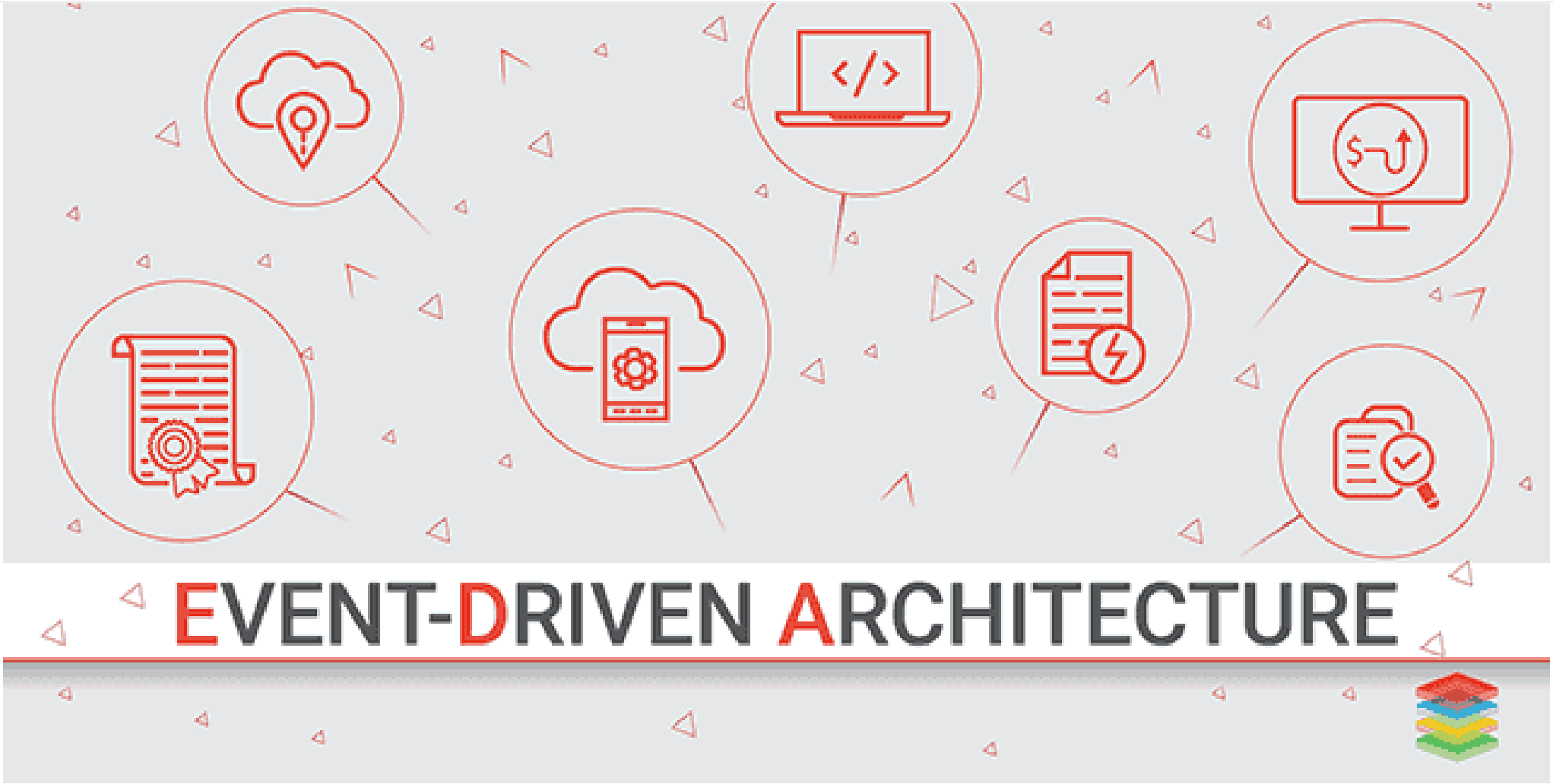
XENONSTACK
A Stack Innovator

Event Driven Architecture Patterns and Best Practices

INSIGHTS | 4 mins read | Oct 11, 2018

XENONSTACK
A Stack Innovator



What is Event Driven Architecture?

An Event Driven Architecture is a software architecture that operates services by the production, detection, and consumption of, and reaction to, significant changes in a system's state (known as events). This architecture consists of event creators and event consumers. Creators are the ones who create events and have knowledge about what events occurred while consumers are the ones who get affected

Why Event Driven Architecture Matters?

- **Loosely Coupled Structure** – Event creators and consumers loosely coupled with the help of event bus which helps them to couple and decouple into different networks in response to various events. It also helps in giving different developers to develop different event processors. And, these components can be used and reused by many different networks.
- **In Real-time Analytics** – Event-driven architecture is highly optimized for real-time analytics. It has a much higher capacity to find, analyze, and then respond to patterns in time before critical events happen. Complex event processing is used to achieve real-time analytics.
- **Versatility** – A system which follows event-driven architecture facilitates greater responsiveness because event-driven systems are, by design, normalized to unpredictable, nonlinear, and asynchronous environments. These systems are highly adaptable in different circumstances.

