Microsoft

Big data streaming:

Choices for high availability and disaster recovery on Microsoft Azure

By Arnab Ganguly

DataCAT

March 2019

Contents

List of figures

Authored by Arnab Ganguly. Edited by Nanette Ray. Reviewed by Microsoft DataCAT and AzureCAT.

Overview

Many enterprises use Microsoft Azure to build real-time data ingestion and processing pipelines that ingest and process messages from IoT devices (typically sensors) into a big data analytics platform. Millions of high-speed messages and events are generated by the IoT devices and sensors. When these are sensitive, time-critical, or linked to human health and safety, downtime is not an option. A high availability (HA) design and a disaster recovery (DR) solution are imperative.

This document analyzes a fictitious use case based on actual big data streaming customers and their real-world HA/DR experiences on Azure. In working directly with enterprises, Microsoft always presents options, so this document does the same. It describes four approaches to an HA/DR configuration and the technical details of each approach—from a customer's perspective. The goal is to make it easy to understand HA/DR requirements and considerations on Azure and to show you options suitable for various IoT streaming scenarios.

Enterprises have multiple choices for implementing HA and DR in an Azure-based big data solution. Ideally, a solution should be designed using a combination of services deployed in Availability Zones and created in ways that take advantage of cross-region availability:

- Availability Zones are unique physical locations within an Azure region. Each zone is made up of one or more datacenters equipped with independent power, cooling, and networking.
- Cross-region availability adds redundancy and availability for Azure resources across defined geographic regions.

The HA/DR options discussed in this guide are relevant for any real-time streaming scenarios that use sensors to manage and monitor complex systems in industries such as smart buildings, communications, manufacturing, retail, and healthcare. However, this guidance is not meant to be prescriptive or to present a reference architecture. The solutions presented in this guide assume that certain technologies are already in place, as is often the case with the enterprises we work with. The intention is to show how a typical enterprise weighs its choices based on its goals. More alternatives are mentioned in the Weighing the options section.

NOTE: A comparison of the solution costs is beyond the scope of this guide. Cost is driven by multiple factors, including data sizes, compute requirements, regions, and available discounts, so a good comparison can't be made. This discussion is limited to the technical possibilities.

The challenge of a single-region architecture

A manufacturer we'll call Fabrikam Inc. runs its IoT message processing architecture in a single cloud region. Its solution ingests and stores messages in real time from approximately 800,000 devices, and the company expects that number to increase to 2 million in the next few years. Its solution analyzes the messages to understand the information emitted through different types of sensors, and it stores the data after analysis for other downstream processing.

When a region-wide outage caused its deployment to go down, Fabrikam was caught by surprise. During the outage, the company could not ship equipment to its customers. In addition, when the IoT devices could no longer reach the datacenter, they switched over to satellites in an attempt to connect. The invoice that resulted from this switch was another unpleasant surprise.

Running its solution in a single region caused an issue for the Fabrikam developers, as well. They were unable to use blue-green deployments, since the only environment was the live one.

Fabrikam knew that it needed to rethink its streaming architecture, make the services more resilient and highly available, and support disaster recovery. Yet the company needed a better understanding of failover in the event of a region-wide failure and needed to know how individual components would behave. Additionally, Fabrikam didn't have the time, budget, or inclination to start from scratch, especially if it could reuse at least some of the existing elements of its single-region architecture. These components included:

- Azure Event Hubs for event aggregation.
- Apache Storm on Azure HDInsight for event processing.
- Apache HBase in HDInsight for event storage.
- Azure Blob storage for storing messages captured by Event Hubs and for lookups by the bolt components in Apache Storm on HDInsight.

Figure 1 depicts the company's single-region architecture. For the purposes of this guide, the producers (IoT devices) and the consumers (apps and dashboards) are considered to be resilient.

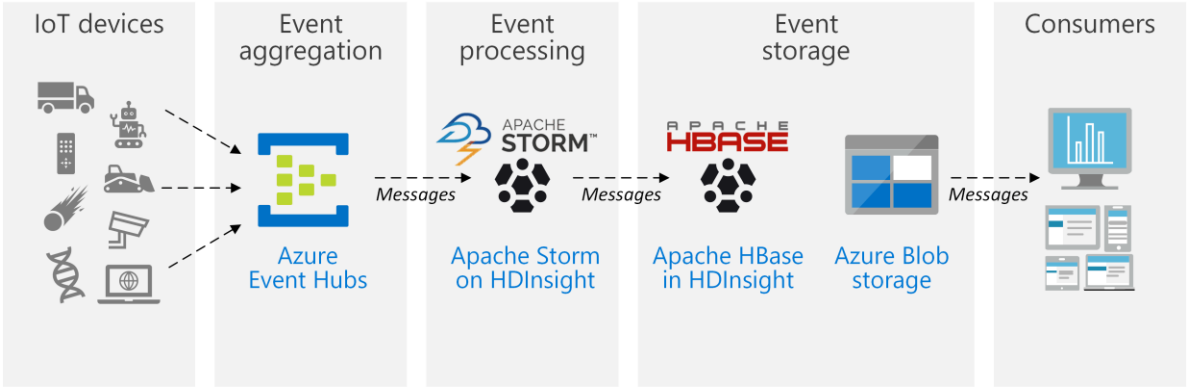

Figure 1. The original Fabrikam solution, deployed in a single cloud region

When Fabrikam sought help from Microsoft, the company identified a number of goals for a new architecture, starting with improvements to:

- **Availability.** The architecture needed to provide high availability across a single region, be operational in the event of a region-wide disaster, and provide resilience against individual service downtime.
- **Resiliency.** The Recovery Point Objective (RPO) and the Recovery Time Objective (RTO) needed to be fewer than five minutes. Message ordering was not necessarily a requirement, but retention mattered—30 days minimum.
- **Flexibility.** The new architecture needed to grow as Fabrikam added devices. In addition, the developers wanted support for blue-green deployments and canary releases.

- **Cost.** The new architecture needed to be cost effective. Fabrikam wanted to use resources more efficiently and to reduce idle resources. The company also hoped to implement automatic scaling so it could eliminate the need to oversize cloud resources in anticipation of fluctuating loads.

