq 'RunsSucceeded') and aggregationType eq 'Total' and startTime eq 2016-09-23 and
endTime eq 2016-09-24 and timeGrain eq duration'PT1H'"
$request =
"https://management.azure.com/subscriptions/${subscriptionId}/resourceGroups/${resourceGroupName}/providers/${re
sourceProviderNamespace}/${resourceType}/${resourceName}/providers/microsoft.insights/metrics?
`$filter=${filter}&api-version=${apiVersion}"
(Invoke-RestMethod -Uri $request `
                   -Headers $authHeader `
                   -Method Get `
                   -Verbose).Value | ConvertTo-Json
```

The result would appear similar to the example following screenshot:

```
{
  "data": [
    {"timeStamp": "2016-09-23T00:00:00Z"},
    {"timeStamp": "2016-09-23T01:00:00Z"},
    {"timeStamp": "2016-09-23T02:00:00Z"},
    {"timeStamp": "2016-09-23T03:00:00Z"},
    {"timeStamp": "2016-09-23T04:00:00Z"},
    {"timeStamp": "2016-09-23T05:00:00Z"},
    {"timeStamp": "2016-09-23T06:00:00Z"},
    {"timeStamp": "2016-09-23T07:00:00Z"},
    {"timeStamp": "2016-09-23T08:00:00Z"},
    {"timeStamp": "2016-09-23T09:00:00Z"},
    {"timeStamp": "2016-09-23T10:00:00Z"},
    {"timeStamp": "2016-09-23T11:00:00Z"},
    {"timeStamp": "2016-09-23T12:00:00Z"},
    {"timeStamp": "2016-09-23T13:00:00Z",
      "total": 8.0
    },
    {"timeStamp": "2016-09-23T14:00:00Z",
      "total": 52.0
    },
    {"timeStamp": "2016-09-23T15:00:00Z",
      "total": 22.0
    },
    {"timeStamp": "2016-09-23T16:00:00Z"},
    {"timeStamp": "2016-09-23T17:00:00Z"},
    {"timeStamp": "2016-09-23T18:00:00Z"},
    {"timeStamp": "2016-09-23T19:00:00Z"},
    {"timeStamp": "2016-09-23T20:00:00Z"},
    {"timeStamp": "2016-09-23T21:00:00Z"},
    {"timeStamp": "2016-09-23T22:00:00Z"},
    {"timeStamp": "2016-09-23T23:00:00Z"}
  ],
  "name": {
    "value": "RunsSucceeded",
    "localizedValue": "Runs Succeeded"
  },
  "unit": "0"
}
```

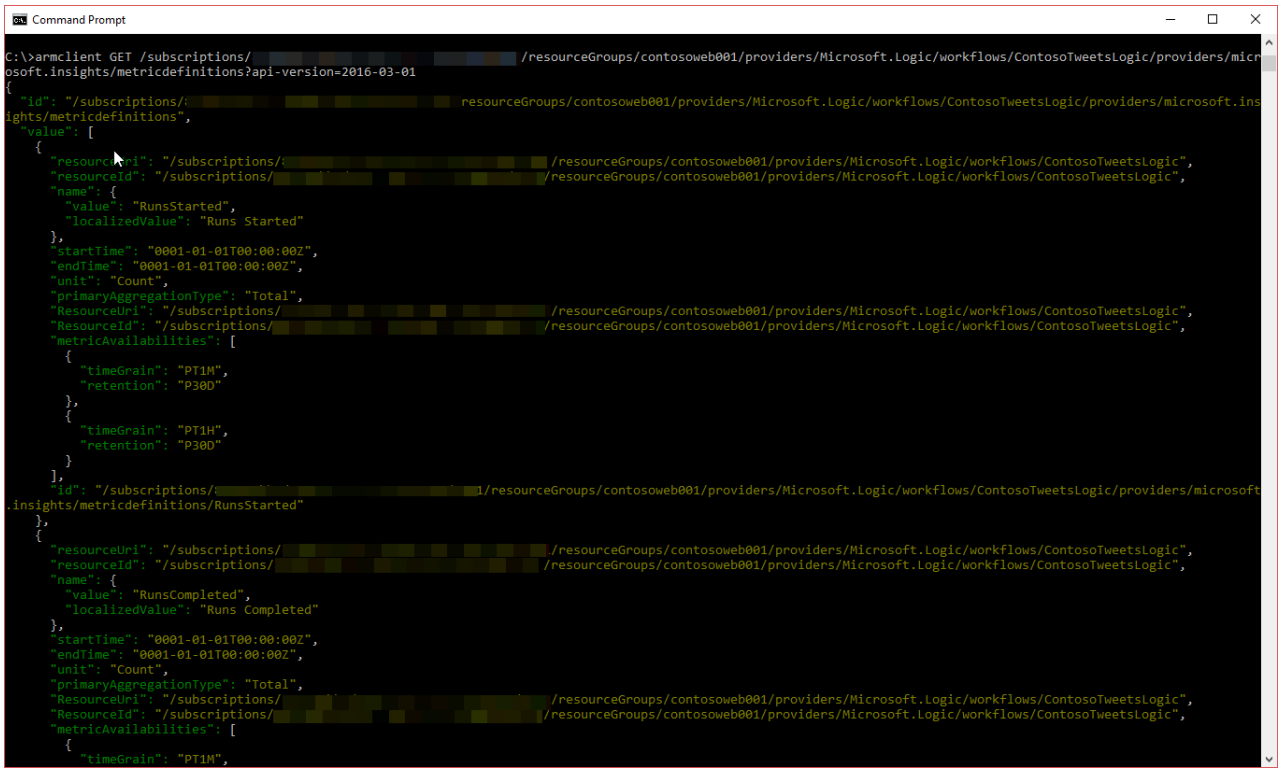

To retrieve multiple data or aggregation points, add the metric definition names and aggregation types to the filter, as seen in the following example:

