Comprehensive Monitoring of Commercial Solutions in Azure

By \_\_\_\_\_

This document addresses the many ways that a software vendor can collect data on solutions running in Azure, log those to a common location, alert on events, facilitate RCA, and enable historical analysis.

# 1.0.0 Scenarios

This document will address the following scenarios:

* Alerting (near-real-time)  
  Log an event and alert someone immediately.
* Root Cause Analysis (<1 hour)  
  Events that have been logged are searchable and can be correlated across multiple platforms.
* Historical Analysis (<24 hours)  
  Event data can be mined for historical significance, trends, etc.

# 2.0.0 Alerting

For certain events it is especially important for support staff to be alerted as soon as an issue has been detected. The process will be described below, but the following chart shows the expected latency for most events:

Azure Monitor does support some near-real-time metric alerts which follow this pattern instead:

## 2.1.0 Detect

The first thing that must happen is that an event must be detected, or the status of the selected service probed. There are a variety of methods that might be used.

### 2.1.1 Azure Status

Azure Status is very high-level, but it lets you know if one or more Azure services are experiencing problems in one or more regions. It is not personalized in any way and there is no REST API for this service, though there is an RSS feed that could be consumed programmatically. More details can be found here: <https://docs.microsoft.com/en-us/azure/service-health/azure-status-overview>.

It is unlikely that you would consume any information at this level. If Azure has service issues that don’t affect you, there isn’t much value.

### 2.1.2 Service Health

Service Health is a personalized version of Azure Status, if any service you are using is affected by a service disruption, planned or unplanned, you should see that information here. This information can feed into Azure Monitor to trigger alerts. More details can be found here: <https://docs.microsoft.com/en-us/azure/service-health/service-health-overview>.

### 2.1.3 Resource Health

Each individual resource in Azure has a health status that you can view in the portal or can be accessed via REST API: <https://docs.microsoft.com/en-us/rest/api/resourcehealth/>. The specific checks that are made are outlined here: <https://docs.microsoft.com/en-us/azure/service-health/resource-health-checks-resource-types>. More details can be found here: <https://docs.microsoft.com/en-us/azure/service-health/resource-health-overview>.

### 2.1.4 Azure Monitor

Telemetry such as metrics, activity logs, and diagnostic logs are all exposed in Azure Monitor. Azure Monitor itself does not produce telemetry, but it aggregates it into a single place and allows you to graph and alert on the data points it is aware of. There is a REST API that can extract information programmatically: <https://docs.microsoft.com/en-us/rest/api/monitor/>.

You will use Azure Monitor to configure all alerts. You could use the Azure Monitor REST API to access data points that need to be logged to Log Analytics.

### 2.1.5 Instance Metadata – Scheduled Events

The Instance Metadata service is accessible only from inside a VM. It allows a VM access to a set of Azure properties about itself. Specify for our monitoring purposes, it also allows the VM to become aware of scheduled events for itself and its peers (in the same Availability Set or Scale Set). Scheduled events will include when a VM will:

* Freeze
* Reboot
* Redeploy

More information can be found here: <https://docs.microsoft.com/en-us/azure/virtual-machines/linux/scheduled-events>.

You could optionally log Instance Metadata to Log Analytics if you wanted to ensure you captured all events related to specific VMs.

## 2.2.0 Log

The next thing that must happen as part of this scenario is that the events must be logged. There are 2 different event stores that will probably be used for this purpose: App Insights and Log Analytics.

App Insights provides the following features:

* Agents are SDKs
* One log store per application
* Instrumentation to capture:
  + Incoming/outgoing HTTP requests
  + System metrics such as CPU usage
  + Unhandled exceptions
  + Events from popular libraries
  + Custom events
* Smart detection of anomalous behavior
* Interactive investigation for failures and performance
* Composite application map (dependencies)
* Out-of-the-box visualizations for common scenarios
* Up to 90-day retention

Log Analytics:

* Agents are deployed to infrastructure
* A single log store across all platforms
* Native integration with many Azure services
* Solution packages allow the analysis of the data collected
* Up to 2 year retention

Both systems store log data, use the same query language, can be visualized, can be searched, etc. but while Log Analytics is very unopinionated in its handling of log data, App Insights is targeted at understanding the health and performance of your custom applications. App Insights is Log Analytics with a very tailored ecosystem of tools to make it easier for its intended purpose.