How to Adopt Event Driven Architecture?

- Multiple subsystems must process the same events.
- Processing at Real-Time with the minimum time lag.
- High velocity and high volume of data, such as IoT.
- Complex event processing, such as pattern matching or aggregation over time windows.

How Event Driven Architecture Works?

Publish/Subscribe (Pub/Sub) Messaging

It is a form of asynchronous service-to-service communication used in microservices and serverless architectures. When an event gets published or generated, it sends the message to each subscriber. A message doesn't repeat after its received, and new subscribers do not see the event. Subscribers do not need to know anything about the publishers, and vice-versa other than the publisher is a legitimate publisher and subscriber is entitled to receive information.

Event Source

Event source sends events to the event processing system. This sources can be one or more which can send an event to a single or multiples event processors.

Event Processor

This system can perform various actions on events –

- Merely enrich a single event; for example, adding a timestamp to the event data called as Simple Event Processing.
- Add information about the source of the event. Process multiple single events, from multiple event sources against event patterns to produce a new derived or compound event called Complex Event Processing.

Event Consumer

The event consumer reacts to the event. An event consumer can be as simple as updating a database or business dashboard, or as complex as required, carrying out new business processing as a result of the event.

Here are some examples of consuming an event –

- Updating a business dashboard
- Updating a database
- Interacting with Web 2.0 components to dynamically update web pages
- Sending an alert by using email or SMS based on event data

- Putting event data on a message queue
- Starting a new piece of application work

Event processing categorized by processing complexity and sophistication having two categories –

- **Simple Event Processing** – Simple events are events which are independent of other events and these events process under this category. These events do not summarize. Such events can provide considerably and provide excellent value business information. Simple processing includes processing such as augmenting the event payload with additional data, changing the events schema from one form to another, generating multiple events based on the payload of a single event, and redirecting the event from one channel or stream to another.
- **Complex Event Processing** – Complex events are events which are dependent on other events and these events process under this category. These events do summarize. Complex processing involves applying a collection of evaluation constraints or conditions over an event set. Results of processing should not be complicated for the user, distinguish between the complex and vibrant nature of the information available from this processing.

Top Terminologies of Event Driven Architecture

Event Bus

An event bus is a system or program or application which receives events from event creators and sends that event to any event consumers used in decoupling event creators and event consumers. Consumers can also post events to event bus if they are dependent on other events and event bus is not responsible if the receiver does not receive any message.

Event Creator/Producers

Event creator or producer or publisher is a program or application which creates events and send events to the event bus. Event creator knew what events occurred.

Event Consumer/Processor

Event consumer or processor or subscriber is a program or application which receive events from event bus without knowing which creator creates that event. Event processor processes the events and from that take particular actions written in events.

Event Log

The messages written to event logs stored on a disk and these messages are the ones written to an event queue. If the system crashes, the system can resume its state by merely replaying the events logs. Backup of event logs can be taken and used in performance tests on new releases before deploying them to production.

Topologies In Event Driven Architecture

Mediator Topology

It has a single event queue in which all events are stored in order and pops out in order only. A mediator which is used to direct each event to relevant event processors. Each event is filtered or pre-processes in an event channel before fed into the event processors.

Broker Topology

There is no event queue in this topology. Broker in this topology is event processors which are responsible for everything like to chain events, obtaining events, processing and publishing another event indicating the end. Once a processor processes an event, another event published so that another processor can proceed.

Best Practices of Event Driven Architecture

- Supports the business demands for better services (no batch, less waiting).
- No point-to-point integrations.
- High performance and highly scalable systems.
- Components can remain autonomous, being capable of coupling and decoupling into different networks in response to different events.

- Advanced analytics can be done using complex event processing.

How useful was this post?

Subscribe and Stay Updated!

Click to subscribe and get the latest updates and notifications of our Blogs and Use Cases to your inbox.

Subscribe Now

Share

XENONSTACK
A Stack Innovator

Product Engineering

Advanced Analytics

- XD Design Studio
- Products and Cloud Platform
- Product and Platform Development
- Machine Learning and Artificial Intelligence
- Data Products and Decision Science
- Business Intelligence and Data Visualization

Solutions

- Application Migration Solutions
- Refactoring and Cloud Native Applications
- DevSecOps and Cloud Security
- Enterprise DevOps and DataOps Transformation
- IoT Platform and Solutions
- Big Data,Continuous Performance and Load Testing
- Hybrid and On-Premises Cloud Infrastructure Solutions

Services

- Managed Cloud Services
- DevOps and Continuous Delivery
- Big Data Engineering
- Modern Data Warehouse
- Streaming and Real Time Analytics
- Big Data Infrastructure and Deployment

Resources

- Blog
- Use Cases
- Insights
- Industries
- Readiness Assessment
- About Us
- Clients and Partners
- Careers
- Contact Us
- Tao of Xenonstack

Managed Services - Big Data and Cloud

Center For Excellence - DARQ

Learn More

Learn More