Building real-time monitoring and observable systems for Media

# Overview

At Microsoft, we get the opportunity to work with customers that lie on a large spectrum when it comes to scale, the types of technologies they deploy, and the environments they operate in.

However, we’ve learned that customers are universally interested in understanding how their systems and applications behave, and in case of failures, errors, or outages their ability—or equally their inability—to take action.

At [IBC 2022](https://cloudblogs.microsoft.com/industry-blog/media/2022/09/09/ibc-conference-transforming-media-and-entertainment-with-the-microsoft-cloud/), we showcased what monitoring and observability mean when it comes to an entire video workflow from transcoding, distribution to CDN, and ultimately consumption on a wide array of end devices that include mobile phones, TVs, laptops/desktops, streaming sticks, etc. We also demonstrated the ability to build a low-cost fault-tolerant system that scales horizontally on Azure using the toolbox of managed Azure services where the complexity of management and scaling is abstracted away from end consumers.

Below we share the principles of building real-time monitoring and observable systems and conclude with a discussion on Azure AI to detect anomalies.

# Monitoring vs. Observability

Observability has become a hot topic in the world of monitoring. Although monitoring and observability are related and complement each other, they are conceptually different.

*Observability* refers to the probes/instrumentation/agents used to get some observable data. In contrast, *monitoring* typically refers to dashboards you build or alerts you configure based on an understanding of the data. The key thing to highlight in *monitoring* is that you understand the data and any problems that might arise. As a result, you can build systems that can catch such errors. For example, if CPU utilization exceeds 90%, then the monitoring system should dispatch a notification.

Graphical user interface, application

Description automatically generated

**Figure 1** – Example of monitoring dashboard showing quality-of-experience and CDN metrics.

In dynamic environments, we may have thousands of microservices scaling up and down, or in a video-on-demand (VOD) setting, we may have heterogeneous clients (mobile, desktop, or TV) using different network connections (Wi-Fi vs cellular). Each combination of dimensions for the *observable* data points can help to establish a baseline of a steady-state system. Furthermore, if there are any “nonconformities” to this baseline, then we must notify systems of such *anomalies*. As an example, a video streaming experience on a 3G cellular connection will have a vastly different baseline profile compared to a wired fiber-optic internet connection. The main idea is that a system needs to be able to associate context with data points. If this sounds like a challenging problem, it is. Luckily, there are easy-to-implement out-of-the-box solutions which are discussed later in this post.

The key thing to understand in distributed, cloud-native environments is that most problems are neither known nor monitored. Consequently, organizations can leverage Artificial Intelligence (AI) to make sense of the data and automate the manual process of configuration and alerting. Later we will learn how *Metrics Advisor* can help us with this task.

# Architecture

Timeline

Description automatically generated

**Figure 2** – Architecture diagram of an observable system. Raw telemetry is streamed to Azure via HTTP and connectors to Blob Storage. The raw telemetry is processed, transformed, and normalized and saved in Data Explorer for analysis. Systems like Grafana and Metrics Advisor can read data from Data Explorer and show insights to end customers.

A few key components are shown in *Figure 2*. At a high level we need to implement:

1. **Instrumentation** – These are the probes or agents we install in our systems to monitor data. These can come in a variety of shapes and forms. For example, in a video-on-demand streaming platform, a company may leverage open-standards [dash.js](https://github.com/Dash-Industry-Forum/dash.js/) to collect Quality-of-Experience (QoE) metrics of their end consumers.
2. **Ingestion** – This raw telemetry can come directly from end clients via HTTP calls or be uploaded by third-party systems to persistent storage and data lakes, e.g., *Azure Blob Storage*. Azure Blog Storage supports the ability to invoke an Azure Function as soon as a new file is uploaded. This trigger mechanism can be leveraged to move raw telemetry from data lakes to structured data warehouses.
3. **Transformation and persistence** – A transformation system may be needed to normalize the data. An *Azure Function* app transforms the data if needed and subsequently writes to *Azure Data Explorer*. Azure Data Explorer is ideal for big data analytics as it is designed to have high performance and throughput on large data sets.
4. **Monitoring** – *Azure Managed Grafana* out-of-the-box supports integration with *Azure Data Explorer*. You can use the click-and-drop features of Grafana to quickly build dashboards and charts. Grafana is a good fit for media monitoring, as it handles sub-minute refreshing of dashboard tiles and can also be used for alerting.
5. **Anomaly detection** – Often in observable systems, we may not even know what a baseline looks like. For this, we can leverage AI. Technologies like *Azure Metrics Advisor* employ an ensemble of machine learning algorithms to automatically understand and detect anomalies based on your time-series data.

