# Monitoring and Alerting Framework for Azure Implementation Guide

## Overview

The Monitoring and Alerting Framework for Azure enables performance and health monitoring and alerting of a number of Azure resource types, including Virtual Machines, App Services (via Application Insights) and more. Notifications can be any of the notification actions supported by Azure Action Groups, including email notifications, phone calls notification, web hook invocations, etc

Monitoring and alerting of resources is controlled through a simple and flexible tag based approach, and alert threshold conditions are defined in JSON parameter files and stored in a repo.

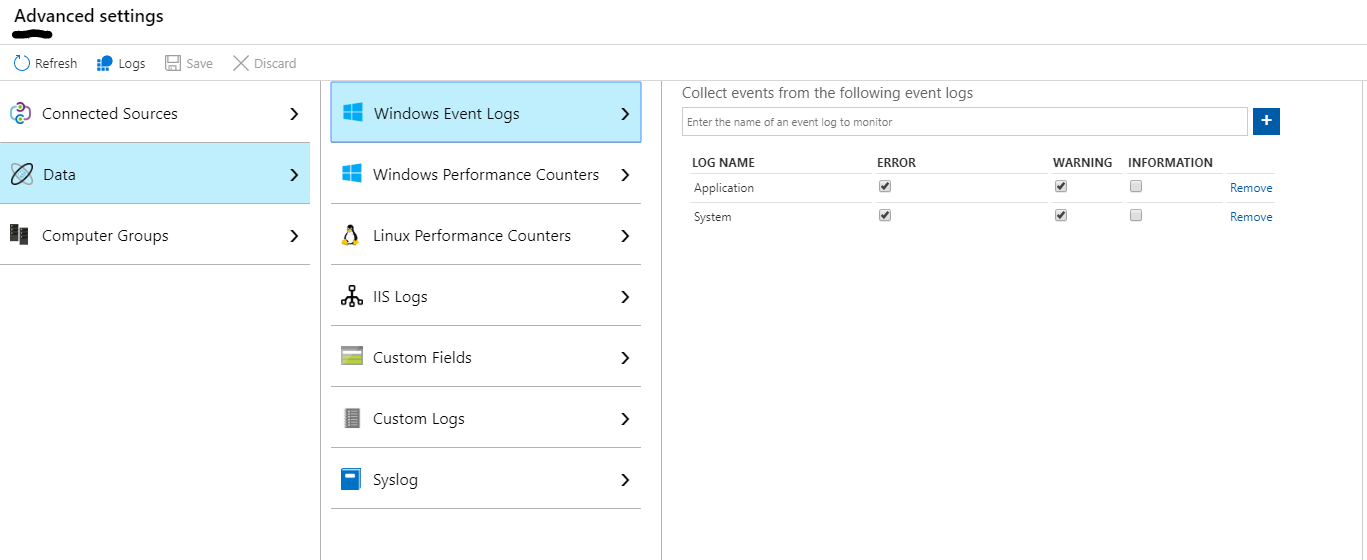
Three different types of alert rules are supported: log analytics query based, metric based, and activity log based.