```
$apiVersion = "2016-06-01"
$filter = "(name.value eq 'ActionsCompleted' or name.value eq 'RunsSucceeded') and (aggregationType eq 'Total'
or aggregationType eq 'Average') and startTime eq 2016-09-23T13:30:00Z and endTime eq 2016-09-23T14:30:00Z and
timeGrain eq duration'PT1M'"
$request =
"https://management.azure.com/subscriptions/${subscriptionId}/resourceGroups/${resourceGroupName}/providers/${re
sourceProviderNamespace}/${resourceType}/${resourceName}/providers/microsoft.insights/metrics?
`$filter=${filter}&api-version=${apiVersion}"
(Invoke-RestMethod -Uri $request `
                   -Headers $authHeader `
                   -Method Get `
                   -Verbose).Value | ConvertTo-Json
```

Use ARMClient

An alternative to using PowerShell (as shown above), is to use ARMClient on your Windows machine. ARMClient handles the Azure AD authentication (and resulting JWT token) automatically. The following steps outline use of ARMClient for retrieving metric data:

1. Install Chocolatey and ARMClient.
2. In a terminal window, type *armclient.exe login*. This prompts you to log in to Azure.
3. Type *armclient GET [your_resource_id]/providers/microsoft.insights/metricdefinitions?api-version=2016-03-01*
4. Type *armclient GET [your_resource_id]/providers/microsoft.insights/metrics?api-version=2016-06-01*



Retrieve the Resource ID

Using the REST API can really help to understand the available metric definitions, granularity, and related values. That information is helpful when using the Azure Management Library.

For the preceding code, the resource ID to use is the full path to the desired Azure resource. For example, to query against an Azure Web App, the resource ID would be:

/subscriptions/{subscription-id}/resourceGroups/{resource-group-name}/providers/Microsoft.Web/sites/{site-name}/

The following list contains a few examples of resource ID formats for various Azure resources:

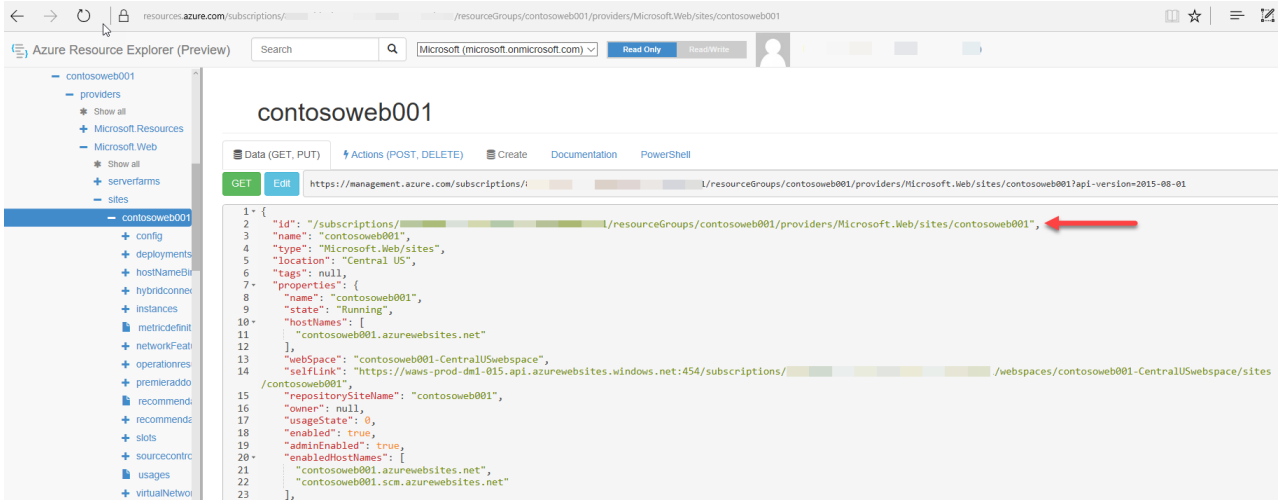
- **IoT Hub** - /subscriptions/*{subscription-id}*/resourceGroups/*{resource-group-name}*/providers/Microsoft.Devices/IotHubs/*{iot-hub-name}*
- **Elastic SQL Pool** - /subscriptions/*{subscription-id}*/resourceGroups/*{resource-group-name}*/providers/Microsoft.Sql/servers/*{pool-db}*/elasticpools/*{sql-pool-name}*
- **SQL Database (v12)** - /subscriptions/*{subscription-id}*/resourceGroups/*{resource-group-name}*/providers/Microsoft.Sql/servers/*{server-name}*/databases/*{database-name}*
- **Service Bus** - /subscriptions/*{subscription-id}*/resourceGroups/*{resource-group-name}*/providers/Microsoft.ServiceBus/*{namespace}*/*{servicebus-name}*
- **VM Scale Sets** - /subscriptions/*{subscription-id}*/resourceGroups/*{resource-group-name}*/providers/Microsoft.Compute/virtualMachineScaleSets/*{vm-name}*
- **VMs** - /subscriptions/*{subscription-id}*/resourceGroups/*{resource-group-name}*/providers/Microsoft.Compute/virtualMachines/*{vm-name}*
- **Event Hubs** - /subscriptions/*{subscription-id}*/resourceGroups/*{resource-group-name}*/providers/Microsoft.EventHub/namespaces/*{eventhub-namespace}*

There are alternative approaches to retrieving the resource ID, including using Azure Resource Explorer, viewing the desired resource in the Azure portal, and via PowerShell or the Azure CLI.

Azure Resource Explorer

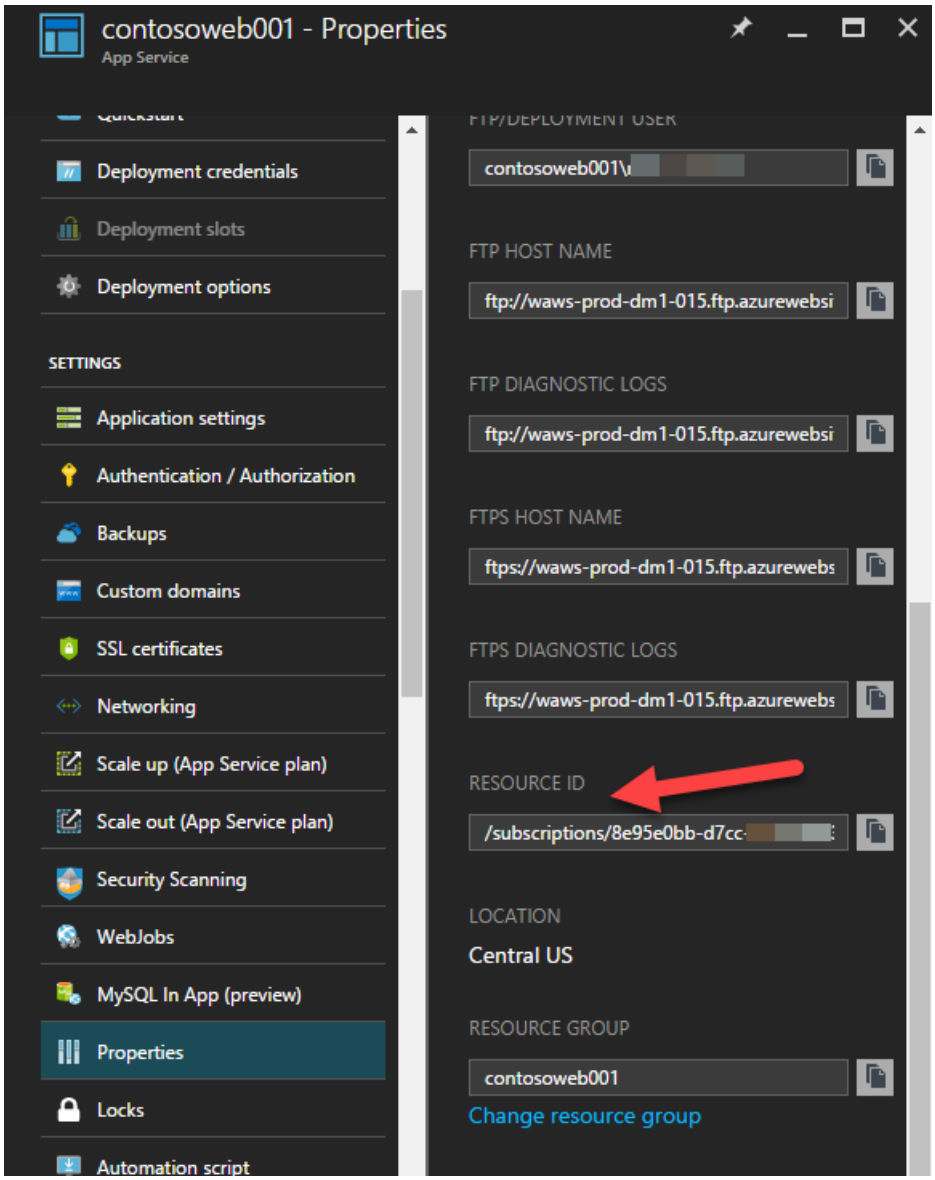
To find the resource ID for a desired resource, one helpful approach is to use the Azure Resource Explorer tool.

Navigate to the desired resource and then look at the ID shown, as in the following screenshot:



Azure portal

The resource ID can also be obtained from the Azure portal. To do so, navigate to the desired resource and then select Properties. The Resource ID is displayed in the Properties blade, as seen in the following screenshot:



Azure PowerShell

The resource ID can be retrieved using Azure PowerShell cmdlets as well. For example, to obtain the resource ID for an Azure Web App, execute the Get-AzureRmWebApp cmdlet, as in the following screenshot:

```
PS C:\> Get-AzureRmWebApp -ResourceGroupName $resourceGroupName -Name $webAppName

SiteName                  : contosoweb001
State                     : Running
HostNames                 : {contosoweb001.azurewebsites.net}
RepositorySiteName        : contosoweb001
UsageState                : Normal
Enabled                   : True
EnabledHostNames          : {contosoweb001.azurewebsites.net, contosoweb001.scm.azurewebsites.net}
AvailabilityState         : Normal
HostNameSslStates         : {contosoweb001.azurewebsites.net, contosoweb001.scm.azurewebsites.net}
ServerFarmId              : /subscriptions/[redacted]/resourceGroups/contosoweb001/providers/Microsoft.Web/serverfarms/contosowebplan
LastModifiedTimeUtc       : 9/20/2016 1:35:40 AM
SiteConfig                : Microsoft.Azure.Management.WebSites.Models.SiteConfig
TrafficManagerHostNames   :
PremiumAppDeployed        :
ScmSiteAlsoStopped        : False
TargetSwapSlot            :
HostingEnvironmentProfile :
MicroService              : WebSites
GatewaySiteName           :
ClientAffinityEnabled     : True
ClientCertEnabled         : False
HostNamesDisabled         : False
OutboundIpAddresses       : 40.86.95.108,40.86.95.228,40.86.92.253,40.86.95.159
ContainerSize             : 0
MaxNumberOfWorkers        :
CloningInfo               :
ResourceGroup             : contosoweb001
IsDefaultContainer        :
DefaultHostName           : contosoweb001.azurewebsites.net
Id                        : /subscriptions/[redacted]/resourceGroups/contosoweb001/providers/Microsoft.Web/sites/contosoweb001
Name                      : contosoweb001
Location                  : Central US
Type                      : Microsoft.Web/sites
Tags                      :
```

Azure CLI

To retrieve the resource ID using the Azure CLI, execute the 'azure webapp show' command, specifying the '--json' option, as shown in the following screenshot:

```
Command Prompt

C:\>azure webapp show --resource-group contosoweb001 --name contosoweb001 --json
{
  "id": "/subscriptions/[redacted]/resourceGroups/contosoweb001/providers/Microsoft.Web/sites/contosoweb001",
  "name": "contosoweb001",
  "location": "Central US",
  "type": "Microsoft.Web/sites",
  "siteName": "contosoweb001",
  "state": "Running",
  "hostNames": [
    "contosoweb001.azurewebsites.net"
  ],
```

Retrieve Activity Log Data

In addition to working with metric definitions and related values, it is also possible to retrieve additional interesting insights related to Azure resources. As an example, it is possible to query activity log data. The following sample demonstrates using the Azure Monitor REST API to query activity log data within a specific date range for an Azure subscription:

```
$apiVersion = "2014-04-01"
$filter = "eventTimestamp ge '2016-09-23' and eventTimestamp le '2016-09-24'and eventChannels eq 'Admin,
Operation'"
$request =
"https://management.azure.com/subscriptions/${subscriptionId}/providers/microsoft.insights/eventtypes/management
/values?api-version=${apiVersion}&`$filter=${filter}"
(Invoke-RestMethod -Uri $request `
                   -Headers $authHeader `
                   -Method Get `
                   -Verbose).Value | ConvertTo-Json
```

Next steps

- Review the Overview of Monitoring.
- View the Supported metrics with Azure Monitor.
- Review the Microsoft Azure Monitor REST API Reference.
- Review the Azure Management Library.

Azure Monitor PowerShell quick start samples

1/17/2017 • 9 min to read • Edit on GitHub

This article shows you sample PowerShell commands to help you access Azure Monitor features. Azure Monitor allows you to AutoScale Cloud Services, Virtual Machines, and Web Apps and to send alert notifications or call web URLs based on values of configured telemetry data.

> **NOTE**
> Azure Monitor is the new name for what was called "Azure Insights" until Sept 25th, 2016. However, the namespaces and thus the commands below still contain the "insights".

Set up PowerShell

If you haven't already, set up PowerShell to run on your computer. For more information, see How to Install and Configure PowerShell .

Examples in this article

The examples in the article illustrate how you can use Azure Monitor cmdlets. You can also review the entire list of Azure Monitor PowerShell cmdlets at Azure Monitor (Insights) Cmdlets.

Sign in and use subscriptions

First, log into your Azure subscription.

