Event Sourcing

In this lesson, we'll study event sourcing.

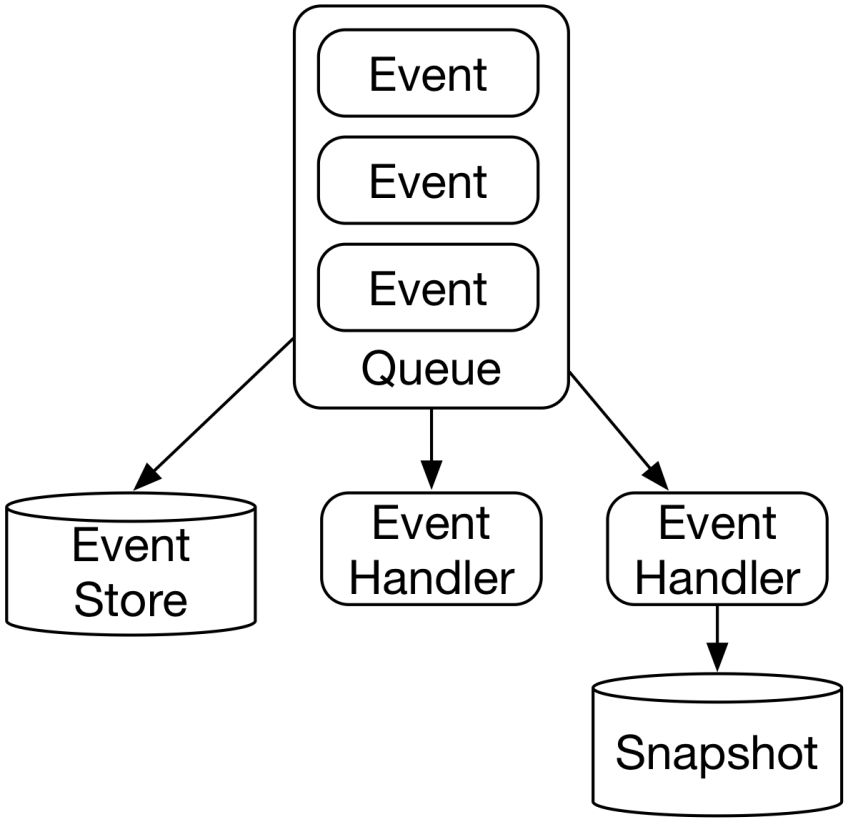
We'll cover the following

- Individual or shared event store?

An architecture's focus on events can also entail **other advantages**.

- The state each microservice has in its database is the result of the events it has received. The **state of a microservice can be restored** by resending all events it has received so far.
- The microservice can even change its internal domain model and then process the events again to rebuild its database with the **new version of the domain model**. This facilitates database schema migration.
- Thus, each microservice can have its own domain model according to the bounded context pattern, but **all microservices are still connected** by the events they send to each other.
- An overall state of the system no longer exists, but when all events are saved and can be retrieved, **the state of each microservice can be reconstructed**.

These ideas form the basis for event sourcing.



Event Sourcing

The elements of an event sourcing implementation are shown in the drawing above:

- The **event queue** sends the events to the recipients.
- The **event store** saves the events.
- **Event handlers** process the events. They can save their state as a *snapshot* in a database.

The event handler can read the current state from the snapshot, the snapshot can be deleted, and the snapshot can be restored on the basis of the events, which can be retrieved from the event store.

As an optimization, an event handler can also reconstruct its state from an older version of the snapshot.

There is a difference between the events for event sourcing and domain

events; see Christian Stettler's blog post (https://www.innoq.com/en/blog/domain-events-versus-event-sourcing/).

Individual or shared event store?

The event store can be part of the microservice that receives events and stores them in its own event store. Alternatively, the infrastructure not only sends the events but also stores them.

At first glance, it seems better if the infrastructure stores the events because it simplifies the implementation of the microservices. In such a case, the event store would be implemented in the event queue.

If each microservice stores the events in its own event store, the microservice can store all relevant data in the event, which the microservice may have collected from different sources.

When storing events in the infrastructure, it is necessary to find a model of the event that satisfies all microservices.

Such a model for the events can be a challenge because of the concept of bounded context. After all, every microservice is a separate bounded context with its own domain model, so finding a common model is difficult.

QUIZ

1

1

What is the difference between an event store and a snapshot?

A) The event store stores events whereas a snapshot stores screenshots of the app at certain intervals which is useful for debugging.

B) The event store stores events whereas a snapshot represents the state.

C) The event store stores app-wide events whereas a snapshot stores events for each microservice only.

Submit Answer

Question 1 of 2
0 attempted

Reset Quiz

In the next lesson, we'll look at some challenges associated with asynchronous microservices.

← Back

Next →

Events

Challenges: Inconsistencies & CAP Th...

Mark as Completed


Report an Issue