## Currently supported performance/availability metrics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Resource Type** | **Alert Rule Set Name** | **Alert Rule Type** | **Alert Rule** | **Description** |
| Windows VM | windowsVMResourceHealthStandard | Activity Log | vm-unhealthy.json | Triggers on Azure VM health events |
| windowsVMStandard | Metric | cpu-percentage.json | Triggers on average percentage CPU above a certain threshold over a specified interval |
| windowsVMStandardLogAnalyticsQueries | Log Analytics Query | disk-slow-read.json | Triggers on Avg. Disk sec/Read above a certain threshold |
| disk-slow-write.json | Triggers on Avg. Disk sec/Write above a certain threshold |
| low-disk-space.json | Triggers on average % Free Space below a certain threshold |
| low-memory.json | Triggers on average Available Mbytes below a certain threshold |
| netlogon-failure.json | Triggers when a Windows Event ID 5719 is logged |
| sqlServerIaaSLogAnalyticsQueries | Log Analytics Query | blocked-processes.json | Triggers when the average number of SQL Server blocked processes is above a certain threshold for a specified number of consecutive intervals |
|  | buffer-cache-hit-ratio.json | Triggers when the SQL Server buffer cache hit ratio is above a certain threshold for a specified number of intervals |
|  | job-failure-package140.json | Triggers when a SQL Server job failure (event id 12291) is logged |
|  | job-failure.json | Triggers when a SQL Server job failure (event id 208) is logged |
|  | lock-timeouts.json | Triggers when the average SQL Server lock timeouts is above a certain threshold for a specified number of intervals |
|  | login-failure.json | Triggers when there’s any SQL Server login failure event logged in the specified time window |
|  | low-total-server-memory.json | Triggers when the average SQL Server Total Server Memory (KB) is below a certain threshold for a specified number of intervals |
|  | page-life-expectancy.json | Triggers when the average SQL Server Page life expectancy is below a certain threshold for a specified number of intervals |
|  | replication-failure.json | Triggers when there’s any SQL Server replication failure event logged in the specified time window |
|  | transaction-log-full.json | Triggers when there’s any SQL Server transaction log full event logged in the specified time window |
| Linux VM | linuxVmResourceHealthStandard | Activity Log | vm-unhealthy.json | Triggers on Azure VM health events |
| linuxVMStandard | Metric | cpu-percentage.json | Triggers on average percentage CPU above a certain threshold over a specified interval |
| linuxVMStandardLogAnalyticsQueries | Log Analytics Query | low-disk-space.json | Triggers on average % Free Space below a certain threshold |
| low-memory.json | Triggers on average Available Mbytes below a certain threshold |
| syslog-errors.json | Triggers on any error logged to the syslog |
| App Services Application Insights | applicationInsightsStandard | Metric | dependencies-duration.json | Triggers on dependency invocations that take longer than the specified threshold in the specified time window |
| failed-dependencies.json | Triggers when the number of failed dependency invocations exceeds a certain threshold within the specified time window |
| failed-requests.json | Triggers when the number of inbound failed HTTP requests exceeds a certain threshold within the specified time window |
| request-duration.json | Triggers when the request duration of inbound HTTP requests exceeds a certain threshold within the specified time window |
| requests-in-queue.json | Triggers when the queue depth of inbound HTTP requests exceeds a certain threshold within the specified time window |
| server-exceptions.json | Triggers when the number of logged server exceptions exceeds a certain threshold within the specified time window |
| App Service Plan | appServicePlanStandard | Metric | cpu-percentage.json | Triggers when the average CPU percentage used across the entire server farm exceeds a certain threshold over the specified time window |
|  |  |  | disk-queue-length.json | Triggers when the average disk queue length used across the entire server farm exceeds a certain threshold over the specified time window |
|  |  |  | http-queue-length.json | Triggers when the average HTTP queue length over the entire server farm exceeds a certain threshold over the specified time window |
|  |  |  | low-memory.json | Triggers when the average memory used across the entire server farm is below a certain threshold over the specified time window |
| Virtual Machine Scale Sets | vmScaleSetStandard | Metric | cpu-percentage.json | riggers when the average CPU percentage used across the scale set exceeds a certain threshold over the specified time window |

