Three paths to delivering a successful EDA

RTInsights.com

Red Hat

We are living in a cloud-native world. For many organizations, scale-out, distributed, microservices-based environments – which can deliver greater scalability, reliability, and efficiency – have become the default approach to software design and deployment.

Yet simply migrating to cloud-native architectures doesn't guarantee the full benefits that this strategy stands to offer. Ensuring that your cloud-native environment is as agile, flexible, and engaging as possible requires the ability to build highly responsive applications that can make decisions in real time – no matter which type of environment they are hosted in, which languages they are written in or which services connect them together.

This is why event-driven architectures, or EDA, have become a foundational part of modern, cloud-native environments. By enabling real-time decisions based on event streaming, easily deployable EDA platforms provide businesses with a means of taking application performance and responsiveness to the next level. They also make it easy to expand analytics functionality over time as use cases grow more complex.

In order to help decision-makers understand the importance of EDA within modern IT strategies, this eBook describes how EDA works, why it's an essential ingredient in the typical environment today, and how to enable EDA using open source technologies such as Apache Kafka (an event streaming platform) and Kubernetes (an orchestrator for application microservices).

As the following pages explain, no matter how your organization's environment is designed or which types of applications you deploy, open-source EDA solutions help bring flexibility and efficiency to development and IT teams while also creating more responsive environments that optimize the end-user experience. In so doing, they smooth the path to modernization.

Chapter 1: The What and Why of EDA

Event-driven architecture, or EDA, is a way of designing applications and services to respond to real-time information based on the sending and receiving of information about individual changes to the state of a business application. Instead of processing data in batches and only then modifying configurations, EDA allows applications to react continuously to events within the environment that hosts them.

Benefits of EDA

EDA offers application design and deployment that benefit a range of stakeholders:

- **Architects:** EDA makes it easier to design applications as a set of loosely-coupled services that use events to determine when and how to communicate with each other.

- **Developers:** Developers can leverage EDA to build highly dynamic applications that respond to changing conditions, including those that developers may not anticipate when writing code. What's more, because EDA can work with applications written in any language or framework, it provides a flexible way for developers to integrate applications within polyglot environments.

- **Customers:** For end-users, EDA means an experience that is more immediate and responsive than conventional applications allow. With EDA, applications can deliver personalized content to each individual end-user based on the events that occur within the user session, leading to more engaging experiences that reflect real-time user input.

For a longer discussion of the advantages of EDA, read on to chapter 2, which dives deep into EDA use cases.

EDA and cloud computing

The origins of EDA stretch back to before the cloud age. However, the migration of the majority of enterprise workloads to the cloud has created a new imperative for integrating EDA into application deployment patterns and management strategies.

EDA is a natural fit for the cloud for several reasons. One is that applications in the cloud are often deployed as loosely-coupled microservices rather than as the monolithic applications that predominated in the era of on-premises computing. When applications are broken into microservices, each service operates autonomously, which translates to greater agility and resiliency. However, each service must also interact constantly with other microservices. Attempting to manage microservices interactions using tightly coupled communications (meaning those based on periodic batch processing) would not only be inefficient but would make it difficult to build microservices applications that can respond quickly to changes in user behavior or expectations.

With EDA, it becomes easy to let events control how microservices behave and interact. For example, a Customer Relations Management (CRM) application could publish changes to customer data to a stream, allowing other applications – such as a call center database, which needs accurate and up-to-date customer data whenever a customer calls – to collect and process data from that stream in real time.

The shift to the cloud has also increased user expectations in ways that make EDA a valuable architectural solution. Users expect a high degree of immediacy and personalization, and developers need to leverage vast volumes of data to deliver it. By using streaming event data to shape application behavior in real time, developers can build applications that efficiently leverage data to deliver personalized user experiences that are updated in real time.

A third advantage of EDA is that it works with virtually any type of IT environment and application. You don't need to write your application in a specific language to take advantage of EDA; you just need to deploy an EDA framework that allows applications to detect and respond to events. Likewise, EDA can be applied equally well to a single-cloud, multi-cloud, hybrid cloud, or even conventional, on-premises environment.

The role of Kafka

Apache Kafka has emerged as a leading solution for streaming the events data that provides the foundation for an EDA approach to application deployment and management. It enjoys the highest market share among even streaming platforms, and is used by tens of thousands of companies, according to research by Enlyft.

Although Kafka is only one of several open source streaming message brokers available today (other popular options include Apache Pulsar and Apache ActiveMQ), Kafka has gained massive adoption thanks to the following factors:

Scalability: Kafka uses a distributed architecture that allows Kafka servers to be added or removed quickly, making it possible to scale the system up as necessary to handle very large volumes of streaming data.

Fault tolerance: The distributed, redundant nature of Kafka also makes the system highly tolerant. Event streaming is not disrupted by the failure of some Kafka servers.