Fabrikam asked Microsoft to show its team the best practices for an IoT message processing architecture and to suggest several possibilities from which to choose.

Configuration considerations

A highly available, redundant architecture consists of two instances deployed in separate regions. The primary instance is always active, and the other instance can be active or passive, as follows:

- **Active-passive.** Only the primary region is active at any given time, and the secondary region is on standby. When a disaster occurs or a component fails in the primary region, failover occurs to the passive configuration in the other region, which becomes active.
- **Active-active.** Both regions are active at any given time, and each receives a subset of the total number of events. Events can be routed to a particular region in multiple ways, the most common being proximity.

Fabrikam was open to considering both *active-active* and *active-passive* configurations, so Microsoft showed the company both, using different combinations of components.

Software considerations

For the new architecture, Fabrikam explored choices for each step in the streaming pipeline.

Event aggregation

The company considered two ingestion technologies:

- Azure Event Hubs, a message queuing solution for ingesting millions of event messages per second. The captured event data can be processed by multiple consumers in parallel. To automate scaling, Event Hubs includes an Auto-Inflate feature that scales up to meet demand by increasing the number of throughput units. Event Hubs also supports geo-disaster recovery for metadata and relies on primary and secondary disaster recovery namespaces to fail over between regions in the event of a disaster. In addition, Event Hubs is also supported by Availability Zones, making the service even more resilient in a single region.
- Apache Kafka, an open source message queuing and stream processing application that can scale to handle millions of messages per second from multiple message producers and can route them to multiple consumers. Apache Kafka on HDInsight is available in Azure as an HDInsight cluster type. To drive real-time scaling in HDInsight clusters, Fabrikam can use the scale API of Azure HDInsight in conjunction with Azure Log Analytics.

Event processing

After capturing real-time messages, the Fabrikam solution processes them by filtering, aggregating, and preparing the data for analysis. Three stream processing technologies were considered:

- Apache Storm, an open source framework for stream processing that uses a topology of spouts and bolts to consume, process, and output the results from real-time streaming data

sources. Storm is available in an Azure HDInsight cluster. Azure HDInsight is a cloud distribution of Apache Hadoop and provides an end-to-end service level agreement on all production workloads. To support real-time scaling, Fabrikam can use the scale API of Azure HDInsight together with Log Analytics.

- Azure Functions, a managed compute platform. Simple, serverless functions process the incoming messages in real time and store them in a database for use downstream.
- Azure Databricks, an Apache Spark–based analytics platform optimized for the Microsoft Azure cloud services platform. Fabrikam could leverage the Spark Streaming functionality of Apache Spark on HDInsight to receive live input streams of data, divide them into batches using Spark micro-batching, and then process them using the Spark engine.

Each of these technologies works with Azure Redis Cache. This cache is based on the popular open source technology that improves the performance and scalability of systems that rely heavily on back-end data stores.

Event storage

The next step in the processing pipeline is for the Fabrikam big data solution to prepare the messages using an analytical data store. For the new architecture, Fabrikam considered the following technologies:

- Apache HBase, an open source NoSQL store. HBase in Azure HDInsight is offered as a managed cluster that is integrated into the Azure environment. The clusters are configured to store data directly in Azure Storage, which provides low latency and increased performance elasticity.
- Azure Cosmos DB, a turnkey database with global distribution that supports serverless architectures. With its high throughput, Azure Cosmos DB is well suited for IoT architectures.

This table summarizes the options presented to Fabrikam for the new architecture. Each is described in more detail in the sections that follow.

Option	Event handling	Stream processing	Database
Minimal change	Event Hubs	Storm on HDInsight	HBase in HDInsight
Open source	Kafka on HDInsight	Storm on HDInsight	HBase in HDInsight
Open source	Kafka on HDInsight	Azure Databricks, Spark on HDInsight	HBase in HDInsight
Serverless	Event Hubs	Azure Functions	Azure Cosmos DB

Minimal change option

The first configuration Fabrikam considered was designed to minimize the degree of change and to reuse as many of the existing components as possible. The minimal change option uses:

- Event Hubs for event aggregation.
- Storm on HDInsight for event processing.
- Hbase in HDInsight for event storage.

The significant difference is that these components are deployed in Availability Zones, wherever possible, and in paired Azure regions—East US 2 (primary) and Central US (backup).

In addition to deploying Event Hubs in Availability Zones in the primary region, in the active-passive configuration shown in Figure 2, event ingestion uses Event Hubs geo-disaster recovery. This globally available feature creates a namespace in the primary region and pairs it with a secondary namespace in the standby region. This pairing is then monitored so the secondary namespace can be activated when failover is triggered. For more information, see Azure Event Hubs - Geo disaster recovery in the Azure documentation.

Storage resiliency can be achieved by using Zone Redundant Storage (ZRS) to deploy storage accounts across Availability Zones or by using read-access geo-redundant storage (RA-GRS) to deploy multi-region locally redundant storage (LRS) accounts. In this case, to handle storage replication across regions, RA-GRS is made available in the passive Central US region. RA-GRS provide reads from the secondary region, irrespective of whether Microsoft initiates a failover.

To meet data processing requirements, identical Storm on HDInsight clusters are hosted in both regions. In addition, HBase Master/Master replication is enabled at the table level to create a completely caught up replica of the HBase table in Central US. In the event of a disaster, the table writes resume at the secondary region. For more information, see the Apache HBase documentation.

