Building real-time monitoring and observable systems for Media

# Abstract

Customers often deploy varied and large-scale technologies to solve their business problems. These systems and end-user devices generate large sets of telemetry data.

This architecture highlights a use case in the Media industry and showcases solutions that provide real-time monitoring and observability of systems and end-user device telemetry data. The nature of media streaming for live and video-on-demand playback requires near real-time identification and response to application issues. To support this, a massive telemetry set must be collected that requires scalable architecture. Once that data is collected, additional analysis approaches, such as AI and anomaly detection become necessary to efficiently identify problems across such a massive data set.

# Problem Statement

When large-scale technologies are deployed to support, the system and end-user devices that interact with them generate massive sets of telemetry data. In traditional scenarios, these data are analyzed via a data warehouse system to provide insights that can be used to support management decisions. This approach may work in some scenarios, but it is not responsive enough for streaming media use cases. To solve this problem, real-time insights are required for the telemetry data generated from monitoring servers, networks, and end-user devices that interact with them. Monitoring systems that can catch failures and errors are common but real-time are difficult and that is the focus of this blog.

# Solution Summary

In a live stream or video-on-demand (VOD) setting, telemetry data are generated from systems and heterogeneous clients (mobile, desktop, or TV). The solution entails taking raw data and being able to associate context with the data points (e.g. dimensions such as geography, end-user operating system, content ID, CDN provider). The raw telemetry is collected, transformed, , and saved in Azure Data Explorer for analysis. Consequently, organizations can leverage Artificial Intelligence (AI) to make sense of the data and automate the manual processes of observation and alerting. Systems like Grafana and Metrics Advisor are used to read data from Data Explorer and show interactive dashboards and trigger alerts.

# Architecture



**Figure 1** – 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, 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 1*. 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 extremely 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** – The Grafana dashboard provides support for “eyes on glass” in the NOC. However, with this large data set and user base spread across a vast array of geos and devices, it becomes inefficient for manual identification of issues using charts and alert rules with hard-coded thresholds. To address 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. In addition, the Kusto data platform has built in anomaly detection functions that account for seasonality and baseline trends in the data.

# Components

* [**Azure Data Explorer**](https://azure.microsoft.com/en-us/products/data-explorer/#overview) is largely a managed data analytics service for real-time analysis on large volumes of data. In our use case, we have employed Azure Data Explorer for storing and querying large datasets. Azure Data Explorer is a great tool when dealing with any large dataset that requires high speed and throughput of data retrieval.
* [**Azure Blob Storage**](https://azure.microsoft.com/en-us/services/storage/blobs/) is used to hold raw telemetry. This may come from your systems or from third-party services. This data can be treated as transient if you do not need to perform additional analysis further down the road. The data from Blob storage will be ingested into Azure Data Explorer clusters.
* [**Azure Event Grid**](https://azure.microsoft.com/en-us/products/event-grid/) is an event delivery system with reliable messaging. It is used to listen to events published by Azure Blob Storage. Azure Storage events allow applications to react to events, such as the creation and deletion of blobs. An Azure Function subscribes to events or topics published by Event Grid.
* [**Azure Event Hubs**](https://azure.microsoft.com/en-us/products/event-hubs/) is a real-time data ingestion service that allows you to ingest millions of events per second from any source. Event Hubs represents the "front door" for an event pipeline, often called an event ingestor in solution architectures. An event ingestor is a component or service that sits between event publishers and event consumers to decouple the production of an event stream from the consumption of those events.
* [**Azure Functions**](https://learn.microsoft.com/en-us/azure/azure-functions/functions-overview) is a serverless solution used to parse data via HTTP requests and blob endpoints and store into the Azure Data Explorer cluster.
* [**Azure Managed Grafana**](https://azure.microsoft.com/en-us/services/managed-grafana/) is used to connect easily to Azure Data Explorer and generate views and dashboards to visualize telemetry data. Azure Managed Grafana integrates with Azure Active Directory which allows you to control which users can use a Grafana instance and/or view particular dashboards.
* [**Azure Metrics Advisor**](https://azure.microsoft.com/en-us/products/metrics-advisor/) is a part of Azure Applied AI Services that uses AI to perform data monitoring and anomaly detection in time series data. This service automates the process of applying models to your data, and provides a set of APIs and a web-based workspace for data ingestion, anomaly detection, and diagnostics - without needing to know machine learning.

# Considerations

## 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 or CDN 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. To load balance Grafana access across more than one region, we recommend [Azure Traffic Manager](https://learn.microsoft.com/en-us/azure/traffic-manager/traffic-manager-overview).

## 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.

# Contributors

This article is maintained by Microsoft. It was originally written by the following contributors:

Principal authors:

* [Uffaz Nathaniel](https://www.linkedin.com/in/uffaz-nathaniel-85588935/) | Principal Software Engineer
* [John Hauppa](https://www.linkedin.com/in/johnhauppa/) | Senior Technical Program Manager

Other contributors:

* [Dilmurod Makhamadaliev](https://www.linkedin.com/in/dilmurod-makhamadaliev/) | Software Engineer
* [Omeed Musavi](https://www.linkedin.com/in/omusavi/)| Principal Software Engineer Lead
* [Ayo Mustapha](https://www.linkedin.com/in/ayo-mustapha/) | Principal Technical Program Manager

# Next steps

* Link to associated blog post: <insert link>
  + Place holder Word doc: <https://microsoft-my.sharepoint.com/:w:/p/unathaniel/EQqhgjmv7jhBip35itZaH7gBQZxquach68a3HJB1_ktNng?e=1P68lJ>
* Code samples to boostrap your development: <https://github.com/Azure-Samples/real-time-monitoring-and-observability-for-media>