What is Event-Driven Architecture (EDA)?

[Ref: https://www.redhat.com/en/topics/integration/what-is-event-driven-architecture]

Overview

Event-driven architecture is a software 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.

Many modern application designs are event-driven, such as customer engagement frameworks 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.

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.

What's an event?

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.

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.

How does event-driven architecture work?

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.

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.
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.

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.

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

Pub/sub model
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.

Event streaming model
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.

There are a few different types of event streaming:

- ➢ 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.
- ➢ Simple event processing is when an event immediately triggers an action in the event consumer.
- ➢ Complex event processing requires an event consumer to process a series of events in order to detect patterns.

Benefits of event-driven architecture
An event-driven architecture can help organizations achieve a flexible system that can adapt to changes and make decisions in real time. Real-time situational awareness means that business decisions, whether manual or automated, can be made using all of the available data that reflects the current state of your systems.
Events are captured as they occur from event sources such as Internet of Things (IoT) devices, applications, and networks, allowing event producers and event consumers to share status and response information in real time.

Organizations can add event-driven architecture to their systems and applications to improve the scalability and responsiveness of applications and access to the data and context needed for better business decisions.