```
Login-AzureRmAccount
```

This requires you to sign in. Once you do, your Account, TenantId and default Subscription Id are displayed. All the Azure cmdlets work in the context of your default subscription. To view the list of subscriptions you have access to, use the following command.

```
Get-AzureRmSubscription
```

To change your working context to a different subscription, use the following command.

```
Set-AzureRmContext -SubscriptionId <subscriptionid>
```

Retrieve Activity log for a subscription

Use the `Get-AzureRmLog` cmdlet. Below are some common examples.

Get log entries from this time/date to present:

```
Get-AzureRmLog -StartTime 2016-03-01T10:30
```

Get log entries between a time/date range:

```
Get-AzureRmLog -StartTime 2015-01-01T10:30 -EndTime 2015-01-01T11:30
```

Get log entries from a specific resource group:

```
Get-AzureRmLog -ResourceGroup 'myrg1'
```

Get log entries from a specific resource provider between a time/date range:

```
Get-AzureRmLog -ResourceProvider 'Microsoft.Web' -StartTime 2015-01-01T10:30 -EndTime 2015-01-01T11:30
```

Get all log entries with a specific caller:

```
Get-AzureRmLog -Caller 'myname@company.com'
```

The following command retrieves the last 1000 events from the activity log:

```
Get-AzureRmLog -MaxEvents 1000
```

`Get-AzureRmLog` supports many other parameters. See the `Get-AzureRmLog` reference for more information.

> **NOTE**
>
> `Get-AzureRmLog` only provides 15 days of history. Using the **-MaxEvents** parameter allows you to query the last N events, beyond 15 days. To access events older than 15 days, use the REST API or SDK (C# sample using the SDK). If you do not include **StartTime**, then the default value is **EndTime** minus one hour. If you do not include **EndTime**, then the default value is current time. All times are in UTC.

Retrieve alerts history

To view all alert events, you can query the Azure Resource Manager logs using the following examples.

```
Get-AzureRmLog -Caller "Microsoft.Insights/alertRules" -DetailedOutput -StartTime 2015-03-01
```

To view the history for a specific alert rule, you can use the `Get-AzureRmAlertHistory` cmdlet, passing in the resource ID of the alert rule.

```
Get-AzureRmAlertHistory -ResourceId
/subscriptions/s1/resourceGroups/rg1/providers/microsoft.insights/alertrules/myalert -StartTime 2016-03-1 -
Status Activated
```

The `Get-AzureRmAlertHistory` cmdlet supports various parameters. More information, see Get-AlertHistory.

Retrieve information on alert rules

All of the following commands act on a Resource Group named "montest".

View all the properties of the alert rule:

```
Get-AzureRmAlertRule -Name simpletestCPU -ResourceGroup montest -DetailedOutput
```

Retrieve all alerts on a resource group:

```
Get-AzureRmAlertRule -ResourceGroup montest
```

Retrieve all alert rules set for a target resource. For example, all alert rules set on a VM.

```
Get-AzureRmAlertRule -ResourceGroup montest -TargetResourceId
/subscriptions/s1/resourceGroups/montest/providers/Microsoft.Compute/virtualMachines/testconfig
```

`Get-AzureRmAlertRule` supports other parameters. See Get-AlertRule for more information.

Create alert rules

You can use the `Add-AlertRule` cmdlet to create, update or disable an alert rule.

You can create email and webhook properties using `New-AzureRmAlertRuleEmail` and `New-AzureRmAlertRuleWebhook`, respectively. In the Alert rule cmdlet, assign these as actions to the **Actions** property of the Alert Rule.

The next section contains a sample that shows you how to create an Alert Rule with various parameters.

Alert rule on a metric

The following table describes the parameters and values used to create an alert using a metric.

PARAMETER	VALUE
Name	simpletestdiskwrite
Location of this alert rule	East US
ResourceGroup	montest
TargetResourceId	/subscriptions/s1/resourceGroups/montest/providers/Microsoft.Compute/virtualMachines/testconfig
MetricName of the alert that is created	\PhysicalDisk(_Total)\Disk Writes/sec. See the `Get-MetricDefinitions` cmdlet below about how to retrieve the exact metric names
operator	GreaterThan
Threshold value (count/sec in for this metric)	1
WindowSize (hh:mm:ss format)	00:05:00
aggregator (statistic of the metric, which uses Average count, in this case)	Average
custom emails (string array)	'foo@example.com','bar@example.com'
send email to owners, contributors and readers	-SendToServiceOwners

Create an Email action

```
$actionEmail = New-AzureRmAlertRuleEmail -CustomEmail myname@company.com
```

Create a Webhook action

```
$actionWebhook = New-AzureRmAlertRuleWebhook -ServiceUri https://example.com?token=mytoken
```

Create the alert rule on the CPU% metric on a classic VM

```
Add-AzureRmMetricAlertRule -Name vmcpu_gt_1 -Location "East US" -ResourceGroup myrg1 -TargetResourceId
/subscriptions/s1/resourceGroups/myrg1/providers/Microsoft.ClassicCompute/virtualMachines/my_vm1 -MetricName
"Percentage CPU" -Operator GreaterThan -Threshold 1 -WindowSize 00:05:00 -TimeAggregationOperator Average -
Actions $actionEmail, $actionWebhook -Description "alert on CPU > 1%"
```

Retrieve the alert rule

```
Get-AzureRmAlertRule -Name vmcpu_gt_1 -ResourceGroup myrg1 -DetailedOutput
```

The Add alert cmdlet also updates the rule if an alert rule already exists for the given properties. To disable an alert rule, include the parameter **-DisableRule**.

Alert on activity log event

> **NOTE**
>
> This feature is in preview and will be removed at some point in the future (it is being replaced).

In this scenario, you'll send email when a website is successfully started in my subscription in resource group *abhingrgtest123*.

Setup an email rule

```
$actionEmail = New-AzureRmAlertRuleEmail -CustomEmail myname@company.com
```

Setup a webhook rule

```
$actionWebhook = New-AzureRmAlertRuleWebhook -ServiceUri https://example.com?token=mytoken
```

Create the rule on the event

```
Add-AzureRmLogAlertRule -Name superalert1 -Location "East US" -ResourceGroup myrg1 -OperationName
microsoft.web/sites/start/action -Status Succeeded -TargetResourceGroup abhingrgtest123 -Actions $actionEmail,
$actionWebhook
```

Retrieve the alert rule

```
Get-AzureRmAlertRule -Name superalert1 -ResourceGroup myrg1 -DetailedOutput
```

The `Add-AlertRule` cmdlet allows various other parameters. More information, see Add-AlertRule.

Get a list of available metrics for alerts

You can use the `Get-AzureRmMetricDefinition` cmdlet to view the list of all metrics for a specific resource.

```
Get-AzureRmMetricDefinition -ResourceId <resource_id>
```

The following example generates a table with the metric Name and the Unit for it.

```
Get-AzureRmMetricDefinition -ResourceId <resource_id> | Format-Table -Property Name,Unit
```

A full list of available options for `Get-AzureRmMetricDefinition` is available at Get-MetricDefinitions.

Create and manage AutoScale settings

A resource, such as a Web app, VM, Cloud Service or VM Scale Set can have only one autoscale setting configured for it. However, each autoscale setting can have multiple profiles. For example, one for a performance-based scale profile and a second one for a schedule based profile. Each profile can have multiple rules configured on it. For more information about Autoscale, see How to Autoscale an Application.

Here are the steps we will use:

1. Create rule(s).
2. Create profile(s) mapping the rules that you created previously to the profiles.
3. Optional: Create notifications for autoscale by configuring webhook and email properties.
4. Create an autoscale setting with a name on the target resource by mapping the profiles and notifications that you created in the previous steps.

The following examples show you how you can create an Autoscale setting for a VM scale set for a Windows operating system based by using the CPU utilization metric.

First, create a rule to scale-out, with an instance count increase .

```
$rule1 = New-AzureRmAutoscaleRule -MetricName "\Processor(_Total)\% Processor Time" -MetricResourceId
/subscriptions/s1/resourceGroups/big2/providers/Microsoft.Compute/virtualMachineScaleSets/big2 -Operator
GreaterThan -MetricStatistic Average -Threshold 0.01 -TimeGrain 00:01:00 -TimeWindow 00:10:00 -
ScaleActionCooldown 00:10:00 -ScaleActionDirection Increase -ScaleActionScaleType ChangeCount -ScaleActionValue
1
```

Next, create a rule to scale-in, with an instance count decrease.

```
$rule2 = New-AzureRmAutoscaleRule -MetricName "\Processor(_Total)\% Processor Time" -MetricResourceId
/subscriptions/s1/resourceGroups/big2/providers/Microsoft.Compute/virtualMachineScaleSets/big2 -Operator
GreaterThan -MetricStatistic Average -Threshold 2 -TimeGrain 00:01:00 -TimeWindow 00:10:00 -ScaleActionCooldown
00:10:00 -ScaleActionDirection Decrease -ScaleActionScaleType ChangeCount -ScaleActionValue 1
```

Then, create a profile for the rules.