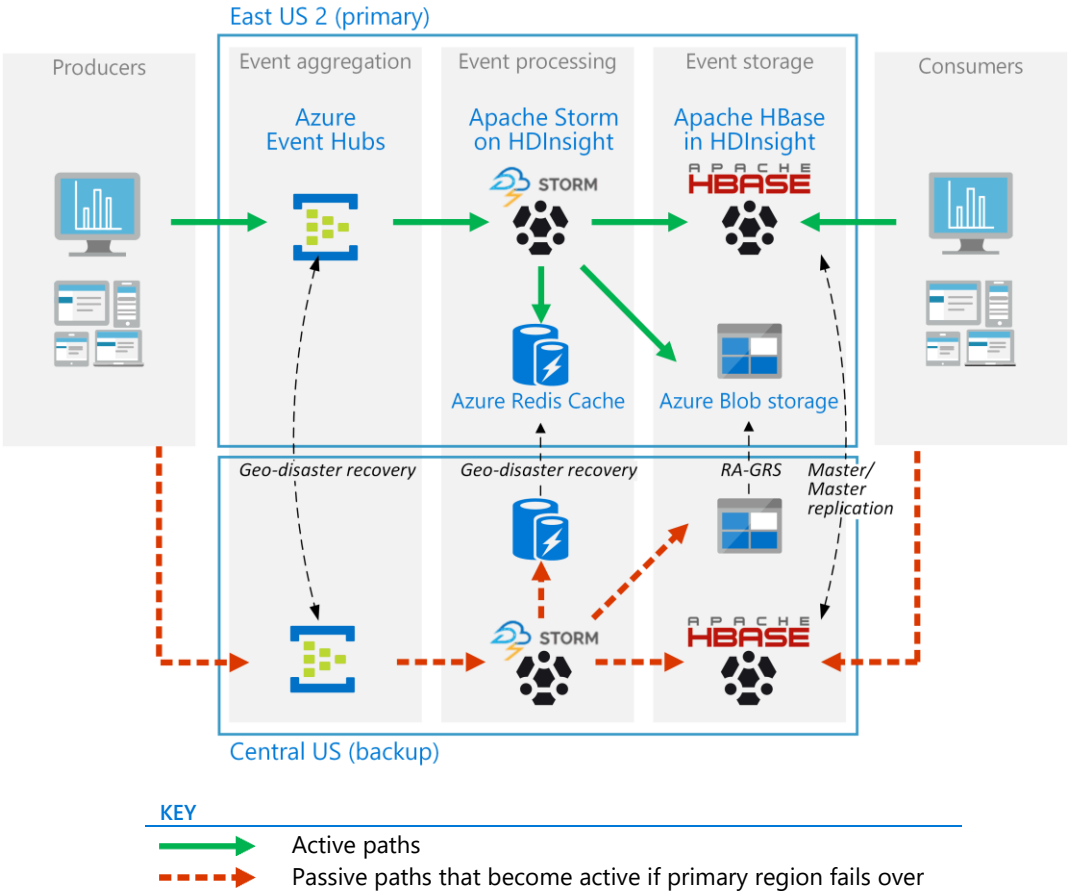

Figure 2. An active-passive configuration that reuses the existing Fabrikam components

The main difference in the active-active configuration shown in Figure 3 is that identical copies of Event Hubs are created in both regions by mirroring the Event Hubs namespaces. To further save costs, Fabrikam can also maintain automation scripts that deploy the Storm on HDInsight clusters when the Event Hubs fails over to the secondary region. Using script-based deployment and message playback capability, Storm on HDInsight can then start processing messages where it left off earlier.

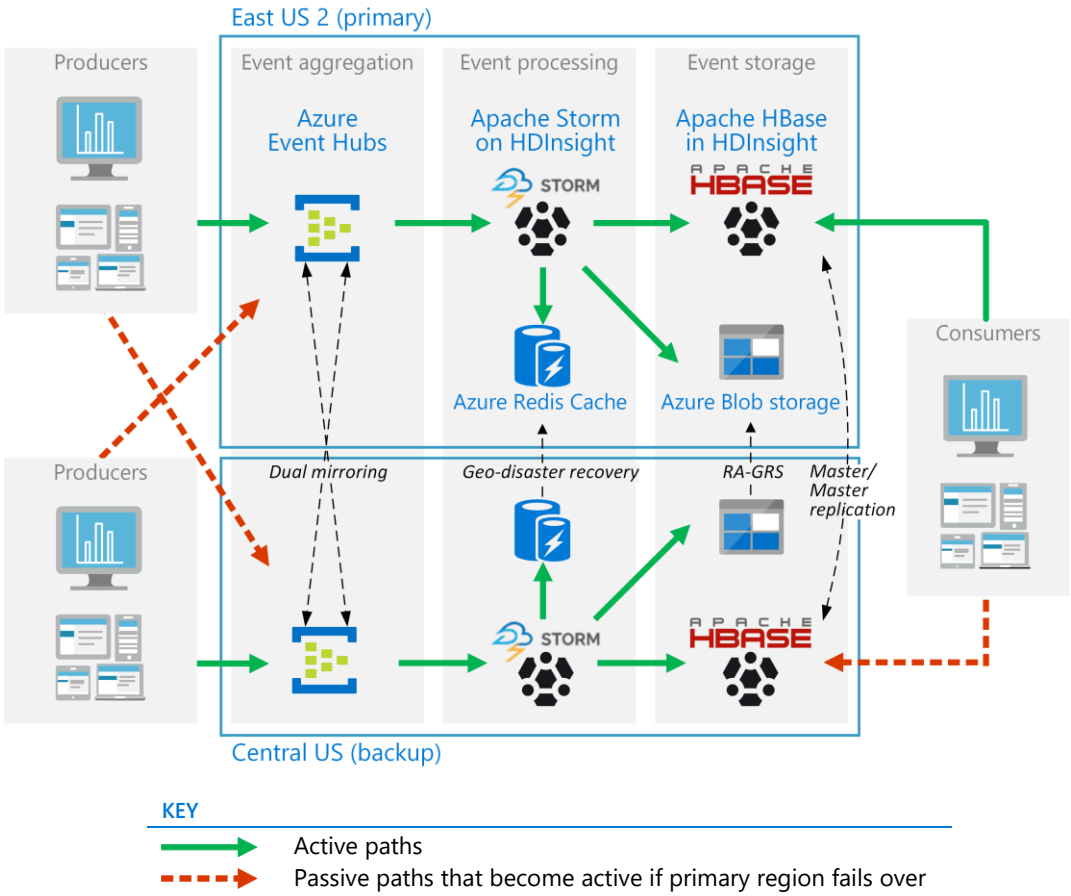

Figure 3. Active-active configuration that reuses the existing services

This configuration provides the key advantage of familiarity and would take the least time to implement. However, Fabrikam had to consider a few tradeoffs, as shown in the following table.

