An event-driven architecture uses events to trigger and communicate between decoupled services and is common in modern applications built with microservices. An event is a change in state, or an update, like an item being placed in a shopping cart on an e-commerce website. Events can either carry the state (the item purchased, its price, and a delivery address) or events can be identifiers (a notification that an order was shipped).

معماري رويداد محور از رويداد ها براي راه اندازي و ايجاد ارتباط بين سرويس هاي جدا استفاده ميكند و در اپلیکیشن های مدرن ساخته شده با میکرو سرویس ها بسیار رایج هستند . یک رویداد یک تغییر یا بروز رسانی وضعیت است ، مانند یک کالا که درسبد خرید یک وب سایت تجارت الکترونیک قرار دارد . رویدادها می‌توانند دارای وضعیت باشند (کالای خریداری‌شده، قیمت آن و آدرس تحویل) یا رویدادها می‌توانند شناسه (اعلان ارسال سفارش) باشند.

Event-driven architectures have three key components: event producers, event routers, and event consumers. A producer publishes an event to the router, which filters and pushes the events to consumers. Producer services and consumer services are decoupled, which allows them to be scaled, updated, and deployed independently

معماری رویداد محور سه بخش کلیدی دارد : رویداد تولیدکنندگان ، رویداد روترها ، و رویداد مصرف کنندگان . یک تولید کننده رویدادی را به سمت روتر نشر می دهد که آن رویداد را فیلتر کرده و به سمت مصرف کننده می فرستد . سرویس های تولید کننده و سرویس های مصرف کننده از هم جدا هستند که به آن ها اجازه می‌دهد به صورت مستقل مقیاس گذاری ، بروز رسانی و به کار گرفته شوند .

Benefits of an event-driven architecture

مزایای معماری رویداد محور

### Scale and fail independently

### 1-به طور مستقل مقایسه و شکست بخورند

### By decoupling your services, they are only aware of the event router, not each other. This means that your services are interoperable, but if one service has a failure, the rest will keep running. The event router acts as an elastic buffer that will accommodate surges in workloads.

### به دلیل جدا بودن سرویس های شما، آنها فقط از رویداد های روتر آگاه هستند .نه از رویداد های هم این بدان معناست که سرویس‌های شما قابلیت همکاری دارند، اما اگر یکی از سرویس‌ها با مشکل مواجه شود، بقیه به کار خود ادامه می‌دهند. رویداد روتر به عنوان یک بافر قابل ارتجاع عمل میکند که با افزایش بارکاری جلوگیری می‌کند.

### Develop with agility

### 2-توسعه چابک

### You no longer need to write custom code to poll, filter, and route events; the event router will automatically filter and push events to consumers. The router also removes the need for heavy coordination between producer and consumer services, speeding up your development process.

### دیگر نیازی به نوشتن کد سفارشی برای نظر سنجی ، فیلتر و مسیر یابی رویداد ها ندارید . رویداد روتر به طور اتوماتیک فیلتر و ارسال میکند رویداد ها رو به مصرف کنندگان . روتر همچنین نیاز به هماهنگی شدید بین خدمات تولید کننده و مصرف کننده را از بین می برد و روند توسعه شما را سرعت می بخشد.

### 3- Audit with ease

### 3-تسهیل در حسابرسی

### An event router acts as a centralized location to audit your application and define policies. These policies can restrict who can publish and subscribe to a router and control which users and resources have permission to access your data. You can also encrypt your events both in transit and at rest.

### روتر رویداد به عنوان یک مکان متمرکز برای بررسی برنامه شما و تعریف خط مشی ها عمل می کند.این خط مشی ها می توانند انحصار ایجاد کنند که چه کسی . همچنین می توانید رویدادهای خود را هم در حالت حمل و نقل و هم در حالت استراحت رمزگذاری کنید.

### 4-cut costs

### 4-کاهش هزینه ها

### Event-driven architectures are push-based, so everything happens on-demand as the event presents itself in the router. This way, you’re not paying for continuous polling to check for an event. This means less network bandwidth consumption, less CPU utilization, less idle fleet capacity, and less SSL/TLS handshakes.