The recommendation is to use both: use App Insights when you are willing to instrument your applications and in exchange get very detailed information regarding their operation; use Log Analytics for your infrastructure and Azure services.

You will likely have several App Insights log stores and a single Log Analytics store. Section 3.2.0 addresses the ways to stitch those together into a single query experience.

### 2.2.1 App Insights SDKs

We offer native SDKs for:

* .NET
* .NET Core
* Node.js
* Java

In addition, we offer integration with Visual Studio App Center which allows for use in mobile applications for:

* Android
* iOS
* React Native
* UWP
* Xamarin
* macOS
* Cordova

And if there is no SDK suitable for your project, you can use the REST API: <https://dev.applicationinsights.io/>.

### 2.2.2 Log Analytics Agents

The following agents and methods are available to get data into Log Analytics (<https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-data-sources>):

* Windows VM
* Linux VM
* Azure Services
  + Application Gateways
  + Application Insights
  + Automation Accounts
  + Batch Accounts
  + Classic Cloud Services
  + Cognitive Services
  + Data Lake Analytics
  + Data Lake Store
  + Event Hub Namespace
  + IoT Hubs
  + Key Vault
  + Load Balancers
  + Logic Apps
  + Network Security Groups
  + Recovery Vaults
  + Search Services
  + Service Bus
  + Service Fabric
  + Azure SQL Database
  + Storage
  + Virtual Machines
  + Virtual Machine Scale Sets
  + Web Server Farms
  + Web Sites
* Collectd performance data
* Nagios and Zabbix alerts
* IIS logs
* Custom logs

If there is not an agent to collect the data you want, you can use the REST API: <https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-data-collector-api>.

### 2.2.3 Log Analytics Custom

If there are no suitable agents to collect the data you wish to collect, you can simply write any arbitrary information to the REST API by following: <https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-data-collector-api>.

## 2.3.0 Dispatch

Some of the logging methods described above batch the log data locally before dispatching to the Azure service. The following agents allow for a variation of batch and dispatch mechanism:

* App Insights All SDKs
* Log Analytics Windows VM agent
* Log Analytics Linux VM agent
* Log Analytics Classic Cloud Services
* Log Analytics Service Fabric
* Log Analytics IIS Logs

Provided there are no connectivity issues, the first three scenarios are dispatched as they are accumulated. You can expect any single event to show up within 15 minutes. For the other services, please refer to: <https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-azure-storage-iis-table>.

The most common reason that data stops flowing into the Log Analytics store is that the daily limit for data volume has been exceeded. This article discusses the typical issues and how to detect them: <https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-faq#q-how-do-i-troubleshoot-if-log-analytics-is-no-longer-collecting-data>. In addition, you can monitor the usage: <https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-usage>.

## 2.4.0 Index

Once an event arrives in either App Insights or Log Analytics, it must be indexed to be available in the query engine. This process happens automatically but under normal operation can take up to 15 minutes to complete.

Recent metrics from the Log Analytics team show that 50% of events are indexed in under 7 minutes and 97% of events are indexed in under 15 minutes.

## 2.5.0 Alert

Once an event has been indexed, the Azure Monitor alerting engine could notify people as appropriate. There can be up to 5 minutes from when an event is indexed to when the alert is fired.

The Azure Monitor alerting process has been rewritten completely, the new engine is referred to as “new metric” or “new log alerts” whereas the old engine is referred to as “classic alerts”. The full process for creating alerts under the new system can be found here: <https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/monitor-alerts-unified-usage>.