Pros	Cons
• Minimally disruptive because it is most like the existing architecture. • Event Hubs is a fully managed PaaS service with Availability Zone support and geo-disaster recovery. • Fabrikam can use the Event Hubs message playback to mitigate pipeline component failures.	• The Event Hubs message size is approximately 256 KB. • No failback is available in Event Hubs. Manual pairing needs to be established again. • With the Fabrikam custom DNS servers, setting up resiliency for the virtual network-to-virtual network port forwarding is challenging.

NOTE: Service limits can often be increased to support specific customer needs. For the latest information about Event Hubs limits and capabilities, see Event Hubs limits.

Open source option

The second option Fabrikam considered was designed to make the most of open source Apache services, since the company was already invested in Storm and HBase. This version of the architecture uses:

- Apache Kafka for event aggregation.
- Storm on HDInsight for event processing.
- HBase in HDInsight for event storage.

In this scenario, the Azure East US 2 region is primary and paired with the Central US region as the secondary. As in the earlier option, Event Hubs is deployed using Availability Zones to provide greater resiliency within a region.

The active-passive configuration shown in Figure 4 ingests events using identical Kafka clusters in each region, with one-way mirroring between the primary and secondary regions.

For event processing, identical Storm on HDInsight clusters are hosted in both the regions. In the passive region, the Azure Redis Cache geo-replication feature creates a replica. For cross-region replication, Azure Storage RA-GRS provides reads from the secondary region, irrespective of whether Microsoft initiates a failover. To further save costs, Fabrikam can maintain automation scripts that deploy the Storm on HDInsight clusters on a need-only basis in the secondary region. Using script-based deployment and message playback capability, Storm on HDInsight can then start processing messages from Kafka where it left off earlier.

For event storage, HBase Master/Master replication is enabled at the table level to create a completely up-to-date replica of the HBase table in Central US. In the event of a disaster, the table writes resume at the secondary region.

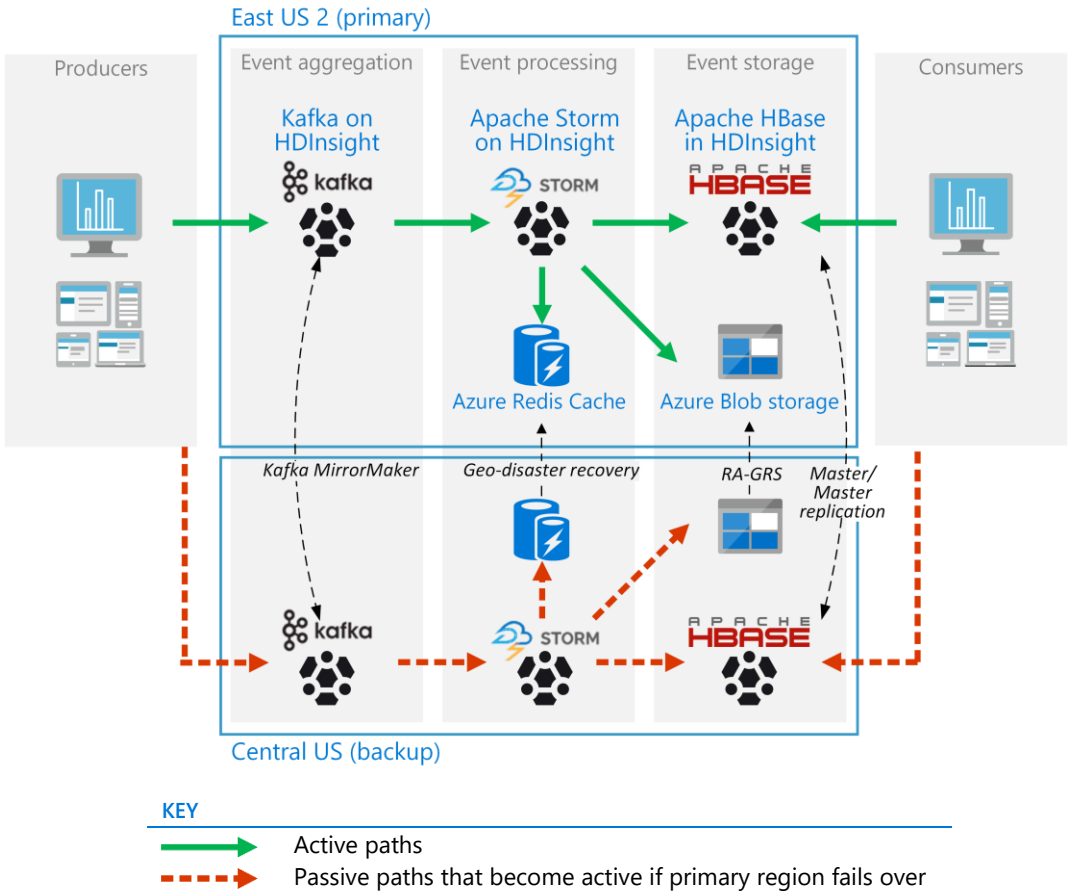

Figure 4. Active-passive configuration of the open source option

In the active-active configuration shown in Figure 5, event ingestion is handled by identical Kafka clusters that are replicated using the MirrorMaker utility's dual-side mirroring. For more information about this feature, see the Azure HDInsight documentation. In addition, HBase Master/Master replication is enabled at the table level.