Low latency: Kafka offers latency rates as low as ten milliseconds, enabling processing in virtually true real time.

To implement EDA with the help of Kafka, businesses also need a means of deploying and orchestrating their application microservices via a platform – such as OpenShift – that automatically manages container-based microservices applications. But as a high-performing and reliable message broker, Kafka provides the foundation for an EDA solution stack.

Challenges in EDA

While EDA offers a range of benefits, it also presents several challenges that organizations must plan for in order to integrate EDA effectively into their IT strategy:

- **Starting the migration:** One is choosing where to start. In most cases, implementing a full-scale EDA across the entire application environment is not practical. Instead, teams must start small – by choosing a single application or use case, for instance – and work up from there to move more of their applications and services to an event-driven approach.

- **Ensuring flexibility:** Devising a flexible EDA strategy is also a challenge. The EDA tools that organizations choose should not lock them into a particular ecosystem or type of architecture. It's critical to plan an EDA solution that can evolve along with your business' overall IT strategy and that gives you the level of control you need over how EDA tools are deployed and managed.

- **Acquiring expertise:** Organizations must ensure that they have access to the expertise they need to roll out and manage EDA tools effectively. EDA is a complicated landscape, with many tools and implementation strategies available. Finding the best approach for your business requires technical skills that not all organizations possess in-house.

These challenges can all be overcome, as the following chapters explain. But it's essential to account for them as you plan and implement an EDA strategy for your business.

Chapter 2: Three top EDA user patterns

The preceding chapter touched briefly on the benefits of EDA. But because EDA can be applied to solve so many different types of problems in modern application environments, it's worth diving deeper into the specific use cases for EDA.

In general, EDA use cases in modern, cloud-native environments can be broken down into three main categories: achieving real-time event streaming, integrating loosely-coupled microservices, and enabling efficient communications within complex cloud architectures.

Move from batch data to real-time streaming

Traditionally, the default approach to data processing within applications was to manage data in batches, or by using a shared data access tier or nonstandard APIs. The first approach meant that applications would wait until they had collected a certain amount of data before processing it, rather than processing each unit of data in real time as it was generated. The second and third approaches required complex tooling and architectures, making management a challenge.

Batch processing is easier to implement, but it hampers the customer experience. When applications have to wait to process data in batches, customers have to wait on results – which is a problem in a world where the typical customer won't wait even two seconds for a website to load. A product inventory in an online store may be updated only every ten minutes, for example, leading to situations where customers are allowed to add an item to their cart even if the item has already sold out due to delays in reconciling the product listings with actual inventory availability.

With streaming data, these frustrations disappear. An eCommerce app that takes advantage of streaming event data can update its inventory in real time as each transaction occurs. As another example, real-time streaming also makes it practical to deliver instant product recommendations tailored to each customer, which would be hard to do when working with batch data.

In short, EDA makes it possible for applications to become as responsive as possible and react to events immediately, as opposed to the periodic updates that batch processing enables.

Interoperate across hybrid cloud

As businesses shift to hybrid cloud architectures that combine on-premises resources with those hosted on one or more public clouds, they need a way to manage those resources centrally.

APIs that are based on open standards are part of the solution to this challenge because open APIs allow resources on disparate platforms to communicate with each other.

But a key limitation of APIs is that they operate on a request-response model. If the state of a resource changes, other resources won't know of the change until they check the API again. For this reason, APIs within hybrid environments work best only for managing synchronous operations.

In contrast, event streams enable asynchronous interoperability without requiring constant check-ins to keep resources informed of changes. Instead, the various resources running in a hybrid environment can publish state changes to an event stream, and other resources can monitor them continuously from that location.

In this respect, EDA brings flexibility to hybrid cloud environments that APIs alone cannot provide.