### معماری رویداد محور مبتنی بر فشار است ، بنابراین همه چیز بر اساس تقاضا اتفاق می‌افتد زیرا رویداد‌ها خود را در روتر ها نمایش می‌دهد . این به معنای کاهش مصرف پهنای باند شبکه ، استفاده کمتر CPU ، ظرفیت ناوگانی کمتر .

### نحوه کارکرد معماری رویداد محور

### نحوه کار این معماری را با ذکر یک مثال توضیح می دهیم

### C:\Users\user\Desktop\للل.jpg.png

### زمان استفاده از این معماری

### Cross-account, cross-region data replication

### تکثیر داده های متقابل، بین منطقه ای

### You can use an event-driven architecture to coordinate systems between teams operating in and deploying across different regions and accounts. By using an event router to transfer data between systems, you can develop, scale, and deploy services independently from other teams.

### شما می توانید از معماری رویداد محور برای هماهنگ کردن سیستم ها بین تیم های عملیاتی و حساب هایی که در ناحیه های متفاوتی قرار گرفته اند استفاد کنید . با اسفتاده از روتر رویداد برای جابجایی داده ها بین سیتم ها ، شما می توانید توسعه دهید ، مقیاس بدهید و سرویس ها را مستقر کنید مستقل از سایر تیم ها .

### Resource state monitoring and alerting

### نظارت و هشدار وضعیت منابع

### Rather than continuously checking on your resources, you can use an event-driven architecture to monitor and receive alerts on any anomalies, changes, and updates. These resources can include storage buckets, database tables, serverless functions, compute nodes, and more.

### به جای چک کردن پیوسته منابع خود ، شما میتوانید از یک معماری رویداد محور برای نظارت و دریافت هشدار در رابطه با هر ناهنجاری ، تغییرات و بروز رسانی ها ، استفاده کنید . منابع می توانند شامل سطل های ذخیره سازی ، جدول پایگاه داده ها ، رویداد ها از دست دادن سرورها ، گره های محاسباتی و غیره باشند .

### Fan-out and parallel processing

### میزان خروجی و پردازش موازی

### If you have a lot of systems that need to operate in response to an event, you can use an event-driven architecture to fan out the event without having to write custom code to push to each consumer. The router will push the event to the systems, each of which can process the event in parallel with a different purpose.

### اگر شما تعداد زیادی سیستم دارید که در پاسخ به یک رویداد نیاز به یک اقدام دارند ، شما میتوانید از معماری رویداد محور برای نمایش رویداد بدون نیاز به نوشتن یک کد اختصاصی برای ارسال به هر مصرف کننده استفاده کنید . روتر رویداد را به سمت سیستم هایی هدایت خواهد کرد که هر یک می توانند رویداد را به طور موازی با اهداف خاص پردازش کنند .

### Integration of heterogeneous systems

### ادغام سیستم‌های ناهمگن

### If you have systems running on different stacks, you can use an event-driven architecture to share information between them without coupling. The event router establishes indirection and interoperability among the systems, so they can exchange messages and data while remaining agnostic.

### اگر شما سیستم هایی دارید که بر روی پشته‌های متفاوتی اجرا میشوند ، شما می‌توایند از معماری رویداد محور برای به اشتراک گذاری اطلاعات بین تیم‌ها بدون گروه شدن استفاده کنید . روتر رویداد غیرمستقیم و قابلیت همکاری بین سیستم ها را ایجاد می کند، بنابراین آن‌ها مي‌توانند پیام‌ها و داده‌ها را مبادله کنند و در عین حال ناشناس باقی بمانند .

### چرا باید از معماری رویداد محور استفاده کنیم ؟

### معماری رویداد محور برای بهبود چابکی و جابجایی سریع بسیار ایده آل است . آن‌ها اغلب در اپلیکیشن‌های مدرنی که از میکروسرویس‌ها استفاده می‌کنند یافته می‌شوند یا هر اپلیکیشنی که اجزاء جدا از هم دارد . زمانی که یک معماری رویداد محور را اتخاذ می‌کنید ، ممکن است لازم باشد در نحوه مشاهده طرح برنامه خود تجدید نظر کنید. برای اینکه خود را برای موفقیت آماده کنید، موارد زیر را در نظر بگیرید:

### دوام منبع رویداد شما منبع رویداد شما باید قابل اعتماد باشد و در صورت نیاز به پردازش تک تک رویدادها، تحویل را تضمین کند.