## Prerequisites

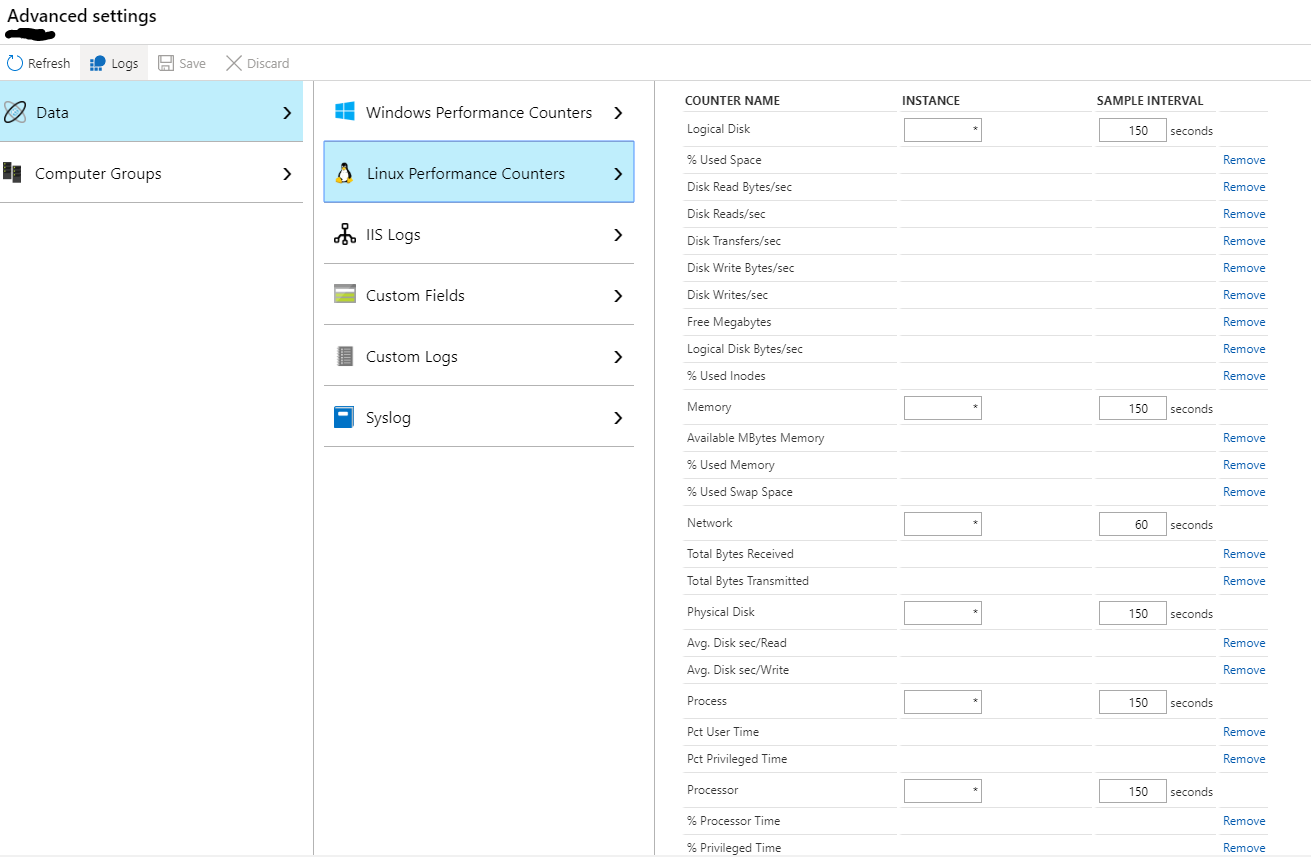
1. Create a log analytics workspace, if you haven’t already done so (see <https://docs.microsoft.com/en-us/azure/azure-monitor/learn/quick-create-workspace>)
2. Configure the log analytics workspace to log Error and Warning level Windows Application and System Event Logs.
   1. Navigate to your log analytics workspace, select the **Advanced settings** blade and select **Windows Event Logs**
   2. If necessary, add the Application and System event logs (Error and Warning levels)