## Azure Functions versus Data Factory/Synapse

You may be wondering why we choose a *Function App* over technologies like *Data Factory* and *Synapse*. The short of it is that *Data Factory* and *Synapse* are not able to provide near real-time monitoring capabilities. *Data Factory* and *Synapse* both have a minimum lag of ~5 minutes from the time of ingestion to persistence. However, if that lag is acceptable, then *Data Factory* and *Synapse* provide industry-leading tools/workspace to build ETL workflows with the ability to track and retry jobs from a graphical interface.

# Scaling

Depending on the scale and frequency of incoming requests, you will find the *Function App* to be a chokepoint. There are two major factors for this:

1. **Cold start** – This is a phenomenon of serverless executions and refers to the scheduling and set-up time required to spin up an environment until the function first starts to execute. In the worst case, this can be on the order of a few seconds.
2. **Frequency of requests** – Imagine you have 1000 HTTP requests but only a single-threaded server to handle these incoming requests. You will not be able to service all the 1000 HTTP requests concurrently. To serve these requests in a timely fashion, you need to deploy more servers, i.e., scale horizontally.

In both cases, we recommended Premium or Dedicated SKUs to 1) eliminate cold start, and 2) handle requirements for concurrent requests by scaling up/down the number of servicing virtual machines. More details on SKUs can be found [here](https://learn.microsoft.com/en-us/azure/azure-functions/functions-scale).

# Redundancy

Your business-critical application needs to run despite a disruptive event like an Azure region outage. In this case, there are two strategies for building redundancy into your system:

* **Active/Active** – Your code and functions are operating in a duplicate manner and either system can take over in case of a failure.
* **Active/Standby** – In this configuration, only one node is active/primary while the other one is waiting to take over in case the primary node goes down.
* **Mixed** – In this strategy, you have some components/services in Active/Active configuration and some in Active/Standby.

It is important to note that not all Azure services have built-in redundancy. For example, Azure Functions run a function app only in a specific region. Depending on how the Function is triggered (HTTP versus pub/sub), there are different strategies discussed in detail [here](https://learn.microsoft.com/en-us/azure/azure-functions/functions-geo-disaster-recovery).

While the ingestion and transformation Function App can run in active/active mode, for Azure Data Explorer, you can run in both [active/active and active/standby configurations](https://techcommunity.microsoft.com/t5/azure-data-explorer-blog/azure-data-explorer-and-business-continuity/ba-p/1332767).

The last piece of monitoring is Grafana. We recommend setting up Grafana in each region where you have Data Explorer deployed to ensure redundancy.

# Anomaly Detection

“No data is good data unless it is actionable”.

When it comes to applications of observability, we’d like our systems to 1) identify patterns—whether seasonal/cyclical, i.e., weekly or daily pattern—2) establish a baseline of expected behavior, and 3) raise an alert (referred to as *anomaly*) if some data point(s) fall out of the expected baseline. *Figure 3* shows an example of anomaly detection in action on time series data.

Graphical user interface, table

Description automatically generated

**Figure 3** – An example of a *Slow Start* anomaly that is triggered in Metrics Advisor due to an outage of ASN.

As you can imagine, time series data behaves differently depending on the scenario. For example, the number of daily visitors to an e-commerce site is vastly different compared to the frequency of vibrations for a conveyor-belt component. *Metrics Advisor* looks at each dataset and automatically selects the best algorithm from the model pool for high accuracy.

More details about the types of algorithms and the classification process can be found [here](https://techcommunity.microsoft.com/t5/ai-customer-engineering-team/introducing-azure-anomaly-detector-api/ba-p/490162).

The code samples we provide set up the necessary plumbing for you to ingest your data into Metrics Advisor.

## Media specific metrics

In conversations with media customers, some of the important metrics include quality-of-experience (QoE) metrics like *slow start*, *low quality*, *stalls/buffering*, *player errors*; CDN-specific metrics like *time-to-first-byte* (TTFB); as well content engagement, e.g., *user location*, *watched time*, and *streaming clients*. However, Metrics Advisor is not just limited to the aforementioned metrics. At IBC 2022, we also demonstrated novel use cases like misuse of assets and discovery of new markets, e.g., asset premieres in region-A but is being watched in region-B; QoE characteristics based on region; and outages. You can view [this](https://ibcdemo2022website.azurewebsites.net/IBC_MSFT_deck.pdf) deck to learn more about our work.

# Next Steps & Code Samples

Our team has prepared end-to-end code samples to 1) bring up the infrastructure using Bicep, and 2) application-level code for ingestion via HTTP and Blob trigger into Data Explorer.

Code samples can be found here: <https://github.com/Azure-Samples/real-time-monitoring-and-observability-for-media>

Please reach out to us with any additional questions or feedback.