When you define a new alert, you will:

1. Resource(s) to examine; you may specify
   1. All resources in a subscription
   2. All resources in a Resource Group
   3. All resources of a type across the subscription
   4. All resources of a type in a Resource Group
   5. A specific resource
   6. A query (arbitrary or saved) in Log Analytics
   7. A query (arbitrary or saved) in App Insights
2. Signal type
   1. Metric – CPU, network, disk performance, etc.
   2. Activity Log – Events such as restart, capture, deallocate, etc.
3. Logic
   1. Metric
      1. Condition – greater than, equal to, etc.
      2. Aggregation – total, average, min, max
      3. Threshold – value
      4. Period – 1 minute to 24 hours
      5. Frequency - 1 minute to 1 hour
   2. Logs
      1. Event Level – critical, info, etc. + all
      2. Status – failed, success, started + all
      3. Event Initiated By – filter to an account
4. Name
5. Description
6. Severity
7. Action Group
   1. Name
   2. Email, SMS, Push, Voice
   3. Webhook
   4. ITSM integration (SCSM, ServiceNow, Provance, Cherwell)
   5. Azure Automation Runbook

The frequency specifies how often the metric is sampled while the period specifies the timeframe over which those samples are considered (based on aggregation) to determine if they should trigger the alert. For instance, you might set a CPU sample **frequency** of 1 minute and a **period** of 5 minutes to alert if the CPU exceeds an **average** of 80%. That would mean for the last period there would be 5 samples, let’s say 75, 95, 88, 76, 71, which yields an average of 81% and the alert would be triggered.

Metrics currently support up to 2 criteria. If you use 2 criteria instead of 1, both must breach in the period for the alert to fire.

Different types of alerting and the number of times they are called can affect pricing as per: <https://azure.microsoft.com/en-us/pricing/details/monitor/>.

### 2.5.1 Query for Alerts

If the data you want to query is available in App Insights or Log Analytics, you can write an alert that runs a saved or arbitrary query.

### 2.5.2 Azure Monitor Alerts

Beyond querying for data in either type of log store, you can alert directly from data captured by Azure Monitor without storing the telemetry. The full list of supported services can be found here: <https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/monitoring-near-real-time-metric-alerts>.

A good example is CPU utilization, if you determine there is no value is storing that information, you might still want to alert if it gets too high.

Azure Monitor also has REST API that could be used to collect, process, and store metrics as desired: <https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/monitoring-rest-api-walkthrough>.

### 2.5.3 Log Analytics Queries

Data stored in Log Analytics or App Insights can be queried in Azure Monitor for alerting. You can run arbitrary queries or any saved queries. The period and frequency can be set between 5 min and 24 hours. Generally, you will alert based on having >0 results in the query, but there are other options.

You can also alert on the activity affecting the Log Analytics instance, for instance, you could alert if the workspace is deleted.

### 2.5.4 Near-Real-Time Metrics

A recent improvement to Azure Monitor allows certain metrics to be pushed through the alerting pipeline faster (the second pipeline in Section 2.0.0). The focus of this effort so far is performance metrics. There are several metrics available, but the 2 to look at are:

**Virtual Machines Metrics**

<https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/monitoring-near-real-time-metric-alerts#metrics-and-dimensions-supported>

**Log Analytics Performance Data**

<https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/monitoring-near-real-time-metric-alerts#support-for-oms-logs-as-metrics-for-alerting>

# 3.0.0 Root Cause Analysis

Your IT group or your customers might have received an alert or have some reason to believe systems are not functioning as they should. The appropriate staff should be able to query a log store for all relevant information and correlate those events across multiple platforms. If the log information were unavailable or incomplete, it could lead to the wrong diagnosis.

## 3.1.0 Query

The query engine for both App Insights and Log Analytics is Kustos. You can find documentation on the query language here: <https://docs.loganalytics.io/>, with a cheat sheet of common queries here: <https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-log-search-transition>.

If you are looking for information that is entirely self-contained in App Insights, you could use the prebuilt charts, reports, etc. in App Insights. If you need to search across data that may be in both App Insights and Log Analytics, you should search with Log Analytics and use one of the integration methods described in Section 3.2.0. The Log Analytics system should be your one-stop for logs and it can have a longer retention policy.