```
$profile1 = New-AzureRmAutoscaleProfile -DefaultCapacity 2 -MaximumCapacity 10 -MinimumCapacity 2 -Rules
$rule1,$rule2 -Name "My_Profile"
```

Create a webhook property.

```
$webhook_scale = New-AzureRmAutoscaleWebhook -ServiceUri "https://example.com?mytoken=mytokenvalue"
```

Create the notification property for the autoscale setting, including email and the webhook that you created previously.

```
$notification1= New-AzureRmAutoscaleNotification -CustomEmails ashwink@microsoft.com -
SendEmailToSubscriptionAdministrators SendEmailToSubscriptionCoAdministrators -Webhooks $webhook_scale
```

Finally, create the autoscale setting to add the profile that you created above.

```
Add-AzureRmAutoscaleSetting -Location "East US" -Name "MyScaleVMSSSetting" -ResourceGroup big2 -
TargetResourceId /subscriptions/s1/resourceGroups/big2/providers/Microsoft.Compute/virtualMachineScaleSets/big2
-AutoscaleProfiles $profile1 -Notifications $notification1
```

For more information about managing Autoscale settings, see Get-AutoscaleSetting.

Autoscale history

The following example shows you how you can view recent autoscale and alert events. Use the activity log search to view the autoscale history.

```
Get-AzureRmLog -Caller "Microsoft.Insights/autoscaleSettings" -DetailedOutput -StartTime 2015-03-01
```

You can use the `Get-AzureRmAutoScaleHistory` cmdlet to retrieve AutoScale history.

```
Get-AzureRmAutoScaleHistory -ResourceId
/subscriptions/s1/resourceGroups/myrg1/providers/microsoft.insights/autoscalesettings/myScaleSetting -StartTime
2016-03-15 -DetailedOutput
```

For more information, see Get-AutoscaleHistory.

View details for an autoscale setting

You can use the `Get-Autoscalesetting` cmdlet to retrieve more information about the autoscale setting.

The following example shows details about all autoscale settings in the resource group 'myrg1'.

```
Get-AzureRmAutoscalesetting -ResourceGroup myrg1 -DetailedOutput
```

The following example shows details about all autoscale settings in the resource group 'myrg1' and specifically the autoscale setting named 'MyScaleVMSSSetting'.

```
Get-AzureRmAutoscalesetting -ResourceGroup myrg1 -Name MyScaleVMSSSetting -DetailedOutput
```

Remove an autoscale setting

You can use the `Remove-Autoscalesetting` cmdlet to delete an autoscale setting.

```
Remove-AzureRmAutoscalesetting -ResourceGroup myrg1 -Name MyScaleVMSSSetting
```

Manage log profiles for activity log

You can create a *log profile* and export data from your activity log to a storage account and you can configure data retention for it. Optionally, you can also stream the data to your Event Hub. Note that this feature is currently in Preview and you can only create one log profile per subscription. You can use the following cmdlets with your current subscription to create and manage log profiles. You can also choose a particular subscription. Although PowerShell defaults to the current subscription, you can always change that using `Set-AzureRmContext`. You can configure activity log to route data to any storage account or Event Hub within that subscription. Data is written as blob files in JSON format.

Get a log profile

To fetch your existing log profiles, use the `Get-AzureRmLogProfile` cmdlet.

Add a log profile without data retention

```
Add-AzureRmLogProfile -Name my_log_profile_s1 -StorageAccountId
/subscriptions/s1/resourceGroups/myrg1/providers/Microsoft.Storage/storageAccounts/my_storage -Locations
global,westus,eastus,northeurope,westeurope,eastasia,southeastasia,japaneast,japanwest,northcentralus,southcent
ralus,eastus2,centralus,australiaeast,australiasoutheast,brazilsouth,centralindia,southindia,westindia
```

Remove a log profile

```
Remove-AzureRmLogProfile -name my_log_profile_s1
```

Add a log profile with data retention

You can specify the **-RetentionInDays** property with the number of days, as a positive integer, where the data is retained.

```
Add-AzureRmLogProfile -Name my_log_profile_s1 -StorageAccountId
/subscriptions/s1/resourceGroups/myrg1/providers/Microsoft.Storage/storageAccounts/my_storage -Locations
global,westus,eastus,northeurope,westeurope,eastasia,southeastasia,japaneast,japanwest,northcentralus,southcent
ralus,eastus2,centralus,australiaeast,australiasoutheast,brazilsouth,centralindia,southindia,westindia -
RetentionInDays 90
```

Add log profile with retention and EventHub

In addition to routing your data to storage account, you can also stream it to an Event Hub. Note that in this preview release and the storage account configuration is mandatory but Event Hub configuration is optional.

```
Add-AzureRmLogProfile -Name my_log_profile_s1 -StorageAccountId
/subscriptions/s1/resourceGroups/myrg1/providers/Microsoft.Storage/storageAccounts/my_storage -serviceBusRuleId
/subscriptions/s1/resourceGroups/Default-ServiceBus-
EastUS/providers/Microsoft.ServiceBus/namespaces/mytestSB/authorizationrules/RootManageSharedAccessKey -
Locations
global,westus,eastus,northeurope,westeurope,eastasia,southeastasia,japaneast,japanwest,northcentralus,southcent
ralus,eastus2,centralus,australiaeast,australiasoutheast,brazilsouth,centralindia,southindia,westindia -
RetentionInDays 90
```

Configure diagnostics logs

Many Azure services provide additional logs and telemetry that can be configured to save data in your Azure Storage account, send to Event Hubs, and/or sent to an OMS Log Analytics workspace. That operation can only be performed at a resource level and the storage account or event hub should be present in the same region as the target resource where the diagnostics setting is configured.

Get diagnostic setting

```
Get-AzureRmDiagnosticSetting -ResourceId
/subscriptions/s1/resourceGroups/myrg1/providers/Microsoft.Logic/workflows/andy0315logicapp
```

Disable diagnostic setting

```
Set-AzureRmDiagnosticSetting -ResourceId
/subscriptions/s1/resourceGroups/myrg1/providers/Microsoft.Logic/workflows/andy0315logicapp -StorageAccountId
/subscriptions/s1/resourceGroups/Default-Storage-
WestUS/providers/Microsoft.Storage/storageAccounts/mystorageaccount -Enable $false
```

Enable diagnostic setting without retention

```
Set-AzureRmDiagnosticSetting -ResourceId
/subscriptions/s1/resourceGroups/myrg1/providers/Microsoft.Logic/workflows/andy0315logicapp -StorageAccountId
/subscriptions/s1/resourceGroups/Default-Storage-
WestUS/providers/Microsoft.Storage/storageAccounts/mystorageaccount -Enable $true
```

Enable diagnostic setting with retention

```
Set-AzureRmDiagnosticSetting -ResourceId
/subscriptions/s1/resourceGroups/myrg1/providers/Microsoft.Logic/workflows/andy0315logicapp -StorageAccountId
/subscriptions/s1/resourceGroups/Default-Storage-
WestUS/providers/Microsoft.Storage/storageAccounts/mystorageaccount -Enable $true -RetentionEnabled $true -
RetentionInDays 90
```

Enable diagnostic setting with retention for a specific log category

```
Set-AzureRmDiagnosticSetting -ResourceId /subscriptions/s1/resourceGroups/insights-
integration/providers/Microsoft.Network/networkSecurityGroups/viruela1 -StorageAccountId
/subscriptions/s1/resourceGroups/myrg1/providers/Microsoft.Storage/storageAccounts/saktestsstorage -Categories
NetworkSecurityGroupEvent -Enable $true -RetentionEnabled $true -RetentionInDays 90
```

Enable diagnostic setting for Event Hubs

```
Set-AzureRmDiagnosticSetting -ResourceId /subscriptions/s1/resourceGroups/insights-
integration/providers/Microsoft.Network/networkSecurityGroups/viruela1 -serviceBusRuleId
/subscriptions/s1/resourceGroups/Default-ServiceBus-
EastUS/providers/Microsoft.ServiceBus/namespaces/mytestSB/authorizationrules/RootManageSharedAccessKey -Enable
$true
```

Enable diagnostic setting for OMS

```
Set-AzureRmDiagnosticSetting -ResourceId /subscriptions/s1/resourceGroups/insights-
integration/providers/Microsoft.Network/networkSecurityGroups/viruela1 -WorkspaceId 76d785fd-d1ce-4f50-8ca3-
858fc819ca0f -Enabled $true
```

Azure Monitor Cross-platform CLI quick start samples

1/17/2017 • 3 min to read • Edit on GitHub

This article shows you sample command-line interface (CLI) commands to help you access Azure Monitor features. Azure Monitor allows you to AutoScale Cloud Services, Virtual Machines, and Web Apps and to send alert notifications or call web URLs based on values of configured telemetry data.

> **NOTE**
> Azure Monitor is the new name for what was called "Azure Insights" until Sept 25th, 2016. However, the namespaces and thus the commands below still contain the "insights".

Prerequisites

If you haven't already installed the Azure CLI, see Install the Azure CLI. If you're unfamiliar with Azure CLI, you can read more about it at Use the Azure CLI for Mac, Linux, and Windows with Azure Resource Manager.

In Windows, install npm from the Node.js website. After you complete the installation, using CMD.exe with Run As Administrator privileges, execute the following from the folder where npm is installed:

```
npm install azure-cli --global
```

Next, navigate to any folder/location you want and type at the command-line:

```
azure help
```

Log in to Azure

The first step is to login to your Azure account.