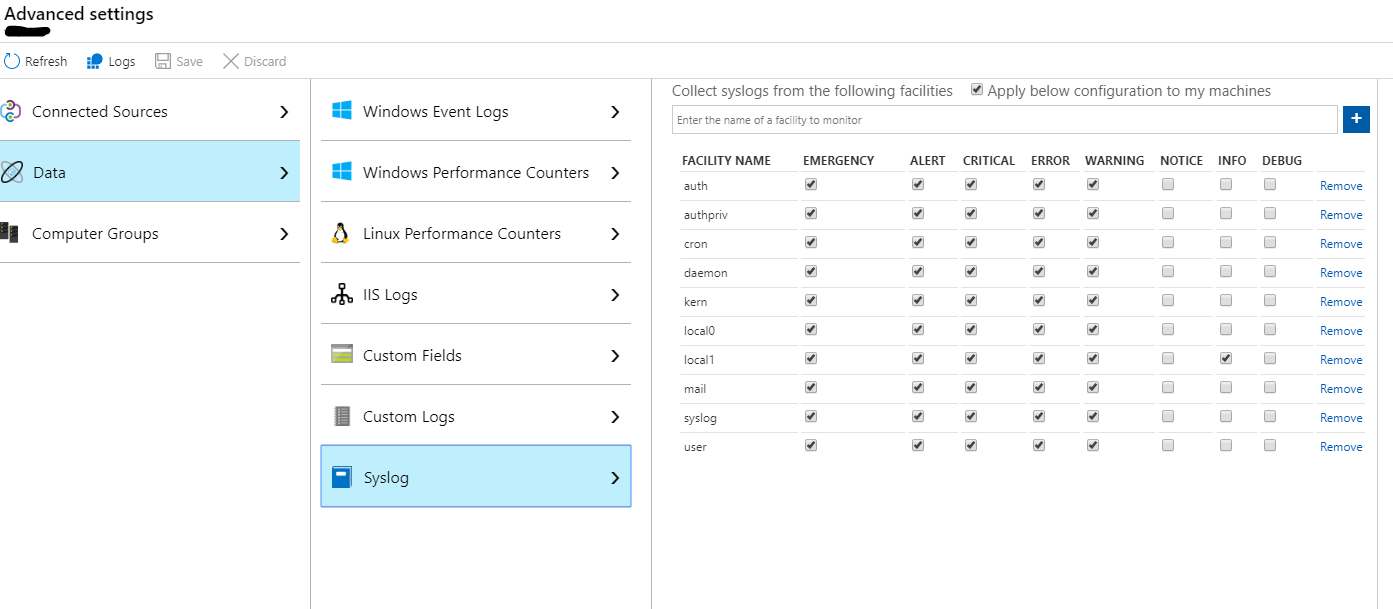
1. Select Windows Performance Counters and configure the log analytics workspace to collect the following Windows Performance Counters:

|  |
| --- |
| LogicalDisk(\*)\% Free Space |
| LogicalDisk(\*)\Avg. Disk sec/Read |
| LogicalDisk(\*)\Avg. Disk sec/Transfer |
| LogicalDisk(\*)\Avg. Disk sec/Write |
| LogicalDisk(\*)\Current Disk Queue Length |
| LogicalDisk(\*)\Disk Bytes/sec |
| LogicalDisk(\*)\Disk Read Bytes/sec |
| LogicalDisk(\*)\Disk Reads/sec |
| LogicalDisk(\*)\Disk Transfers/sec |
| LogicalDisk(\*)\Disk Write Bytes/sec |
| LogicalDisk(\*)\Disk Writes/sec |
| LogicalDisk(\*)\Free Megabytes |
| Memory(\*)\% Committed Bytes In Use |
| Memory(\*)\Available MBytes |
| Network Adapter(\*)\Bytes Received/sec |
| Network Adapter(\*)\Bytes Sent/sec |
| Network Interface(\*)\Bytes Total/sec |
| Network Interface(\*)\Packets Outbound Errors |
| Network Interface(\*)\Packets Received Errors |
| Processor(\_Total)\% Processor Time |
| SQLServer:Buffer Manager(\*)\Buffer cache hit ratio |
| SQLServer:Buffer Manager(\*)\Page life expectancy |
| SQLServer:General Statistics(\*)\Processes blocked |
| SQLServer:Locks(\*)\Lock Timeouts (timeout > 0)/sec |
| SQLServer:Memory Manager(\*)\Total Server Memory (KB) |
| System(\*)\Processor Queue Length |

1. Select Linux Performance Counters and configure the log analytics workspace to collect the following Linux Performance Counters:



1. Configure the log analytics workspace to collect the syslog events:



1. Enable logging to the log analytics workspace for any Windows and Linux VMs you will be monitoring (see <https://docs.microsoft.com/en-us/azure/azure-monitor/learn/quick-collect-azurevm>).

## Implementation steps overview

Details for each step can be found later in this document.