### الزامات کنترل عملکرد شما برنامه شما باید بتواند ماهیت ناهمزمان روترهای رویداد را مدیریت کند.

### ردیابی جریان رویداد شما غیر جهت معرفی شده توسط یک معماری رویداد محور، امکان ردیابی پویا را از طریق خدمات نظارتی فراهم می کند، اما نه ردیابی ایستا از طریق تجزیه و تحلیل کد.

### داده ها در منبع رویداد شما. اگر نیاز به بازسازی حالت دارید، منبع رویداد شما باید کپی شده و سفارش داده شود.

### رفرنس این بخش https://aws.amazon.com/event-driven-architecture/

**Overview**

### بررسی اجمالی

Event-driven architecture is a [software architecture](https://www.redhat.com/en/topics/cloud-native-apps/what-is-an-application-architecture) and model for application design. With an event-driven system, the capture, communication, processing, and persistence of events are the core structure of the solution. This differs from a traditional request-driven model.

معماری رویداد محور یک معماری نرم افزاری و مدل برای طراحی نرم افزار است . با یک سیستم رویداد محور، ضبط، ارتباط، پردازش و تداوم رویدادها ساختار اصلی راه حل هستند.این متفاوت از مدل سنتی درخواست محور است .

Many modern application designs are event-driven, such as [customer engagement frameworks](https://www.redhat.com/en/solutions/customer-experience) that must utilize customer data in real time. Event-driven apps can be created in any programming language because event-driven is a programming approach, not a language. Event-driven architecture enables minimal coupling, which makes it a good option for modern, distributed application architectures.

بسیای از طراحی های اپلیکیشن مدرن رویداد محور است ، مانند چارچوب های تعامل با مشتری که باید از داده های مشتری در زمان واقعی استفاده کنند . اپلیکیشن های رویداد محور می توانند با تمام زبان های برنامه نویسی به وجود بیایند زیرا معماری رویداد محور یک رویکرد است نه یک زبان . معماری رویداد محور حداقل کوپلینگ را ممکن می کند، که آن را به گزینه خوبی برای معماری های کاربردی مدرن و توزیع شده تبدیل می کند.

An event-driven architecture is loosely coupled because event producers don’t know which event consumers are listening for an event, and the event doesn’t know what the consequences are of its occurrence.

یک معماری رویداد محور به طور ضعیفی پیوند خورده است، زیرا تولیدکنندگان رویداد نمی‌دانند مصرف‌کنندگان رویداد به کدام رویداد گوش می‌دهند، و رویداد نمی‌داند عواقب وقوع آن چیست.

یک رویداد چیست ؟

An event is any significant occurrence or change in state for system hardware or software. An event is not the same as an event notification, which is a message or notification sent by the system to notify another part of the system that an event has taken place.

یک رویداد هر رخداد قابل توجه یا تغییر در وضعیت برای سیستم سخت افزاری یا نرم افزاری است . یک رویداد همان اعلام رویداد نیست، که پیام یا اطلاع بوسیله سیستم برای اطلاع دادن به بخش دیگری از سیستم که یک رویداد صورت گرفته است ، ارسال شود .

The source of an event can be from internal or external inputs. Events can generate from a user, like a mouse click or keystroke, an external source, such as a sensor output, or come from the system, like loading a program.

منبع یک رویداد میتواند ناشی از ورودی های خروجی یا داخلی باشد .رویداد میتواند توسط یک کاربر به وجود آورد ، برای مثال یک کلیک موس یا کیبورد ، یک منبع خارجی مانند سنسور خارجی ، یا از سیستم مانند لود شدن یک برنامه .

چگونه یک معماری رویداد محور کار می‌کند؟

Event-driven architecture is made up of event producers and event consumers. An event producer detects or senses an event and represents the event as a message. It does not know the consumer of the event, or the outcome of an event.

