


Monitoring application health for reliability in Azure

04/10/2019 • 8 minutes to read • Contributors 

In this article

[Instrumentation](#)
[Collection and storage](#)
[Analysis and diagnosis](#)
[Visualization and alerts](#)
[Health probes and check functions](#)
[Long-running workflow failures](#)
[Application logs](#)
[Remote call statistics](#)
[Transient exceptions and retries](#)
[Early warning system](#)
[Subscription and service limitations](#)
[Third-party services](#)
[Next steps](#)

Monitoring and diagnostics are crucial for resiliency. If something fails, you need to know *that* it failed, *when* it failed — and *why*.

Monitoring is not the same as *failure detection*. For example, your application might detect a transient error and retry, avoiding downtime. But it should also log the retry operation so that you can monitor the error rate to get an overall picture of application health.

Think of the monitoring and diagnostics process as a pipeline with four distinct stages: Instrumentation, collection and storage, analysis and diagnosis, and visualization and alerts.

Instrumentation

It's not practical to monitor your application directly, so instrumentation is key. A large-scale distributed system might run on dozens of virtual machines (VMs), which are added and removed over time. Likewise, a cloud application might use a number of data stores and a single user action might span multiple subsystems.

Provide rich instrumentation:

- For failures that are likely but have not yet occurred, provide enough data to determine the cause, mitigate the situation, and ensure that the system remains available.
- For failures that have already occurred, the application should return an appropriate error message to the user but should attempt to continue running, albeit with reduced functionality.

Monitoring systems should capture comprehensive details so that applications can be restored efficiently and, if necessary, designers and developers can modify the system to prevent the situation from recurring.

The raw data for monitoring can come from a variety of sources, including:

- Application logs, such as those produced by the [Azure Application Insights](#) service.
- Operating system performance metrics collected by [Azure monitoring agents](#).
- [Azure resources](#), including metrics collected by Azure Monitor.

- [Azure Service Health](#), which offers a dashboard to help you track active events.
- [Azure AD logs](#) built into the Azure platform.

Most Azure services have metrics and diagnostics that you can configure to analyze and determine the cause of problems. To learn more, see [Monitoring data collected by Azure Monitor](#).

Collection and storage

Raw instrumentation data can be held in various locations and formats, including:

- Application trace logs
- IIS logs
- Performance counters

These disparate sources are collected, consolidated, and placed in reliable data stores in Azure, such as Application Insights, Azure Monitor metrics, Service Health, storage accounts, and Azure Log Analytics.

Analysis and diagnosis

Analyze data consolidated in these data stores to troubleshoot issues and gain an overall view of application health. Generally, you can [search for and analyze](#) the data in Application Insights and Log Analytics using Kusto queries or view preconfigured graphs using [management solutions](#). Or use Azure Advisor to view recommendations with a focus on [resiliency](#) and [performance](#).

Visualization and alerts

Present telemetry data in a format that makes it easy for an operator to notice problems or trends quickly, such as a dashboard or email alert.

Get a full-stack view of application state by using [Azure dashboards](#) to create a consolidated view of monitoring graphs from Application Insights, Log Analytics, Azure Monitor metrics, and Service Health. And use [Azure Monitor alerts](#) to create notifications on Service Health, resource health, Azure Monitor metrics, logs in Log Analytics, and Application Insights.

For more information about monitoring and diagnostics, see [Monitoring and diagnostics](#).

Health probes and check functions

The health and performance of an application can degrade over time, and degradation might not be noticeable until the application fails.

Implement probes or check functions, and run them regularly from outside the application. These checks can be as simple as measuring response time for the application as a whole, for individual parts of the application, for specific services that the application uses, or for separate components.

Check functions can run processes to ensure that they produce valid results, measure latency and check availability, and extract information from the system.

Long-running workflow failures

Long-running workflows often include multiple steps, each of which should be independent.

Track the progress of long-running processes to minimize the likelihood that the entire workflow will need to be rolled back or that multiple compensating transactions will need to be executed.



Tip

Monitor and manage the progress of long-running workflows by implementing a pattern such as [Scheduler Agent Supervisor](#).

Application logs

Application logs are an important source of diagnostics data. To gain insight when you need it most, follow best practices for application logging.

Log data in the production environment

Capture robust telemetry data while the application is running in the production environment, so you have sufficient information to diagnose the cause of issues in the production state.