To setup monitoring and alerting of supported Azure resource types, you will need to:

1. Review the solution design (see Monitoring and Alerting Framework for Azure Solution Design.docx)
2. Create Azure Action Groups for each operations support team that should be notified when an alert condition is triggered.
3. Optionally fork the GitHub repo so that you can modify existing alert rule definitions or extend the set of alert rules that you want created. Whenever you fork, consider creating pull requests to contribute your changes to the open source project if you believe they will be useful to other GC departments and agencies.
4. Tag the resources that you want to monitor
5. Create the Azure Automation Account powershell runbook using the provided powershell code that is used to create the Azure Alert Rules based on the tagging you performed in step 3 and the alert rule definitions in your Git fork.
6. Optionally schedule the above runbook to run on a specific schedule, or alternatively use a manual approach to creating and updating Azure alert rules.

If you forked the repo, you can also modify existing alert rule definitions or create new alert rule definitions in your fork (see details below).

## Create Azure Action Groups for each of your operations support teams

Define Azure Action Groups for each of your operations support teams. These could be application dev support teams, infrastructure operations support teams, etc.

Follow the steps here to create one or more action groups: <https://docs.microsoft.com/en-us/azure/azure-monitor/platform/action-groups>

## Fork the GitHub repo where the alert rule definitions are stored (optional)

If you plan on modifying or adding alert rule definitions, you can fork the canada-ca/accelerators\_accelerateurs-azure repo (<https://github.com/canada-ca/accelerators_accelerateurs-azure.git>) and then modify existing alert rule definitions or add alert rule definitions.

In this case, you must also make sure to modify the automation runbook powershell script (described below) to point to your fork rather than the main repo.

### Repository structure

|  |  |
| --- | --- |
| **Folder** | **Description** |
| activity-log-alert-rules | Alert rule template parameter files for activity log based alerts |
| log-analytics-query-alert-rules | Alert rule template parameter files for log analytics query based alerts |
| metric-alert-rules | Alert rule template parameter files templates for metric based alerts |
| scripts | Powershell scripts to create the alert rules based on tagging |
| templates | ARM templates used by the powershell scripts to create the alert rules |

### Alert Rule template definitions

There are 3 Alert Rule templates defined in the templates folder:

1. metric-alert-rule-template.json which is a parameterized ARM template for creating metric alert rules
2. log-analytics-query-alert-rule-template.json which is a parameterized ARM template for creating log analytics query based alert rules
3. activity-log-alert-rule-template.json which is a parameterized ARM template for creating activity log based alert rules

### Alert Rule set definitions

The trigger conditions for alert rules are stored in JSON parameter files, and referenced when deploying the alert rule ARM template.

The alert rule JSON parameter files are organized into folders matching the name of the corresponding alert rule set, as shown below:

log-analytics-query-alert-rules (*alert rule type*)

sqlServerIaaSLogAnalyticsQueries (*alert rule set name*)

lock-wait-time.json (*alert rule parameter file*)

…

metric-alert-rules (*alert rule type*)

windowsVMStandard (*alert rule set name*)

low-memory-alert.json (*alert rule parameter file*)

low-disk-space.json (*alert rule parameter file*)

…

linuxVMStandard (*alert rule set name*)

low-memory-alert.json (*alert rule parameter file*)

low-disk-space.json (*alert rule parameter file*)

…

…

activity-log-alert-rules (*alert rule type*)

linuxVMResourceHealthStandard (*alert rule set name*)

vm-unhealthy.json (*alert rule parameter file*)

windowsVMResourceHealthStandard (*alert rule set name*)

vm-unhealthy.json (*alert rule parameter file*)

#### Modify or create an activity alert rule definition

If required, you can create a new alert rule set by creating a new folder under the activity-log-alert-rules folder. Make sure that the name of the alert rule set is unique.

If you created a new alert rule set, and if the alert rule set is a standard alert rule set (ie. will apply by default), make sure to modify the getStandardAlertRuleSet function of the create-alert-rules powershell script (and runbook) to map the alert rule set name to its associated resource type.