Connect loosely-coupled microservices

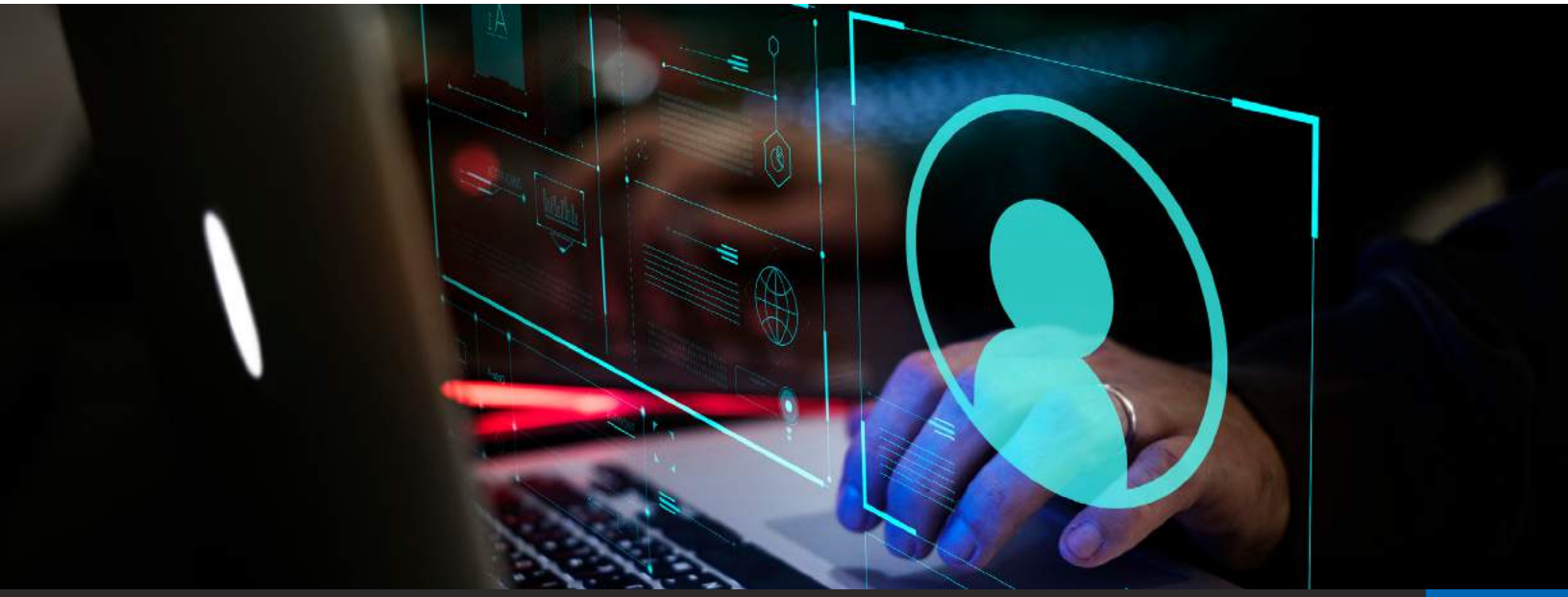
As noted above, modern applications are often deployed as a set of microservices. Each microservice operates independently yet depends centrally on other microservices to do its job. One microservice might handle user logins, for example, while a second, front-end microservice generates content to display to the user upon login, and a third tracks user activity in order to help build an individualized profile of user preferences and behavior.

Microservices offer a variety of benefits to developers, including the ability to keep different areas of application functionality discrete and independent. However, they also present a major challenge in that it can be difficult to implement an efficient means of coordinating microservices activity while still ensuring that each microservice remains independent and loosely coupled from others. If you make the behavior of one microservice contingent upon another microservice, you end up with microservices that are tightly coupled and that lack scalability and resilience.

For instance, imagine a scenario where a microservice that handles user logins can't process a transaction until it receives data from the front-end microservice that is responsible for generating the content that the user will see upon login. In this case, not only will it take longer to log the user in (because the login transaction can't complete until the front-end content is generated), but there is also a risk that the login may fail entirely if the front-end service times out or is unresponsive.

Problems like this can be avoided with an EDA approach in which microservices operate in parallel rather than depending on each other. In an event-driven environment, a login microservice and a front-end microservice could react independently to the event of a user login request. Each microservice would do its work independently and in parallel, leading both to a faster transaction and to a lower risk of failure in the event that one microservice is slow to respond.

These examples highlight the benefits that EDA offers as a way to achieve the full potential of loosely coupled microservices architectures. Without EDA, a microservices application is at risk of behaving more like a monolith – unable to scale and ridden with single points of failure.



Chapter 3: Optimizing EDA for Architects and Developers

EDA enables a more positive user experience and allows architects and developers to take full advantage of cloud-native application architectures. However, implementing EDA can also pose challenges from a technical perspective.

As chapter 1 noted, there are a number of EDA solutions available and multiple approaches to implementing EDA. A key task for architects and developers, then, is to identify an EDA solution that delivers a powerful and flexible experience from a technical perspective while also enabling all of the customer-centric benefits that EDA unlocks.

There are several factors to consider in this regard.

Uniting the entire IT organization

To deliver their full potential, EDA tools should not be limited to use by certain teams. Instead, they should enable all teams to communicate and integrate their IT assets via events – regardless of which types of assets the teams develop or which tools they use to create them.

In this way, EDA ensures that teams retain the independence they need to operate with agility while still providing a consistent framework for unifying communications across all corners of the IT estate.

Polyglot application support

As noted above, EDA tools should allow any application to record, detect and respond to events, no matter how the application is designed or which language it is written in. Solutions that work with only certain types of applications prevent organization-wide adoption of EDA as a means of unifying disparate clouds and application environments.