```
azure login
```

After running this command, you have to sign in via the instructions on the screen. Afterward, you see your Account, TenantId, and default Subscription Id. All commands work in the context of your default subscription.

To list the details of your current subscription, use the following command.

```
azure account show
```

To change working context to a different subscription, use the following command.

```
azure account set "subscription ID or subscription name"
```

To use Azure Resource Manager and Azure Monitor commands, you need to be in Azure Resource Manager mode.

```
azure config mode arm
```

To view a list of all supported Azure Monitor commands, perform the following.

```
azure insights
```

View activity log for a subscription

To view a list of activity log events, perform the following.

```
azure insights logs list [options]
```

Try the following to view all available options.

```
azure insights logs list -help
```

Here is an example to list logs by a resourceGroup

```
azure insights logs list --resourceGroup "myrg1"
```

Example to list logs by caller

```
azure insights logs list --caller "myname@company.com"
```

Example to list logs by caller on a resource type, within a start and end date

```
azure insights logs list --resourceProvider "Microsoft.Web" --caller "myname@company.com" --startTime 2016-03-
08T00:00:00Z --endTime 2016-03-16T00:00:00Z
```

Work with alerts

You can use the information in the section to work with alerts.

Get alert rules in a resource group

```
azure insights alerts rule list abhingrgtest123
azure insights alerts rule list abhingrgtest123 --ruleName andy0323
```

Create a metric alert rule

```
azure insights alerts actions email create --customEmails foo@microsoft.com
azure insights alerts actions webhook create https://someuri.com
azure insights alerts rule metric set andy0323 eastus abhingrgtest123 PT5M GreaterThan 2
/subscriptions/df602c9c-7aa0-407d-a6fb-eb20c8bd1192/resourceGroups/Default-Web-
EastUS/providers/Microsoft.Web/serverfarms/Default1 BytesReceived Total
```

Create a log alert rule

```
azure insights alerts rule log set ruleName eastus resourceGroupName someOperationName
```

Create webtest alert rule

```
azure insights alerts rule webtest set leowebtestr1-webtestr1 eastus Default-Web-WestUS PT5M 1 GSMT_AvRaw
/subscriptions/b67f7fec-69fc-4974-9099-a26bd6ffeda3/resourcegroups/Default-Web-
WestUS/providers/microsoft.insights/webtests/leowebtestr1-webtestr1
```

Delete an alert rule

```
azure insights alerts rule delete abhingrgtest123 andy0323
```

Log profiles

Use the information in this section to work with log profiles.

Get a log profile

```
azure insights logprofile list
azure insights logprofile get -n default
```

Add a log profile without retention

```
azure insights logprofile add --name default --storageId /subscriptions/1a66ce04-b633-4a0b-b2bc-
a912ec8986a6/resourceGroups/insights-
integration/providers/Microsoft.Storage/storageAccounts/insightsintegration7777 --locations
global,westus,eastus,northeurope,westeurope,eastasia,southeastasia,japaneast,japanwest,northcentralus,southcent
ralus,eastus2,centralus,australiaeast,australiasoutheast,brazilsouth,centralindia,southindia,westindia
```

Remove a log profile

```
azure insights logprofile delete --name default
```

Add a log profile with retention

```
azure insights logprofile add --name default --storageId /subscriptions/1a66ce04-b633-4a0b-b2bc-
a912ec8986a6/resourceGroups/insights-
integration/providers/Microsoft.Storage/storageAccounts/insightsintegration7777 --locations
global,westus,eastus,northeurope,westeurope,eastasia,southeastasia,japaneast,japanwest,northcentralus,southcent
ralus,eastus2,centralus,australiaeast,australiasoutheast,brazilsouth,centralindia,southindia,westindia --
retentionInDays 90
```

Add a log profile with retention and EventHub

```
azure insights logprofile add --name default --storageId /subscriptions/1a66ce04-b633-4a0b-b2bc-
a912ec8986a6/resourceGroups/insights-
integration/providers/Microsoft.Storage/storageAccounts/insightsintegration7777 --serviceBusRuleId
/subscriptions/1a66ce04-b633-4a0b-b2bc-a912ec8986a6/resourceGroups/Default-ServiceBus-
EastUS/providers/Microsoft.ServiceBus/namespaces/testshoeboxeastus/authorizationrules/RootManageSharedAccessKey
--locations
global,westus,eastus,northeurope,westeurope,eastasia,southeastasia,japaneast,japanwest,northcentralus,southcent
ralus,eastus2,centralus,australiaeast,australiasoutheast,brazilsouth,centralindia,southindia,westindia --
retentionInDays 90
```

Diagnostics

Use the information in this section to work with diagnostic settings.

Get a diagnostic setting

```
azure insights diagnostic get --resourceId /subscriptions/df602c9c-7aa0-407d-a6fb-
eb20c8bd1192/resourceGroups/andyrg/providers/Microsoft.Logic/workflows/andy0315logicapp
```

Disable a diagnostic setting

```
azure insights diagnostic set --resourceId /subscriptions/df602c9c-7aa0-407d-a6fb-
eb20c8bd1192/resourceGroups/andyrg/providers/Microsoft.Logic/workflows/andy0315logicapp --storageId
/subscriptions/df602c9c-7aa0-407d-a6fb-eb20c8bd1192/resourceGroups/Default-Storage-
WestUS/providers/Microsoft.Storage/storageAccounts/shibanitesting --enabled false
```

Enable a diagnostic setting without retention

```
azure insights diagnostic set --resourceId /subscriptions/df602c9c-7aa0-407d-a6fb-
eb20c8bd1192/resourceGroups/andyrg/providers/Microsoft.Logic/workflows/andy0315logicapp --storageId
/subscriptions/df602c9c-7aa0-407d-a6fb-eb20c8bd1192/resourceGroups/Default-Storage-
WestUS/providers/Microsoft.Storage/storageAccounts/shibanitesting --enabled true
```

Autoscale

Use the information in this section to work with autoscale settings. You need to modify these examples.

Get autoscale settings for a resource group

```
azure insights autoscale setting list montest2
```

Get autoscale settings by name in a resource group

```
azure insights autoscale setting list montest2 -n setting2
```

Set auotoscale settings