If you created a new alert rule set, and if the alert rule set is an extension alert rule set (ie. must be configured via alertRuleExtensions tag), make sure to modify the getResourceTypeForAlertRuleExtension function of the create-alert-rules powershell script (and runbook) to map the alert rule set name to its associated resource type.

To create new alert rule definitions, clone an existing alert rule definition JSON parameter file and then modify its parameter values; make sure you change the name of the alert (and file) to a name that does not conflict with other alert rule definitions.

Activity alert rule definitions are the simplest and involve the following parameters:

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| alertName | Name of the alert; must be unique |
| isEnabled | Set to true if the alert rule should be enabled |
| criteria | Condition which defines when the alert should be triggered. See <https://docs.microsoft.com/en-us/azure/azure-monitor/platform/alerts-activity-log> |

#### Modify or create a metric alert rule definition

If required, you can create a new alert rule set by creating a new folder under the metric-alert-rules folder. Make sure that the name of the alert rule set is unique.

If you created a new alert rule set, and if the alert rule set is a standard alert rule set (ie. will apply by default), make sure to modify the getStandardAlertRuleSet function of the create-alert-rules powershell script (and runbook) to map the alert rule set name to its associated resource type.

If you created a new alert rule set, and if the alert rule set is an extension alert rule set (ie. must be configured via alertRuleExtensions tag), make sure to modify the getResourceTypeForAlertRuleExtension function of the create-alert-rules powershell script (and runbook) to map the alert rule set name to its associated resource type.

To create new alert rule definitions, clone an existing alert rule definition JSON parameter file and then modify its parameter values; make sure you change the name of the alert (and file) to a name that does not conflict with other alert rule definitions.

Parameters:

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| alertName | Name of the alert; must be unique |
| alertDescription | Alert description |
| alertSeverity | Alert severity (0=Critical, 1=Error, 2=Warning, 3=Informational, 4=Verbose) |
| isEnabled | Set to true if the alert rule should be enabled |
| odataType | Odata type. See <https://docs.microsoft.com/en-us/azure/azure-monitor/platform/alerts-metric-create-templates> |
| criteria | Trigger criteria. See <https://docs.microsoft.com/en-us/azure/azure-monitor/platform/alerts-metric-create-templates> |
| windowSize | Period of time (from current) to be considered when evaluating the alert rule. Must be between five minutes and one day. ISO 8601 duration format. |
| evaluationFrequency | How often the metric alert is to be evaluated represented in ISO 8601 duration format |

#### Modify or create a log analytics query alert rule definition

If required, you can create a new alert rule set by creating a new folder under the log-analytics-query-alert-rules folder. Make sure that the name of the alert rule set is unique.

If you created a new alert rule set, and if the alert rule set is a standard alert rule set (ie. will apply by default), make sure to modify the getStandardAlertRuleSet function of the create-alert-rules powershell script (and runbook) to map the alert rule set name to its associated resource type.

If you created a new alert rule set, and if the alert rule set is an extension alert rule set (ie. must be configured via alertRuleExtensions tag), make sure to modify the getResourceTypeForAlertRuleExtension function of the create-alert-rules powershell script (and runbook) to map the alert rule set name to its associated resource type.

To create new alert rule definitions, clone an existing alert rule definition JSON parameter file and then modify its parameter values; make sure you change the name of the alert (and file) to a name that does not conflict with other alert rule definitions.

Parameters:

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| alertName | Name of the alert; must be unique |
| alertDescription | Alert description |
| alertSeverity | Alert severity (0=Critical, 1=Error, 2=Warning, 3=Informational, 4=Verbose) |
| isEnabled | Set to true if the alert rule should be enabled |
| location | Region where the alert rule should be created |
| query | Log analytics query string. See details below on how to build and format your query string. |
| queryType | Only **ResultCount** is supported |
| trigger | Trigger criteria (see <https://docs.microsoft.com/bs-latn-ba/azure/azure-monitor/platform/alerts-unified-log#metric-measurement-alert-rules>) |
| schedule | frequencyInMinutes specifies how often the alert rule should be evaluated.  timeWindowInMinutes specifies the time period (from current) to be considered. |
| suppressTimeInMinutes | Number of minutes to suppress additional alerts for this alert rule |

To build the query string, follow the steps here <https://docs.microsoft.com/en-us/azure/azure-monitor/log-query/get-started-queries>. Once you’ve tested the query and ensure it works in log analytics, remove all carriage returns and escape all double quotes by prepending backslash (eg. “example” becomes \”example\”. For VM based log analytics queries, replace the VM name with the variable **SubstituteVMName**; when the script runs it will dynamically substitute the name of the VM for this variable.