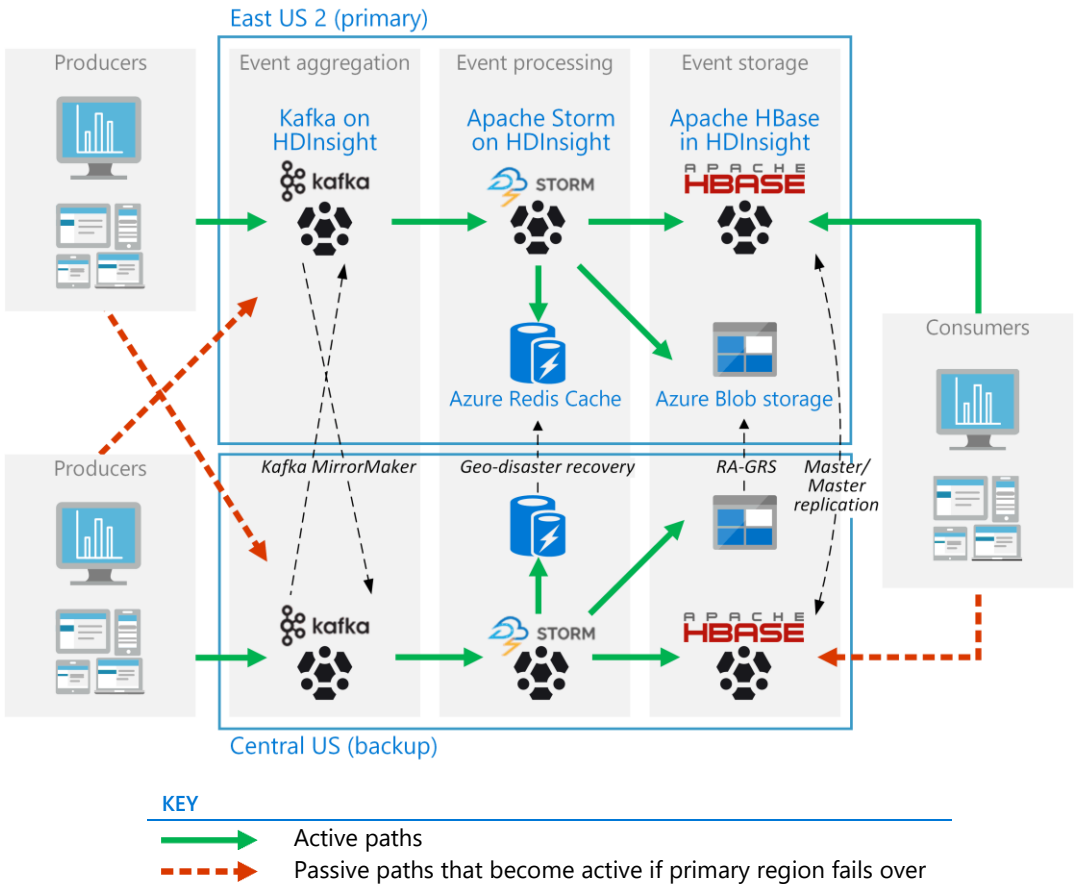

Figure 5. Active-active configuration of the open source option

Like the minimal change option, familiarity is an advantage for Fabrikam with this configuration. However, the company needs to weigh that against a few new challenges, as shown in the following table.

Pros	Cons
• Messages can fluctuate in size, and large messages are not an issue. • Fabrikam can use the Kafka message playback capability to mitigate pipeline component failures. • Components and architecture are reliable, operate at scale, and have been time-tested by multiple enterprises running very large Kafka clusters on Azure. • Fabrikam can retain messages for any number of days.	• HDInsight components don't provide automatic scale-in and scale-out features, and this lack can lead to increased cluster sizes and inefficient use. • With Fabrikam custom DNS servers, setting up resiliency for the virtual-network-to-virtual-network port forwarding is challenging.

Open source alternative

The third option Fabrikam considered was, like the previous one, designed to make the most of open source Apache services. The difference is that this option uses Spark message processing instead of Storm. This version of the architecture uses:

- Kafka on HDInsight for event aggregation.
- Spark on HDInsight and Azure Databricks for event processing.
- HBase in HDInsight for event storage.

Azure Databricks is an Apache Spark–based analytics platform optimized for Azure. In this scenario, events generated from the data sources are sent to the stream ingestion layer through Kafka on Azure HDInsight as a stream of messages. A Kafka consumer, Azure Databricks picks up the message in real time from the Kafka topic to process the data based on the business logic and then sends it to HBase for storage.