```
azure insights autoscale setting set montest2 -n setting2 --settingSpec
```

Supported metrics with Azure Monitor

1/17/2017 • 18 min to read • Edit on GitHub

Azure Monitor provides several ways to interact with metrics, including charting them in the portal, accessing them through the REST API, or querying them using PowerShell or CLI. Below is a complete list of all metrics currently available with Azure Monitor's metric pipeline.

> **NOTE**
> Other metrics may be available in the portal or using legacy APIs. This list only includes public preview metrics available using the public preview of the consolidated Azure Monitor metric pipeline.

Microsoft.AnalysisServices/servers

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
qpu_metric	QPU	Count	Average	QPU. Range 0-100 for S1, 0-200 for S2 and 0-400 for S4
memory_metric	Memory	Bytes	Average	Memory. Range 0-25 GB for S1, 0-50 GB for S2 and 0-100 GB for S4

Microsoft.Batch/batchAccounts

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
CoreCount	Core Count	Count	Total	Total number of cores in the batch account
TotalNodeCount	Node Count	Count	Total	Total number of nodes in the batch account
CreatingNodeCount	Creating Node Count	Count	Total	Number of nodes being created
StartingNodeCount	Starting Node Count	Count	Total	Number of nodes starting
WaitingForStartTaskNodeCount	Waiting For Start Task Node Count	Count	Total	Number of nodes waiting for the Start Task to complete
StartTaskFailedNodeCount	Start Task Failed Node Count	Count	Total	Number of nodes where the Start Task has failed

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
IdleNodeCount	Idle Node Count	Count	Total	Number of idle nodes
OfflineNodeCount	Offline Node Count	Count	Total	Number of offline nodes
RebootingNodeCount	Rebooting Node Count	Count	Total	Number of rebooting nodes
ReimagingNodeCount	Reimaging Node Count	Count	Total	Number of reimaging nodes
RunningNodeCount	Running Node Count	Count	Total	Number of running nodes
LeavingPoolNodeCount	Leaving Pool Node Count	Count	Total	Number of nodes leaving the Pool
UnusableNodeCount	Unusable Node Count	Count	Total	Number of unusable nodes
TaskStartEvent	Task Start Events	Count	Total	Total number of tasks that have started
TaskCompleteEvent	Task Complete Events	Count	Total	Total number of tasks that have completed
TaskFailEvent	Task Fail Events	Count	Total	Total number of tasks that have completed in a failed state
PoolCreateEvent	Pool Create Events	Count	Total	Total number of pools that have been created
PoolResizeStartEvent	Pool Resize Start Events	Count	Total	Total number of pool resizes that have started
PoolResizeCompleteEvent	Pool Resize Complete Events	Count	Total	Total number of pool resizes that have completed
PoolDeleteStartEvent	Pool Delete Start Events	Count	Total	Total number of pool deletes that have started
PoolDeleteCompleteEvent	Pool Delete Complete Events	Count	Total	Total number of pool deletes that have completed

Microsoft.Cache/redis

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
connectedclients	Connected Clients	Count	Maximum	
totalcommandsprocessed	Total Operations	Count	Total	
cachehits	Cache Hits	Count	Total	
cachemisses	Cache Misses	Count	Total	
getcommands	Gets	Count	Total	
setcommands	Sets	Count	Total	
evictedkeys	Evicted Keys	Count	Total	
totalkeys	Total Keys	Count	Maximum	
expiredkeys	Expired Keys	Count	Total	
usedmemory	Used Memory	Bytes	Maximum	
usedmemoryRss	Used Memory RSS	Bytes	Maximum	
serverLoad	Server Load	Percent	Maximum	
cacheWrite	Cache Write	BytesPerSecond	Maximum	
cacheRead	Cache Read	BytesPerSecond	Maximum	
percentProcessorTime	CPU	Percent	Maximum	
connectedclients0	Connected Clients (Shard 0)	Count	Maximum	
totalcommandsprocessed0	Total Operations (Shard 0)	Count	Total	
cachehits0	Cache Hits (Shard 0)	Count	Total	
cachemisses0	Cache Misses (Shard 0)	Count	Total	
getcommands0	Gets (Shard 0)	Count	Total	
setcommands0	Sets (Shard 0)	Count	Total	
evictedkeys0	Evicted Keys (Shard 0)	Count	Total	
totalkeys0	Total Keys (Shard 0)	Count	Maximum	
expiredkeys0	Expired Keys (Shard 0)	Count	Total	

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
usedmemory0	Used Memory (Shard 0)	Bytes	Maximum	
usedmemoryRss0	Used Memory RSS (Shard 0)	Bytes	Maximum	
serverLoad0	Server Load (Shard 0)	Percent	Maximum	
cacheWrite0	Cache Write (Shard 0)	BytesPerSecond	Maximum	
cacheRead0	Cache Read (Shard 0)	BytesPerSecond	Maximum	
percentProcessorTime0	CPU (Shard 0)	Percent	Maximum	
connectedclients1	Connected Clients (Shard 1)	Count	Maximum	
totalcommandsprocessed1	Total Operations (Shard 1)	Count	Total	
cachehits1	Cache Hits (Shard 1)	Count	Total	
cachemisses1	Cache Misses (Shard 1)	Count	Total	
getcommands1	Gets (Shard 1)	Count	Total	
setcommands1	Sets (Shard 1)	Count	Total	
evictedkeys1	Evicted Keys (Shard 1)	Count	Total	
totalkeys1	Total Keys (Shard 1)	Count	Maximum	
expiredkeys1	Expired Keys (Shard 1)	Count	Total	
usedmemory1	Used Memory (Shard 1)	Bytes	Maximum	
usedmemoryRss1	Used Memory RSS (Shard 1)	Bytes	Maximum	
serverLoad1	Server Load (Shard 1)	Percent	Maximum	
cacheWrite1	Cache Write (Shard 1)	BytesPerSecond	Maximum	
cacheRead1	Cache Read (Shard 1)	BytesPerSecond	Maximum	
percentProcessorTime1	CPU (Shard 1)	Percent	Maximum	

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
connectedclients2	Connected Clients (Shard 2)	Count	Maximum	
totalcommandsprocessed2	Total Operations (Shard 2)	Count	Total	
cachehits2	Cache Hits (Shard 2)	Count	Total	
cachemisses2	Cache Misses (Shard 2)	Count	Total	
getcommands2	Gets (Shard 2)	Count	Total	
setcommands2	Sets (Shard 2)	Count	Total	
evictedkeys2	Evicted Keys (Shard 2)	Count	Total	
totalkeys2	Total Keys (Shard 2)	Count	Maximum	
expiredkeys2	Expired Keys (Shard 2)	Count	Total	
usedmemory2	Used Memory (Shard 2)	Bytes	Maximum	
usedmemoryRss2	Used Memory RSS (Shard 2)	Bytes	Maximum	
serverLoad2	Server Load (Shard 2)	Percent	Maximum	
cacheWrite2	Cache Write (Shard 2)	BytesPerSecond	Maximum	
cacheRead2	Cache Read (Shard 2)	BytesPerSecond	Maximum	
percentProcessorTime2	CPU (Shard 2)	Percent	Maximum	
connectedclients3	Connected Clients (Shard 3)	Count	Maximum	
totalcommandsprocessed3	Total Operations (Shard 3)	Count	Total	
cachehits3	Cache Hits (Shard 3)	Count	Total	
cachemisses3	Cache Misses (Shard 3)	Count	Total	
getcommands3	Gets (Shard 3)	Count	Total	
setcommands3	Sets (Shard 3)	Count	Total	
evictedkeys3	Evicted Keys (Shard 3)	Count	Total	

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
totalkeys3	Total Keys (Shard 3)	Count	Maximum	
expiredkeys3	Expired Keys (Shard 3)	Count	Total	
usedmemory3	Used Memory (Shard 3)	Bytes	Maximum	
usedmemoryRss3	Used Memory RSS (Shard 3)	Bytes	Maximum	
serverLoad3	Server Load (Shard 3)	Percent	Maximum	
cacheWrite3	Cache Write (Shard 3)	BytesPerSecond	Maximum	
cacheRead3	Cache Read (Shard 3)	BytesPerSecond	Maximum	
percentProcessorTime3	CPU (Shard 3)	Percent	Maximum	
connectedclients4	Connected Clients (Shard 4)	Count	Maximum	
totalcommandsprocessed4	Total Operations (Shard 4)	Count	Total	
cachehits4	Cache Hits (Shard 4)	Count	Total	
cachemisses4	Cache Misses (Shard 4)	Count	Total	
getcommands4	Gets (Shard 4)	Count	Total	
setcommands4	Sets (Shard 4)	Count	Total	
evictedkeys4	Evicted Keys (Shard 4)	Count	Total	
totalkeys4	Total Keys (Shard 4)	Count	Maximum	
expiredkeys4	Expired Keys (Shard 4)	Count	Total	
usedmemory4	Used Memory (Shard 4)	Bytes	Maximum	
usedmemoryRss4	Used Memory RSS (Shard 4)	Bytes	Maximum	
serverLoad4	Server Load (Shard 4)	Percent	Maximum	
cacheWrite4	Cache Write (Shard 4)	BytesPerSecond	Maximum	
cacheRead4	Cache Read (Shard 4)	BytesPerSecond	Maximum	

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
percentProcessorTime4	CPU (Shard 4)	Percent	Maximum	
connectedclients5	Connected Clients (Shard 5)	Count	Maximum	
totalcommandsprocessed5	Total Operations (Shard 5)	Count	Total	
cachehits5	Cache Hits (Shard 5)	Count	Total	
cachemisses5	Cache Misses (Shard 5)	Count	Total	
getcommands5	Gets (Shard 5)	Count	Total	
setcommands5	Sets (Shard 5)	Count	Total	
evictedkeys5	Evicted Keys (Shard 5)	Count	Total	
totalkeys5	Total Keys (Shard 5)	Count	Maximum	
expiredkeys5	Expired Keys (Shard 5)	Count	Total	
usedmemory5	Used Memory (Shard 5)	Bytes	Maximum	
usedmemoryRss5	Used Memory RSS (Shard 5)	Bytes	Maximum	
serverLoad5	Server Load (Shard 5)	Percent	Maximum	
cacheWrite5	Cache Write (Shard 5)	BytesPerSecond	Maximum	
cacheRead5	Cache Read (Shard 5)	BytesPerSecond	Maximum	
percentProcessorTime5	CPU (Shard 5)	Percent	Maximum	
connectedclients6	Connected Clients (Shard 6)	Count	Maximum	
totalcommandsprocessed6	Total Operations (Shard 6)	Count	Total	
cachehits6	Cache Hits (Shard 6)	Count	Total	
cachemisses6	Cache Misses (Shard 6)	Count	Total	
getcommands6	Gets (Shard 6)	Count	Total	

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
setcommands6	Sets (Shard 6)	Count	Total	
evictedkeys6	Evicted Keys (Shard 6)	Count	Total	
totalkeys6	Total Keys (Shard 6)	Count	Maximum	
expiredkeys6	Expired Keys (Shard 6)	Count	Total	
usedmemory6	Used Memory (Shard 6)	Bytes	Maximum	
usedmemoryRss6	Used Memory RSS (Shard 6)	Bytes	Maximum	
serverLoad6	Server Load (Shard 6)	Percent	Maximum	
cacheWrite6	Cache Write (Shard 6)	BytesPerSecond	Maximum	
cacheRead6	Cache Read (Shard 6)	BytesPerSecond	Maximum	
percentProcessorTime6	CPU (Shard 6)	Percent	Maximum	
connectedclients7	Connected Clients (Shard 7)	Count	Maximum	
totalcommandsprocessed7	Total Operations (Shard 7)	Count	Total	
cachehits7	Cache Hits (Shard 7)	Count	Total	