#### Modify create-alert-rules script and runbook to point to your fork

You need to modify the configuration in the create-alert-rules powershell script to pull the alert rule definitions from your fork and not the main repo. Update the repoOwner and repoName to match that of your fork (see below):

# ----------------------- Configurable settings start ------------------------------

# Update these defaults if you forked the canada-ca/accelerators\_accelerateurs-azure repo

$repoOwner = "canada-ca"

$repoName = "accelerators\_accelerateurs-azure"

# Update the relative path from root of repo to the alert rule automation artifacts folder if you moved the folder

$alertRulesPathPrefix = "Quickstarts\MonitoringAndAlertingFramework"

# ----------------------- Configurable settings end --------------------------------

## Tag your resources for monitoring and alerting

The tags that are used are as follows:

1. The **Alert** tag, which when set to True enables alerting for an individual resource or all resources in a resource group.
2. The **AlertRuleExtensions** tag, which defines which alert rule extension sets that should be created for a particular resource in addition to the standard set of alert rules corresponding to the resource. For example, the sqlServerIaaSLogAnalyticsQueries extension set defines a set of alert rules for SQL Server performance metrics for the tagged resource.
3. The **AlertActions** tag, which defines which support teams (defined via Action Groups) should be notified for one or more alert rule sets. This tag can be defined at the individual resource level and/or the resource group level.
4. The **AlertRulesExclusions** tag, which defines any standard alert rule sets and/or individual alert rules that shouldn’t be created for this resource. This may be used in cases where a customization is required to a particular alert rule; the alert rule would be created as per normal on the initial run of the script, modified manually, and then excluded using this tag from further script runs to ensure that the customized configuration isn’t overwritten.

### Alert tag

The Alert tag can be defined at the resource group and/or individual resource level and is used to control whether monitoring and alerting for supported resource types is turned on or off.

#### Resource Group level

You can enable alerting for all supported resource types in a resource group by tagging the resource group with the **Alert** tag name set to a value of **True**.

You can also disable alerting for one or more resource in a resource group tagged with the **Alert** tag by tagging the resource with the **Alert** tag name set to a value of **False**.

#### Resource level

Tag each resource that you want to monitor with the tag name **Alert** and a value of **True**.

### AlertActions tag

The AlertActions tag can be defined at the resource group and/or individual resource level and defines which support teams (defined via Action Groups) should be notified for one or more alert rule sets.

The format of the Alert Actions tag value is as follows:

***JSON string of the following format:***

***{***

***“comma separated list of alert rule set names 1”: “comma separated list of action group names to be notified”,***

***“comma separated list of alert rule set names 2”: “comma separated list of action group names to be notified”,***

***…***

***}***

***In place of a list of alert rule set names, a wildcard (\* character) can be used to designate “All Alert Rule Sets”.***

The supported alert rule set names are as follows:

|  |  |
| --- | --- |
| **Azure Resource Type** | **Supported Alert Rule Set Names** |
| Azure Linux Virtual Machines | linuxVMStandard, linuxVMStandardLogAnalyticsQueries,linuxVMResourceHealthStandard |
| Azure Windows Virtual Machines | windowsVMStandard,windowsVMStandardLogAnalyticsQueries,windowsVMResourceHealthStandard, sqlServerIaaSLogAnalyticsQueries |
| Azure App Service Application Insights | applicationInsightsStandard |
| Azure Virtual Machine Scale Sets | vmScaleSetStandard |
| Azure App Service Plan | appServicePlanStandard |

In place of a list of alert rule set names, a wildcard (\* character) can be used to designate “All Alert Rule Sets”.