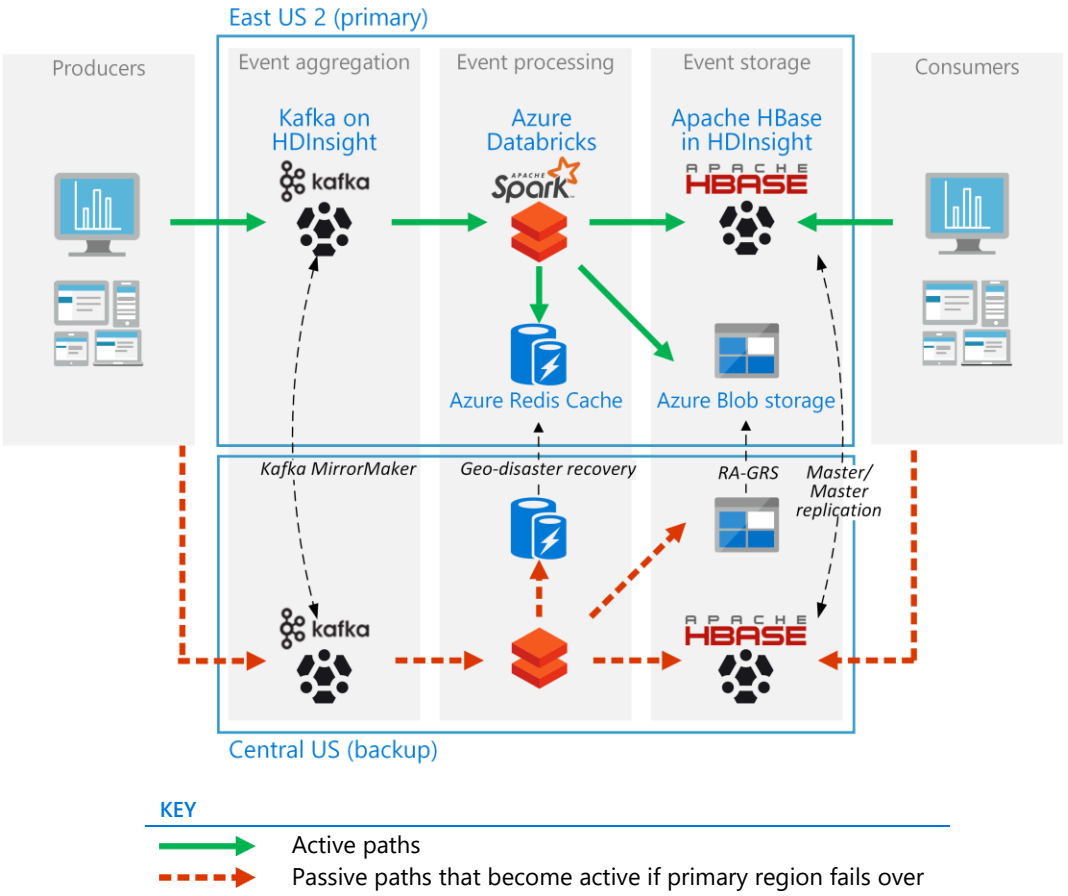

Figure 6. Active-passive configuration of the open source alternative

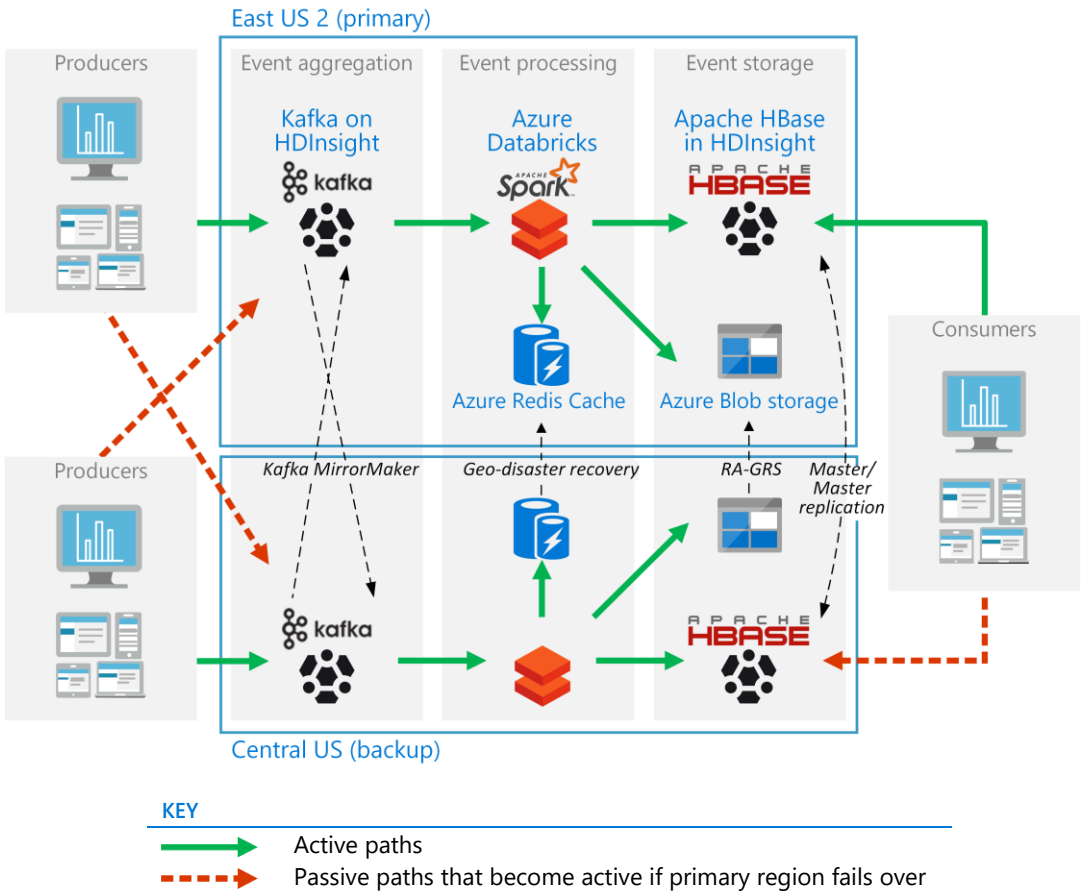

Figure 7. Active-active configuration of the open source alternative

This alternative gives Fabrikam many of the same advantages as the other open source option, as the following table shows.

Pros	Cons
• Messages can fluctuate in size, and large messages are not an issue. • Fabrikam can use the Kafka message playback capability to mitigate pipeline component failures. • Components and architecture are reliable, operate at scale, and have been time-tested by multiple enterprises running very large Kafka clusters on Azure. • Fabrikam can retain messages for any number of days. • Azure Databricks provides automatic scale-in and scale-out features.	• With Fabrikam custom DNS servers, setting up resiliency for the virtual network-to-virtual network port forwarding is challenging. • Fabrikam currently uses Storm, so its Java code (Storm) must be rewritten in Scala or Python (Spark), which can take more time and effort.

Serverless option

Fabrikam has the option to use only Azure services. This approach redesigns the event transformation tier using a serverless architecture and reimagines the back-end storage using a globally distributed, multi-model database service. This option uses:

- Event Hubs for event aggregation.
- Azure Functions for event processing.
- Azure Cosmos DB for event storage.

Like the other two options, two Azure regions are paired to provide high availability. In the passive region shown in Figure 8, event ingestion is handled by Event Hubs backed by Availability Zones. The geo-disaster recovery feature is used as it is in the minimal change configuration—that is, a primary namespace is paired with a secondary namespace to support failover. In the passive region, once again, Azure Redis Cache geo-replication creates the replica and Azure Storage RA-GRS supports cross-region replication of storage.

The big change is seen in event processing and storage. Event Hubs bindings are used to trigger function code created in Azure Functions. Simple, serverless functions scale to meet demand so Fabrikam doesn't have to worry about servers or infrastructure. Azure Cosmos DB, the recommended database for serverless computing architectures, replaces HBase for event storage. Azure Cosmos DB includes a multi-master feature that supports global active-active apps. Fabrikam can configure its databases to be globally distributed and available in any Azure regions. An application doesn't need to be paused or redeployed to add or remove a region. It continues to be highly available all the time because of the multihoming capabilities that the service provides.