## 3.2.0 Integration

You will likely have quite a few App Insights log stores (one per application) and a single Log Analytics store. There are a few ways to ensure that when you can search inside your Log Analytics system you find data across all the various log stores.

### 3.2.1 Write to Both

If a specific log entry is relevant in both systems and if it has a retention requirement that is greater than 90 days, you could write the log from your application directly into both App Insights and Log Analytics.

### 3.2.2 Use Logic Apps

If you want to write events to App Insights only and then have a process to read some or all of that data out of that store and write them to Log Analytics, you could use Logic Apps and follow a process as described here:

<https://www.stefanroth.net/2017/10/10/azure-logic-app-send-data-from-application-insights-to-azure-log-analytics/>

### 3.2.3 App Insights Connector

If you don’t want to duplicate the data into Log Analytics and the retention requirement is 90 days or less, you could simply keep the data in your App Insights system and then query it from Log Analytics using the following connector: <https://blogs.technet.microsoft.com/msoms/2016/09/26/application-insights-connector-in-oms/>.

# 4.0.0 Historical Analysis

Having a long-term store of event data, like Log Analytics can provide is useful for historical analysis, such looking at trends or investigating potentially similar issues that occurred in the past.

## 4.1.0 Query

Using the techniques described in Section 3.2.1 or Section 3.2.2, you should have a complete history of relevant events in Log Analytics that can be queried.

## 4.2.0 Dashboard

Any query that you can run and aggregate in Log Analytics can be visualized in a dashboard that you can create. <https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-tutorial-dashboards>

## 4.3.0 Monitor Alerts

Using alerting, the appropriate people can be notified of relevant events. In addition, using Azure Monitor, you can see all alerts that have been raised, when they were raised, and who was notified.

## 4.4.0 Solutions

Log Analytics supports “solutions” that can mine useful data from the events and generally provide some pre-built visualizations as well as recommendations. The complete solution list can be found here: <https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-add-solutions>.

# 5.0.0 Implementation

The following section of this document describes an implementation of recommended logging coverage. These are ordered such that the highest value implementations are discussed first and you can continue down through the recommendations as there is a desire to go further.

## 5.1.0 Platforms

I recommend deploying:

* 1 Log Analytics instance
* “x” App Insights instances (1 per software solution)

I recommend setting the retention policy on Log Analytics to at least 1 year so you can build a reasonable store of historical data.

I recommend deploying “x” Logic Apps as per 3.2.2 to copy ***relevant*** data from each App Insight environment into the consolidated Log Analytics instance. Relevant data should be information that is useful to store for a longer retention period and might be used for the Section 4.0.0 Historical Analysis. If there are a lot of App Insight instances and it becomes complex to manage the Logic App implementation, you could also develop a solution using Azure Functions and the REST APIs. In either case, the execution of this copy operation on a routine schedule can be handled by Azure Scheduler.

## 5.2.0 Resource Health

I recommend:

* Create an Azure Function per subscription to read the status of all resources in the subscription as per: <https://docs.microsoft.com/en-us/rest/api/resourcehealth/availabilitystatuses/listbysubscriptionid>.
* Log any relevant events into Log Analytics.
* Setup an Alert in Azure Monitor to query for all negative status events in the past “x” minutes and alert whenever there is >0 results.
* Schedule the Function to run on a routine schedule (every “x” minutes) using Azure Scheduler.

## 5.3.0 Service Monitoring

I recommend:

* Implement App Insights for your solutions, including:
  + Find and diagnose run-time exceptions: <https://docs.microsoft.com/en-us/azure/application-insights/app-insights-tutorial-runtime-exceptions>.
  + Find performance issues: <https://docs.microsoft.com/en-us/azure/application-insights/app-insights-tutorial-performance>.
* Setup alerts in Azure Monitor by querying appropriate data from App Insights.