معماری رویداد محور از تولید کنندگان رویداد و مصرف کنندگان رویداد تشکیل شده است . یک تولید کننده رویداد یک رویداد را تشخیص و حس می‌کند و رویداد را تحت یک پیام نشان می‌دهند .

After an event has been detected, it is transmitted from the event producer to the event consumers through event channels, where an event processing platform processes the event asynchronously. Event consumers need to be informed when an event has occurred. They might process the event or may only be impacted by it.

بعد از اینکه یک رویداد تشخیص داده می‌شود،توسط کانال‌های رویداد از تولید کننده رویداد به مصرف کننده رویداد منتقل می‌شود ، جایی که یک پلتفرم پردازش رویداد ، رویداد ها را به صورت ناهمزان پردازش می‌شود . مصرف کنندگان رویداد نیازدارند که هنگام رخداد یک رویداد ، اطلاع رسانی شوند.

آنها ممکن است که رویداد را پردازش کنند یا فقط از آن تاثیر بپذیرند .

The event processing platform will execute the correct response to an event and send the activity downstream to the right consumers. This downstream activity is where the outcome of an event is seen.

پلتفرم پردازش رویداد پاسخ صحیح را به یک رویداد اجرا می‌کنند و فعالیت پایین دست را به سمت مصرف کننده مناسب ارسال کند . این فعالیت پایین دست خروجی رویدادی است که دیده شده است .

[Apache Kafka](https://www.redhat.com/en/topics/integration/what-is-apache-kafka) is a distributed data streaming platform that is a popular event processing choice. It can handle publishing, subscribing to, storing, and processing event streams in real time. Apache Kafka supports a range of use cases where high throughput and scalability are vital, and by minimizing the need for point-to-point integrations for data sharing in certain applications, it can reduce latency to milliseconds.

There are other middleware event managers available that can serve as an event processing platform.

مدل های معماری رویداد محور

An event driven architecture may be based on either a pub/sub model or an event stream model.

معماری رویداد محور ممکن است براس یک مدل pub/sub باشد یا بر اساس مدل جریان رویداد باشد .

Pub/sub model

مدل pub/sub

This is a messaging infrastructure based on subscriptions to an event stream. With this model, after an event occurs, or is published, it is sent to subscribers that need to be informed.

این یک زیرساخت پیام رسانی مبتنی بر اشتراک در جریان رویداد است . با این مدل، بغد از رخداد یک رویداد ، یا منتشر شدن آن ، مشترکین نیاز به پیام رسانی دارد .

Event streaming model

مدل جریان رویداد

With an event streaming model, events are written to a log. Event consumers don’t subscribe to an event stream. Instead, they can read from any part of the stream and can join the stream at any time.

با یک مدل جریان رویداد ، رویداد‌ها در یک گزارش نوشته می‌شوند. مصرف کنندگان رویداد در یک جریان رویداد مشترک نمی‌شوند . در عوض، ، آنها می توانند از هر بخشی از جریان بخوانند و در هر زمان می توانند به جریان بپیوندند.

چند نوع مختلف جریان رویداد وجود دارد:

* **Event stream processing** uses a data streaming platform, like Apache Kafka, to ingest events and process or transform the event stream. Event stream processing can be used to detect meaningful patterns in event streams.
* **پردازش جریان رویداد** از یک پلتفرم جریان داده استفاده می‌کنند ، مانند اپاچی کافکا ، برای جذب رویداد ها و پردازش یا جابجایی یک جریان رویداد استفاده می‌کند . پردازش جریان رویداد ميتوانند استفاده شوند براي تشخيص الگوهاي معنادار در رويداد جريان استفاده شوند .
* **Simple event processing** is when an event immediately triggers an action in the event consumer.
* **پردازش رویداد ساده** زمانی است که یک رویداد بلافاصله اقدامی را در مصرف کننده رویداد راه اندازی میکند .
* **Complex event processing** requires an event consumer to process a series of events in order to detect patterns.
* **پردازش رویداد پیچیده** یک مصرف کننده رویداد نیاز دارد تا یک سری از رویداد ها را به ترتیب برای تشخیص الگو پردازش کند .