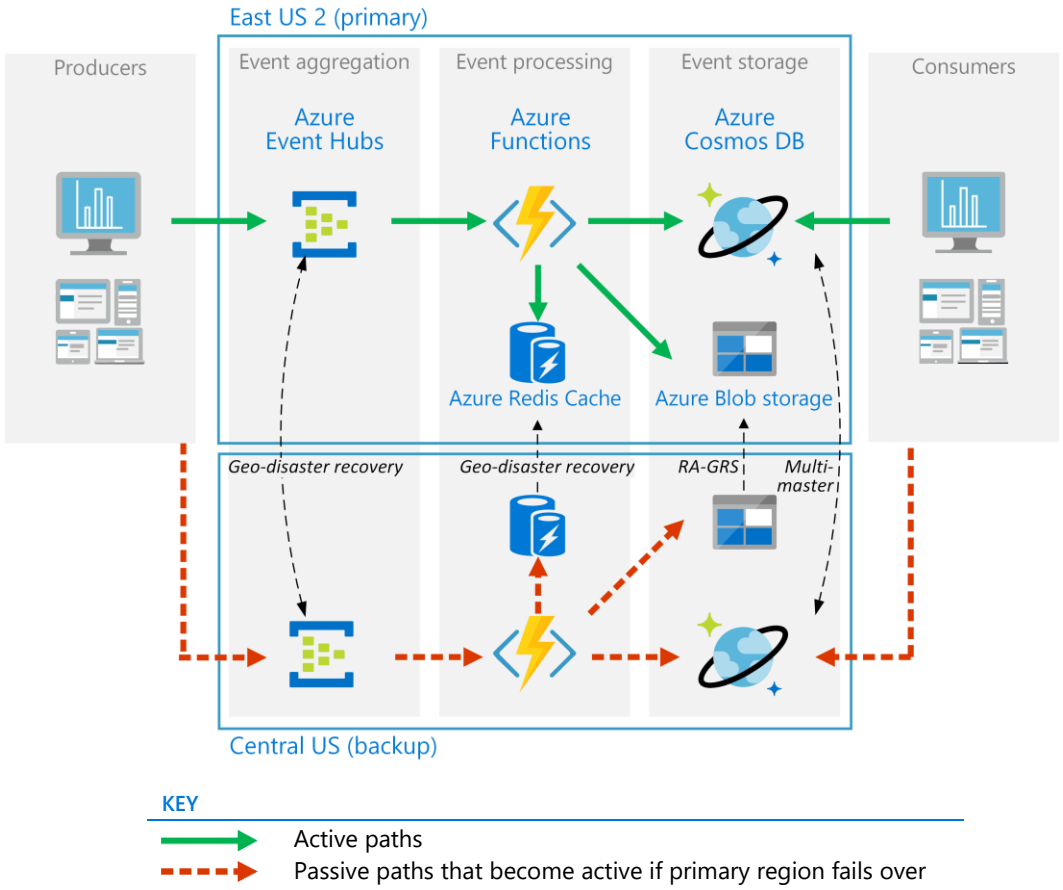

Figure 8. Active-passive configuration of the serverless option

In the active-active configuration shown in Figure 9, each producer sends about half of its data to each region in a dual-site configuration. As with all the options, this approach works when the producers are close to the regions they send data to. Identical copies of Event Hubs are created in both regions by mirroring the Event Hubs namespaces. Similarly, identical instances of Azure Cosmos DB transparently replicate the data across all the Azure regions associated with the Fabrikam account. Multi-master replication makes every region a write region and also readable.

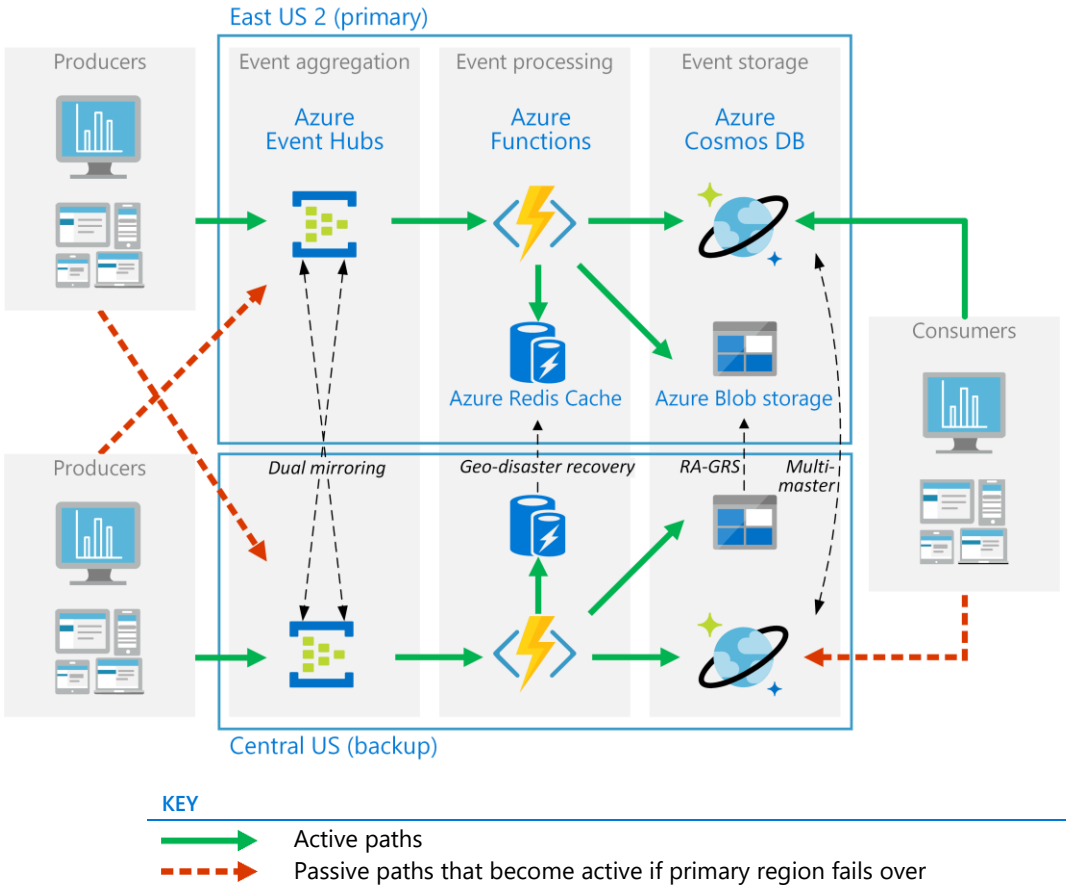

Figure 9. Active-active configuration of the serverless option

Of all the options presented to Fabrikam, this serverless architecture requires the least infrastructure to manage.