For micro-services based applications, particularly those that span multiple regions, I recommend using some tool for managing dependency chains, quality of service, preferred regions, etc. This information can be used to route customers to the most appropriate instance of a service. I am developing such a tool that you could adapt or there may be others.

* <https://github.com/plasne/Multi-Service-Availability>

## 5.4.0 Virtual Machines

I recommend:

* Install the VM agent on all VMs that are important for the delivery of your solutions: <https://docs.microsoft.com/en-us/azure/virtual-machines/linux/extensions-oms>.
* Add the Agent Health solution to Log Analytics as a way to track that all agents are reporting: <https://docs.microsoft.com/en-us/azure/operations-management-suite/oms-solution-agenthealth>.

## 5.5.0 HDInsight

These steps must be applied to each HDInsight instance. I recommend:

* Activate the logging to Log Analytics feature in HDInsight: <https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-hadoop-oms-log-analytics-tutorial>.

## 5.6.0 Data Factory

These steps only apply to ADFv2 and must be applied to each ADF instance. I recommend:

* Activate the logging to Log Analytics feature in Data Factory via REST: <https://docs.microsoft.com/en-us/azure/data-factory/monitor-using-azure-monitor>.

It is also possible to alert on Metrics for ADF, for example, who many failed or successful pipeline runs there were over a period of time. I don’t imagine this is very useful, but the alerting can be configured in Azure Monitor.

## 5.7.0 Azure Container Service

For each instance of ACS, I recommend:

* Install the Containers solution for Log Analytics: <https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-containers>.

For AKS instances, there is a more specific set of instructions:

* <https://docs.microsoft.com/en-us/azure/aks/tutorial-kubernetes-monitor>

## 5.8.0 App Insights Connector

The Historical Analysis use-case should be covered by copying relevant data from App Insights to Log Analytics, however for the RCA use-case, it may be beneficial to search all logs (not just those kept for longer retention) across your App Insights and Log Analytics instances. For this you should install the App Insight connector for Log Analytics as appropriate: <https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-app-insights-connector>.

## 5.9.0 Service Health

This is of relatively low value if you are already capturing the health of individual resources, but you could imagine a case where there is a reported diminished capability for a service that covers functionality not examined by Resource Health. For example, imagine the Virtual Machine control plane is not working in a region, and perhaps you cannot perform actions such as shutdown, restart, etc. – the ability to perform these actions is not instrumented by Resource Health and so you would have diminished capability even without a disruption of availability.

I recommend alerting on Service Health:

* <https://docs.microsoft.com/en-us/azure/monitoring-and-diagnostics/monitoring-activity-log-alerts-on-service-notifications>

If there is value in logging the events to Log Analytics:

* Add an alert that is a webhook that calls a Logic App or Azure Function.
* Build a Logic App or Azure Function with a web endpoint to receive the message and send the event to Log Analytics.

## 5.10.0 Instance Metadata – Scheduled Events

Planned maintenance is surfaced in Resource Health, so there is low value in recording the information from Instance Metadata, but it does provide one additional source of significant events very close to the target (meaning the VM being affected by a maintenance or failure event could report its own experience). The challenge with capturing this information is that a reporting mechanism would need to be deployed to every single VM.

If you are going to log this information:

* Use a configuration management solution to deploy a script to all your VMs.
* The script should execute every “x” minutes where “x” is less than 10 minutes. The shortest notification period for an event is 10 minutes.
* The script should call the Instance Metadata service on the local system.
* If a pending event is discovered for the VM itself, then it should make a REST API call to Log Analytics and record the event.
* If you desire alerts, Azure Monitor can be configured to run a query against Log Analytics and look for these events.

Information on the service can be found here: <https://docs.microsoft.com/en-us/azure/virtual-machines/linux/scheduled-events>.

## 5.11.0 Solution Packages

There are quite a few solution packages that can analyze events in Log Analytics, you could review and implement these as appropriate: <https://docs.microsoft.com/en-us/azure/log-analytics/log-analytics-add-solutions>.