For example, an AlertActions tag value of **{“\*”: “Infrastructure Support Team”}** means the Action Group “Infrastructure Support Team” will be notified for any triggered alert for all resources within a resource group if defined at the resource group level, or for a particular resource if defined at the resource level.

#### Resource Group level

You can optionally tag a resource group with the AlertActions tag name.

AlertActions defined at the resource group level will be additive to those defined for an individual resource within the resource group.

#### Resource level

You can optionally tag a resource with the AlertActions tag name.

AlertActions defined at the resource level will be additive to those defined at the resource group level.

**N.B.: If alerting is enabled for a resource (via the Alert tag), then the AlertActions tag must be defined for at least one of the scopes (resource or resource group level).**

### AlertRuleExtensions tag

You can optionally tag individual resources with the **AlertRuleExtensions** tag, which will extend the alert rules beyond the standard sets.

The format of the **AlertRuleExtensions** is as follows:

***Comma separated list of alert rule set names***

For example, a value of **sqlServerIaaSLogAnalyticsQueries** will cause alert rules to be created for all alert rule parameter files in the **sqlServerIaaSLogAnalyticsQueries** folder.

### AlertRuleExclusions tag

You can optionally tag individual resources with the **AlertRuleExclusions** tag, which defines any standard alert rule sets and/or individual alert rules that shouldn’t be created for this resource.

This may be required in cases where a customization is required to a particular alert rule. In this scenario, the alert rule would be created as per normal on the initial run of the powershell script, then the alert rule would be modified manually, and then excluded using this tag from further script runs to ensure that the alert rule’s customized configuration isn’t overwritten.

The format of the AlertRulesExclusions is as follows:

***Comma separated list of alert rule set names and/or alert rule parameter file names.***

For example, a value of **windowsVMStandard,netlogon-failure.json** for a Windows VM will skip creating all alert rule parameter files under the **windowsVMStandard** folder, in addition to the **netlogon-failure** alert rule.

## Create an Azure Automation Account powershell runbook

Steps:

1. You need an Azure Automation Account (Azure serverless compute environment for running automated scripts); make sure the Automation Account you use has been created with a Run As account. Follow the steps outlined here if you do not already have an Automation Account: <https://docs.microsoft.com/en-us/azure/automation/automation-quickstart-create-account>

Also ensure that your Automation Account has the latest versions of the powershell modules installed; by default a newly created Automation Account will not have the latest powershell module versions. To update the versions of the installed powershell modules, follow the instructions provided here: <https://docs.microsoft.com/en-us/azure/automation/automation-update-azure-modules>

1. Create a new powershell runbook in your Azure Automation Account, as follows:
   1. In the Azure Portal, select the Runbooks blade from your Automation Account.
   2. Select “+ Create a runbook”
   3. Enter “create-alert-rules” for the runbook name and select “PowerShell” for the runbook type, then select “Create”
   4. Copy the source code from <https://github.com/canada-ca/accelerators_accelerateurs-azure/blob/master/Quickstarts/MonitoringAndAlertingFramework/scripts/create-alert-rules.ps1>

and paste it as the source code for the runbook.

* 1. If you forked the repo, modify the configurable settings at the beginning of the script as follows:
     1. Update the **repoOwner** to the name of your GitHub repo owner name
     2. If necessary, update the **repoName** to the name of your repo (if necessary)
     3. If you moved the path from the root of the repo to the location of the MonitoringAndAlertingFramework folder, set the **alertRulesPathPrefix** to the new relative path
  2. Click Save and then Publish.

## Running your Automation Account runbook to create alert rules

You can run the runbook you created previously on an as needed basis, or schedule it to run periodically. If you are rarely adding resources to be monitored, you may want to stick with the manual run approach.

To run the runbook manually:

1. In the Azure Portal, select the Runbooks blade from your Automation Account.
2. Select the **create-alert-rules** runbook
3. Select Start

To schedule the runbook to run periodically follow the steps outlined here to create a new schedule and link it to your runbook: <https://docs.microsoft.com/en-us/azure/automation/shared-resources/schedules>.