cachemisses7	Cache Misses (Shard 7)	Count	Total	
getcommands7	Gets (Shard 7)	Count	Total	
setcommands7	Sets (Shard 7)	Count	Total	
evictedkeys7	Evicted Keys (Shard 7)	Count	Total	
totalkeys7	Total Keys (Shard 7)	Count	Maximum	
expiredkeys7	Expired Keys (Shard 7)	Count	Total	
usedmemory7	Used Memory (Shard 7)	Bytes	Maximum	
usedmemoryRss7	Used Memory RSS (Shard 7)	Bytes	Maximum	
serverLoad7	Server Load (Shard 7)	Percent	Maximum	

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
cacheWrite7	Cache Write (Shard 7)	BytesPerSecond	Maximum	
cacheRead7	Cache Read (Shard 7)	BytesPerSecond	Maximum	
percentProcessorTime7	CPU (Shard 7)	Percent	Maximum	
connectedclients8	Connected Clients (Shard 8)	Count	Maximum	
totalcommandsprocessed8	Total Operations (Shard 8)	Count	Total	
cachehits8	Cache Hits (Shard 8)	Count	Total	
cachemisses8	Cache Misses (Shard 8)	Count	Total	
getcommands8	Gets (Shard 8)	Count	Total	
setcommands8	Sets (Shard 8)	Count	Total	
evictedkeys8	Evicted Keys (Shard 8)	Count	Total	
totalkeys8	Total Keys (Shard 8)	Count	Maximum	
expiredkeys8	Expired Keys (Shard 8)	Count	Total	
usedmemory8	Used Memory (Shard 8)	Bytes	Maximum	
usedmemoryRss8	Used Memory RSS (Shard 8)	Bytes	Maximum	
serverLoad8	Server Load (Shard 8)	Percent	Maximum	
cacheWrite8	Cache Write (Shard 8)	BytesPerSecond	Maximum	
cacheRead8	Cache Read (Shard 8)	BytesPerSecond	Maximum	
percentProcessorTime8	CPU (Shard 8)	Percent	Maximum	
connectedclients9	Connected Clients (Shard 9)	Count	Maximum	
totalcommandsprocessed9	Total Operations (Shard 9)	Count	Total	
cachehits9	Cache Hits (Shard 9)	Count	Total	

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
cachemisses9	Cache Misses (Shard 9)	Count	Total	
getcommands9	Gets (Shard 9)	Count	Total	
setcommands9	Sets (Shard 9)	Count	Total	
evictedkeys9	Evicted Keys (Shard 9)	Count	Total	
totalkeys9	Total Keys (Shard 9)	Count	Maximum	
expiredkeys9	Expired Keys (Shard 9)	Count	Total	
usedmemory9	Used Memory (Shard 9)	Bytes	Maximum	
usedmemoryRss9	Used Memory RSS (Shard 9)	Bytes	Maximum	
serverLoad9	Server Load (Shard 9)	Percent	Maximum	
cacheWrite9	Cache Write (Shard 9)	BytesPerSecond	Maximum	
cacheRead9	Cache Read (Shard 9)	BytesPerSecond	Maximum	
percentProcessorTime9	CPU (Shard 9)	Percent	Maximum	

Microsoft.CognitiveServices/accounts

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
TotalCalls	Total Calls	Count	Total	Total number of calls.
SuccessfulCalls	Successful Calls	Count	Total	Number of successful calls.
TotalErrors	Total Errors	Count	Total	Total number of calls with error response (HTTP response code 4xx or 5xx).
BlockedCalls	Blocked Calls	Count	Total	Number of calls that exceeded rate or quota limit.
ServerErrors	Server Errors	Count	Total	Number of calls with service internal error (HTTP response code 5xx).

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
ClientErrors	Client Errors	Count	Total	Number of calls with client side error (HTTP response code 4xx).
DataIn	Data In	Bytes	Total	Size of incoming data in bytes.
DataOut	Data Out	Bytes	Total	Size of outgoing data in bytes.
Latency	Latency	MilliSeconds	Average	Latency in milliseconds.

Microsoft.Compute/virtualMachines

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
Percentage CPU	Percentage CPU	Percent	Average	The percentage of allocated compute units that are currently in use by the Virtual Machine(s)
Network In	Network In	Bytes	Total	The number of bytes received on all network interfaces by the Virtual Machine(s) (Incoming Traffic)
Network Out	Network Out	Bytes	Total	The number of bytes out on all network interfaces by the Virtual Machine(s) (Outgoing Traffic)
Disk Read Bytes	Disk Read Bytes	Bytes	Total	Total bytes read from disk during monitoring period
Disk Write Bytes	Disk Write Bytes	Bytes	Total	Total bytes written to disk during monitoring period
Disk Read Operations/Sec	Disk Read Operations/Sec	CountPerSecond	Average	Disk Read IOPS
Disk Write Operations/Sec	Disk Write Operations/Sec	CountPerSecond	Average	Disk Write IOPS

Microsoft.Compute/virtualMachineScaleSets

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
Percentage CPU	Percentage CPU	Percent	Average	The percentage of allocated compute units that are currently in use by the Virtual Machine(s)
Network In	Network In	Bytes	Total	The number of bytes received on all network interfaces by the Virtual Machine(s) (Incoming Traffic)
Network Out	Network Out	Bytes	Total	The number of bytes out on all network interfaces by the Virtual Machine(s) (Outgoing Traffic)
Disk Read Bytes	Disk Read Bytes	Bytes	Total	Total bytes read from disk during monitoring period
Disk Write Bytes	Disk Write Bytes	Bytes	Total	Total bytes written to disk during monitoring period
Disk Read Operations/Sec	Disk Read Operations/Sec	CountPerSecond	Average	Disk Read IOPS
Disk Write Operations/Sec	Disk Write Operations/Sec	CountPerSecond	Average	Disk Write IOPS

Microsoft.Compute/virtualMachineScaleSets/virtualMachines

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
Percentage CPU	Percentage CPU	Percent	Average	The percentage of allocated compute units that are currently in use by the Virtual Machine(s)
Network In	Network In	Bytes	Total	The number of bytes received on all network interfaces by the Virtual Machine(s) (Incoming Traffic)
Network Out	Network Out	Bytes	Total	The number of bytes out on all network interfaces by the Virtual Machine(s) (Outgoing Traffic)

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
Disk Read Bytes	Disk Read Bytes	Bytes	Total	Total bytes read from disk during monitoring period
Disk Write Bytes	Disk Write Bytes	Bytes	Total	Total bytes written to disk during monitoring period
Disk Read Operations/Sec	Disk Read Operations/Sec	CountPerSecond	Average	Disk Read IOPS
Disk Write Operations/Sec	Disk Write Operations/Sec	CountPerSecond	Average	Disk Write IOPS

Microsoft.Devices/IotHubs

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
d2c.telemetry.ingress.allProtocol	Telemetry Message Send Attempts	Count	Total	Number of device-to-cloud telemetry messages attempted to be sent to your IoT hub
d2c.telemetry.ingress.success	Telemetry Messages Sent	Count	Total	Number of device-to-cloud telemetry messages sent successfully to your IoT hub
d2c.telemetry.egress.success	Telemetry messages delivered	Count	Total	The count of all successful writes to an endpoint
d2c.telemetry.egress.invalid	Invalid telemetry message delivery attempts	Count	Total	The count of messages not delivered due to incompatibility with the endpoint
d2c.telemetry.egress.dropped	Dropped telemetry messages	Count	Total	The count of messages dropped because an endpoint was unhealthy
d2c.telemetry.egress.fallback	Telemetry messages matching fallback condition	Count	Total	The count of messages matching the fallback route
d2c.telemetry.egress.orphaned	Orphaned telemetry messages	Count	Total	The count of messages not matching any routes including the fallback route

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
d2c.endpoints.latency.eventHubs	Message latency for Event Hub endpoints	Milliseconds	Average	The average, min, and max latency between message ingress to the IoT hub and message ingress into an Event Hub endpoint, in milliseconds
d2c.endpoints.latency.serviceBusQueues	Message latency for Service Bus Queue endpoints	Milliseconds	Average	The average, min, and max latency between message ingress to the IoT hub and message ingress into a Service Bus Queue endpoint, in milliseconds
d2c.endpoints.latency.serviceBusTopics	Message latency for Service Bus Topic endpoints	Milliseconds	Average	The average, min, and max latency between message ingress to the IoT hub and message ingress into a Service Bus Topic endpoint, in milliseconds
c2d.commands.egress.complete.success	Commands Completed	Count	Total	Number of Cloud to Device commands completed successfully by the device
c2d.commands.egress.abandon.success	Commands Abandoned	Count	Total	Number of Cloud to Device commands abandoned by the device
c2d.commands.egress.reject.success	Commands Rejected	Count	Total	Number of Cloud to Device commands rejected by the device
devices.totalDevices	Total Devices	Count	Total	Number of devices registered to your IoT hub
devices.connectedDevices.allProtocol	Connected Devices	Count	Total	Number of devices connected to your IoT hub

Microsoft.EventHub/namespaces

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
INREQS	Incoming Requests	Count	Total	Event Hub incoming message throughput for a namespace
SUCCREQ	Successful Requests	Count	Total	Total successful requests for a namespace
FAILREQ	Failed Requests	Count	Total	Total failed requests for a namespace
SVRBSY	Server Busy Errors	Count	Total	Total server busy errors for a namespace
INTERR	Internal Server Errors	Count	Total	Total internal server errors for a namespace
MISCERR	Other Errors	Count	Total	Total failed requests for a namespace
INMSGS	Incoming Messages	Count	Total	Total incoming messages for a namespace
OUTMSGS	Outgoing Messages	Count	Total	Total outgoing messages for a namespace
EHINMBS	Incoming bytes per second	BytesPerSecond	Total	Event Hub incoming message throughput for a namespace
EHOUTMBS	Outgoing bytes per second	BytesPerSecond	Total	Total outgoing messages for a namespace
EHABL	Archive backlog messages	Count	Total	Event Hub archive messages in backlog for a namespace
EHAMSGS	Archive messages	Count	Total	Event Hub archived messages in a namespace
EHAMBS	Archive message throughput	BytesPerSecond	Total	Event Hub archived message throughput in a namespace