Diverse architecture support

Along similar lines, EDA tools that only work with certain architectures – such as only within a certain public cloud or only on-prem – limit teams' ability to leverage EDA in a way that covers their entire IT estate while also maximizing flexibility. Instead, teams need EDA solutions that work with any and all types of IT environments – on-prem, single cloud, multi-cloud, and hybrid cloud.

Self-managed vs. fully-managed deployment options

Different teams have different levels of expertise when it comes to working with EDA. They also have varying priorities with regard to the amount of control they need over their EDA framework.

For that reason, developers and architects can derive the greatest benefit from EDA solutions that give them the option of deploying and managing EDA tools themselves or using a fully managed approach. The former strategy delivers more control but also requires more effort and expertise. The latter provides the most seamless EDA experience but comes with some restrictions regarding how tools can be configured and used.

Choice rules

All of the above can be summed up by saying that, when it comes to optimizing EDA from the perspective of developers and architects, having a maximum choice is a key priority. The more choices teams have in how they use EDA tools, which applications and architectures the tools support, and how the tools are configured and deployed, the better-positioned teams are to implement EDA in a way that supports rather than hinders their overall IT strategy.

Chapter 4: Integrating EDA into the Cloud-Native Ecosystem

So far, we've discussed what EDA means, why it's important to both end-users and technical teams, and which main factors to consider when selecting an EDA solution.

There remains one important consideration to address: how to fit EDA into the broader ecosystem of cloud-native tools and processes that power your business. What we mean here is not tool integration in a technical sense but rather implementing EDA in such a way that it aligns with and enhances other technological priorities.

This is important because, once again, there are a variety of possible approaches to EDA, but they don't all jibe with existing IT assets in equally effective ways.

Embrace the ecosystem

That's why it's critical to devise an EDA strategy that ties EDA into the broader technological ecosystem that powers your business. The best EDA solutions are those that fit naturally into larger platforms and that are integrated by default with the other tools and technologies your organization already uses.

Without taking care to fit your EDA solution neatly within your broader technology ecosystem, it becomes much harder to implement an efficient and flexible EDA strategy.

Open source

Choosing an open source approach to EDA is one way to ensure that EDA integrates smoothly into your broader ecosystem. Open source EDA solutions, like Kafka deployed on top of OpenShift, provide maximum flexibility for businesses to integrate EDA with all of their applications and environments.

An open source approach also means teams receive rapid updates as open source tools evolve, while avoiding the risk of being locked into a proprietary vendor platform.

Vendor experience and expertise

The most effective EDA solutions come from vendors who have deep experience not just with EDA but with the broader set of cloud-native technologies, of which EDA is but one part. Less experienced vendors who specialize in EDA alone are less likely to deliver tools that are compatible with the full suite of cloud-native services and architectures available today.

Support

Obtaining reliable, continuous support for EDA solutions is also important for ensuring that they can align with and grow alongside your other cloud-native investments. Without proper support, it is much more challenging not only to troubleshoot issues that arise within streaming data architectures but also to identify the best ways to evolve your EDA strategy over time.

Achieving EDA success with Red Hat OpenShift Streams for Apache Kafka

Red Hat® OpenShift® for Apache Kafka is a fully hosted and managed Kafka service for stream-based applications that reduces the operational cost and complexity of delivering real-time applications across hybrid-cloud environments using an EDA approach.

Red Hat, which has been actively working to deploy and support Kafka in enterprise environments since 2018, has learned how best to take advantage of EDA technology in demanding applications. Red Hat offers the expertise not just to configure and manage Kafka, but also how to optimize the entire EDA solutions stack and management strategy. In OpenShift Streams for Apache Kafka, Red Hat offers an opinionated service that makes it fast and easy to get started with Kafka-driven EDA.

To learn more about how OpenShift Streams ***for Apache Kafka*** helps businesses unify and streamline their environments using EDA, ***try Kafka service***.

RTInsights.com
Accelerate Your Business with Real-Time Insights

RTInsights is an independent, expert-driven web resource for senior business and IT enterprise professionals in vertical industries. We help our readers understand how they can transform their businesses to higher-value outcomes and new business models with AI, real-time analytics, and IoT. We provide clarity and direction amid the often confusing array of approaches and vendor solutions. We provide our partners with a unique combination of services and deep domain expertise to improve their product marketing, lead generation, and thought leadership activity.

Red Hat

Red Hat is the world's leading provider of enterprise open source software solutions, using a community-powered approach to deliver reliable and high-performing Linux, hybrid cloud, container, eventing, and Kubernetes technologies. Red Hat helps customers develop cloud-native applications, integrate existing and new IT applications, and automate and manage complex environments. A trusted adviser to the Fortune 500, Red Hat provides award-winning support, training, and consulting services that bring the benefits of open innovation to any industry. Red Hat is a connective hub in a global network of enterprises, partners, and communities, helping organizations grow, transform, and prepare for the digital future.