Log events at service boundaries

Include a correlation ID that flows across service boundaries. If a transaction flows through multiple services and one of them fails, the correlation ID helps you track requests across your application and pinpoints why the transaction failed.

Use semantic (structured) logging

With structured logs, it's easier to automate the consumption and analysis of the log data, which is especially important at cloud scale. Generally, we recommend storing Azure resources metrics and diagnostics data in a Log Analytics workspace rather than in a storage account. This way, you can use Kusto queries to obtain the data you want quickly and in a structured format. You can also use Azure Monitor APIs and Azure Log Analytics APIs.

Use asynchronous logging

Synchronous logging operations sometimes block your application code, causing requests to back up as logs are written. Use asynchronous logging to preserve availability during application logging.

Separate application logging from auditing

Audit records are commonly maintained for compliance or regulatory requirements and must be complete. To avoid dropped transactions, maintain audit logs separately from diagnostic logs.

Remote call statistics

Track and report remote call statistics in real time and provide an easy way to review this information, so the operations team has an instantaneous view into the health of your application. Summarize remote call metrics, such as latency, throughput, and errors in the 99 and 95 percentiles.

Transient exceptions and retries

Track the number of transient exceptions and retries over time to uncover issues or failures in your application's retry logic. A trend of increasing exceptions over time may indicate that the service is having an issue and may fail. For more information, see [Retry service specific guidance](#).

Early warning system

Monitor your application for warning signs that might require proactive intervention. Tools that assess the overall health of the application and its dependencies help you to recognize quickly when a system or its components suddenly become unavailable. Use them to implement an early warning system.

1. Identify the key performance indicators of your application's health, such as transient exceptions and remote call latency.
2. Set thresholds at levels that identify issues before they become critical and require a recovery response.
3. Send an alert to operations when the threshold value is reached.

Consider [Microsoft System Center 2016](#) or third-party tools to provide monitoring capabilities. Most monitoring solutions track key performance counters and service availability. [Azure resource health](#) provides some built-in health status checks, which can help diagnose throttling of Azure services.

Subscription and service limitations

Azure subscriptions have limits on certain resource types, such as number of resource groups, cores, and storage accounts. To ensure that your application doesn't run up against Azure subscription limits, create alerts that poll for services nearing their limits and quotas.

Address the following subscription limits with alerts.

Individual services

Individual Azure services have consumption limits on storage, throughput, number of connections, requests per second, and other metrics. Your application will fail if it attempts to use resources beyond these limits, resulting in service throttling and possible downtime.

Depending on the specific service and your application requirements, you can often stay under these limits by scaling up (choosing another pricing tier, for example) or scaling out (adding new instances).

Azure storage scalability and performance targets

Azure allows a maximum number of storage accounts per subscription. If your application requires more storage accounts than are currently available in your subscription, create a new subscription with additional storage accounts. For more information, see [Azure subscription and service limits, quotas, and constraints](#).

If you exceed Azure storage scalability and performance targets, your application will experience storage throttling. For more information, see [Azure Storage scalability and performance targets](#).

Scalability targets for virtual machine disks

An Azure infrastructure as a service (IaaS) VM supports attaching a number of data disks, depending on several factors, including the VM size and the type of storage account. If your application exceeds the scalability targets for virtual machine disks, provision additional storage accounts and create the virtual machine disks there. For more information, see [Scalability and performance targets for VM disks on Windows](#).

VM size

If the actual CPU, memory, disk, and I/O of your VMs approach the limits of the VM size, your application may experience capacity issues. To correct the issues, increase the VM size. VM sizes are described in [Sizes for virtual machines in Azure](#).

If your workload fluctuates over time, consider using virtual machine scale sets to automatically scale the number of VM instances. Otherwise, you need to manually increase or decrease the number of VMs. For more information, see the [virtual machine scale sets overview](#).

Azure SQL Database

If your Azure SQL Database tier isn't adequate to handle your application's Database Transaction Unit (DTU) requirements, your data use will be throttled. For more information on selecting the correct service plan, see [Azure SQL Database purchasing models](#).

Third-party services

If your application has dependencies on third-party services, identify how these services can fail and what effect failures will have on your application.

A third-party service might not include monitoring and diagnostics. Log calls to these services and correlate them with your application's health and diagnostic logging using a unique identifier. For more information on proven practices for monitoring and diagnostics, see [Monitoring and diagnostics guidance](#).

Next steps

Plan for disaster recovery