Microsoft.Logic/workflows

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
RunsStarted	Runs Started	Count	Total	Number of workflow runs started.
RunsCompleted	Runs Completed	Count	Total	Number of workflow runs completed.
RunsSucceeded	Runs Succeeded	Count	Total	Number of workflow runs succeeded.
RunsFailed	Runs Failed	Count	Total	Number of workflow runs failed.
RunsCancelled	Runs Cancelled	Count	Total	Number of workflow runs cancelled.
RunLatency	Run Latency	Seconds	Average	Latency of completed workflow runs.
RunSuccessLatency	Run Success Latency	Seconds	Average	Latency of succeeded workflow runs.
RunThrottledEvents	Run Throttled Events	Count	Total	Number of workflow action or trigger throttled events.
RunFailurePercentage	Run Failure Percentage	Percent	Total	Percentage of workflow runs failed.
ActionsStarted	Actions Started	Count	Total	Number of workflow actions started.
ActionsCompleted	Actions Completed	Count	Total	Number of workflow actions completed.
ActionsSucceeded	Actions Succeeded	Count	Total	Number of workflow actions succeeded.
ActionsFailed	Actions Failed	Count	Total	Number of workflow actions failed.
ActionsSkipped	Actions Skipped	Count	Total	Number of workflow actions skipped.
ActionLatency	Action Latency	Seconds	Average	Latency of completed workflow actions.
ActionSuccessLatency	Action Success Latency	Seconds	Average	Latency of succeeded workflow actions.
ActionThrottledEvents	Action Throttled Events	Count	Total	Number of workflow action throttled events..

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
TriggersStarted	Triggers Started	Count	Total	Number of workflow triggers started.
TriggersCompleted	Triggers Completed	Count	Total	Number of workflow triggers completed.
TriggersSucceeded	Triggers Succeeded	Count	Total	Number of workflow triggers succeeded.
TriggersFailed	Triggers Failed	Count	Total	Number of workflow triggers failed.
TriggersSkipped	Triggers Skipped	Count	Total	Number of workflow triggers skipped.
TriggersFired	Triggers Fired	Count	Total	Number of workflow triggers fired.
TriggerLatency	Trigger Latency	Seconds	Average	Latency of completed workflow triggers.
TriggerFireLatency	Trigger Fire Latency	Seconds	Average	Latency of fired workflow triggers.
TriggerSuccessLatency	Trigger Success Latency	Seconds	Average	Latency of succeeded workflow triggers.
TriggerThrottledEvents	Trigger Throttled Events	Count	Total	Number of workflow trigger throttled events.
BillableActionExecutions	Billable Action Executions	Count	Total	Number of workflow action executions getting billed.
BillableTriggerExecutions	Billable Trigger Executions	Count	Total	Number of workflow trigger executions getting billed.
TotalBillableExecutions	Total Billable Executions	Count	Total	Number of workflow executions getting billed.

Microsoft.Network/applicationGateways

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
Throughput	Throughput	BytesPerSecond	Average	

Microsoft.Search/searchServices

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
SearchLatency	Search Latency	Seconds	Average	Average search latency for the search service
SearchQueriesPerSecond	Search queries per second	CountPerSecond	Average	Search queries per second for the search service
ThrottledSearchQueriesPercentage	Throttled search queries percentage	Percent	Average	Percentage of search queries that were throttled for the search service

Microsoft.ServiceBus/namespaces

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
CPUXNS	CPU usage per namespace	Percent	Maximum	Service bus premium namespace CPU usage metric
WSXNS	Memory size usage per namespace	Percent	Maximum	Service bus premium namespace memory usage metric

Microsoft.Sql/servers/databases

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
cpu_percent	CPU percentage	Percent	Average	CPU percentage
physical_data_read_percent	Data IO percentage	Percent	Average	Data IO percentage
log_write_percent	Log IO percentage	Percent	Average	Log IO percentage
dtu_consumption_percent	DTU percentage	Percent	Average	DTU percentage
storage	Total database size	Bytes	Maximum	Total database size
connection_successful	Successful Connections	Count	Total	Successful Connections
connection_failed	Failed Connections	Count	Total	Failed Connections
blocked_by_firewall	Blocked by Firewall	Count	Total	Blocked by Firewall
deadlock	Deadlocks	Count	Total	Deadlocks
storage_percent	Database size percentage	Percent	Maximum	Database size percentage

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
xtp_storage_percent	In-Memory OLTP storage percent(Preview)	Percent	Average	In-Memory OLTP storage percent(Preview)
workers_percent	Workers percentage	Percent	Average	Workers percent
sessions_percent	Sessions percent	Percent	Average	Sessions percent
dtu_limit	DTU limit	Count	Average	DTU limit
dtu_used	DTU used	Count	Average	DTU used
service_level_objective	Service level objective of the database	Count	Total	Service level objective of the database
dwu_limit	dwu limit	Count	Maximum	dwu limit
dwu_consumption_percent	DWU percentage	Percent	Average	DWU percentage
dwu_used	DWU used	Count	Average	DWU used

Microsoft.Sql/servers/elasticPools

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
cpu_percent	CPU percentage	Percent	Average	CPU percentage
physical_data_read_percent	Data IO percentage	Percent	Average	Data IO percentage
log_write_percent	Log IO percentage	Percent	Average	Log IO percentage
dtu_consumption_percent	DTU percentage	Percent	Average	DTU percentage
storage_percent	Storage percentage	Percent	Average	Storage percentage
workers_percent	Workers percent	Percent	Average	Workers percent
sessions_percent	Sessions percent	Percent	Average	Sessions percent
eDTU_limit	eDTU limit	Count	Average	eDTU limit
storage_limit	Storage limit	Bytes	Average	Storage limit
eDTU_used	eDTU used	Count	Average	eDTU used
storage_used	Storage used	Bytes	Average	Storage used

Microsoft.StreamAnalytics/streamingjobs

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
ResourceUtilization	SU % Utilization	Percent	Maximum	SU % Utilization
InputEvents	Input Events	Count	Total	Input Events
InputEventBytes	Input Event Bytes	Bytes	Total	Input Event Bytes
LateInputEvents	Late Input Events	Count	Total	Late Input Events
OutputEvents	Output Events	Count	Total	Output Events
ConversionErrors	Data Conversion Errors	Count	Total	Data Conversion Errors
Errors	Runtime Errors	Count	Total	Runtime Errors
DroppedOrAdjustedEvents	Out of order Events	Count	Total	Out of order Events
AMLCalloutRequests	Function Requests	Count	Total	Function Requests
AMLCalloutFailedRequests	Failed Function Requests	Count	Total	Failed Function Requests
AMLCalloutInputEvents	Function Events	Count	Total	Function Events

Microsoft.Web/serverfarms

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
CpuPercentage	CPU Percentage	Percent	Average	CPU Percentage
MemoryPercentage	Memory Percentage	Percent	Average	Memory Percentage
DiskQueueLength	Disk Queue Length	Count	Total	Disk Queue Length
HttpQueueLength	Http Queue Length	Count	Total	Http Queue Length
BytesReceived	Data In	Bytes	Total	Data In
BytesSent	Data Out	Bytes	Total	Data Out

Microsoft.Web/sites (including Azure Functions)

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
CpuTime	CPU Time	Seconds	Total	CPU Time

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
Requests	Requests	Count	Total	Requests
BytesReceived	Data In	Bytes	Total	Data In
BytesSent	Data Out	Bytes	Total	Data Out
Http101	Http 101	Count	Total	Http 101
Http2xx	Http 2xx	Count	Total	Http 2xx
Http3xx	Http 3xx	Count	Total	Http 3xx
Http401	Http 401	Count	Total	Http 401
Http403	Http 403	Count	Total	Http 403
Http404	Http 404	Count	Total	Http 404
Http406	Http 406	Count	Total	Http 406
Http4xx	Http 4xx	Count	Total	Http 4xx
Http5xx	Http Server Errors	Count	Total	Http Server Errors
MemoryWorkingSet	Memory working set	Bytes	Total	Memory working set
AverageMemoryWorkingSet	Average memory working set	Bytes	Total	Average memory working set
AverageResponseTime	Average Response Time	Seconds	Average	Average Response Time
FunctionExecutionUnits	Function Execution Units	Count	Average	Function Execution Units
FunctionExecutionCount	Function Execution Count	Count	Average	Function Execution Count

Microsoft.Web/sites/slots

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
CpuTime	CPU Time	Seconds	Total	CPU Time
Requests	Requests	Count	Total	Requests
BytesReceived	Data In	Bytes	Total	Data In
BytesSent	Data Out	Bytes	Total	Data Out

METRIC	METRIC DISPLAY NAME	UNIT	AGGREGATION TYPE	DESCRIPTION
Http101	Http 101	Count	Total	Http 101
Http2xx	Http 2xx	Count	Total	Http 2xx
Http3xx	Http 3xx	Count	Total	Http 3xx
Http401	Http 401	Count	Total	Http 401
Http403	Http 403	Count	Total	Http 403
Http404	Http 404	Count	Total	Http 404
Http406	Http 406	Count	Total	Http 406
Http4xx	Http 4xx	Count	Total	Http 4xx
Http5xx	Http Server Errors	Count	Total	Http Server Errors
MemoryWorkingSet	Memory working set	Bytes	Total	Memory working set
AverageMemoryWorkingSet	Average memory working set	Bytes	Total	Average memory working set
AverageResponseTime	Average Response Time	Seconds	Average	Average Response Time
FunctionExecutionUnits	Function Execution Units	Count	Average	Function Execution Units
FunctionExecutionCount	Function Execution Count	Count	Average	Function Execution Count

Next steps

- Read about metrics in Azure Monitor
- Create alerts on metrics
- Export metrics to storage, Event Hub, or Log Analytics