Pros	Cons
• A serverless architecture scales on demand, and Fabrikam pays only for the resources it consumes. • Using managed Azure services gives Fabrikam the minimum of management overhead.	• Fabrikam prefers familiarity, and this option is the most unlike its existing architecture. • Message size is still an issue.

For another example of a representative IoT event processing pipeline based on Functions and Event Hubs, see the Processing 100,000 Events Per Second on Azure Functions blog post.

Weighing the options

Azure makes it easy, fast, and cost effective to process massive amounts of data. But any company that embarks on a mission-critical, cloud-based big data solution—whether for real-time or batch workloads—needs to plan for contingencies. The Fabrikam story is a lesson in planning for availability at all levels.

To achieve comprehensive business continuity on Azure, Fabrikam could choose to rebuild its application architecture using any of the options in this guide and a combination of Availability Zones with Azure region pairs. It can synchronously replicate applications and data using Availability Zones within an Azure region for high availability and can asynchronously replicate across Azure regions for disaster recovery protection.

Azure provides several options for implementing a digital data hub like that of Fabrikam. The following diagram illustrates the possibilities for stream ingestion, transformation, data hub, and various application services. Even in a single region deployment, each piece of the puzzle can be implemented in a variety of ways.

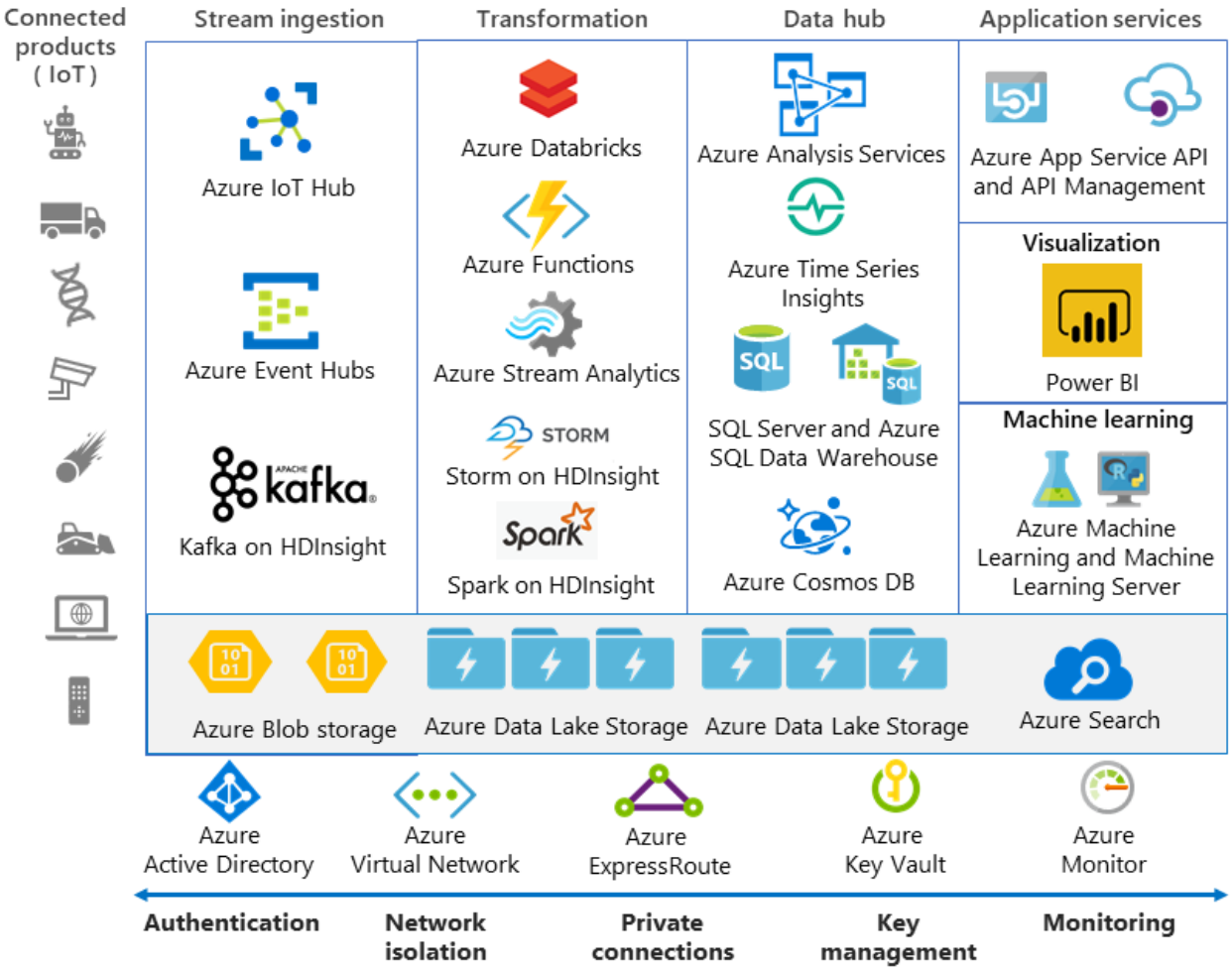

Figure 10. Options for deploying a digital data hub

Learn more

For more information, see the following resources:

- Example scenario: IoT and data analytics in the construction industry
- Example scenario: Ingestion and processing of real-time automotive IoT data
- Azure Architecture Center: Real-time processing
- What is Azure HDInsight and the Apache Hadoop technology stack