American Journal of Computer Architecture 2024, 11(1): 10-13
DOI: 10.5923/j.ajca.20241101.02

Solution Architecture for Salesforce CRM to Publish and Subscribe Events to Kafka

Swaran Kumar Poladi

Architect, IT - CRM, Medical Devices Company, Raleigh, USA

Abstract Companies that use Salesforce as their CRM system and use Salesforce CPQ, Sales or Service Cloud and transfer real-time events to ERP and other applications within company can become quite complex. This solution outlines how to process millions of real-time events on Salesforce by overcoming various governor and processing limits and making the solution scalable. The solution outlines the different API and transformation approaches used and how to publish or subscribe events to Kafka.

Keywords Salesforce, CRM, Solution Architecture, Kafka, Event-based framework

1. Introduction

In the current world of interconnected, real-time events play a pivotal role in making the right decisions at the right time. Today, Salesforce CRM which is widely used by many companies as a Customer Relationship Management platform would need to connect to various applications to receive and send real-time data events. Apache Kafka is a distributed event streaming platform. Salesforce offers a wide variety of ways to publish and subscribe to these real-time events. Performing transformations on Salesforce to process events is a complex activity. As Salesforce is a multi-tenant application it has many governor limits to ensure that shared resources are not monopolized for one single organization/tenant.

2. Overview of Salesforce Streaming APIs

Salesforce offers a wide variety of ways of APIs that can be used for publishing and subscribing events to and from Salesforce.

2.1. Pub/Sub API

1. Pub/Sub API is based on gRPC and HTTP/2 which efficiently publishes and delivers event messages in Apache Avro or JSON format.
2. This API provides a single interface for both publishing and subscribing to SFDC events.
3. Pub\Sub API has key benefits related to performance improvements compared to REST, HTTP/2 support, and bidirectional streaming.

2.2. Streaming API

1. Streaming API uses Bayeux protocol and CometD, which can be used to stream events out of Salesforce.
2. This API uses HTTP/1.1 request-response model.

2.3. Outbound Notifications

1. Outbound messaging uses the notification call to send SOAP messages over HTTP(S) to a designated endpoint.
2. This API uses the HTTP/1.1 request-response model.
3. Messages are queued locally in Salesforce and the designated endpoint application needs to process these messages.

3. Problem Statement

To publish or subscribe events directly from or to Kafka, Salesforce doesn't offer any standard libraries with Apex language. The code language that is used on the Salesforce platform is Apex. To overcome this, we either need to go with a middle-ware type of solution or we can use the open-source connectors to act as an intermediate to publish or subscribe events back and forth between Salesforce and Kafka. When using the open-source connectors or other paid connectors, event transformation needs to be handled within the Salesforce platform and needs to make sure the solution is optimized to work within the Platform without running into any issues.

* Corresponding author:
swaranssk@gmail.com (Swaran Kumar Poladi)
Received: Apr. 18, 2024; Accepted: May 7, 2024; Published: May 13, 2024
Published online at http://journal.sapub.org/ajca

4. Solution Architecture

4.1. Solution Overview

This solution approach has three main components. Salesforce, Open-Source Connector hosted on Google Cloud, and Kafka managed services hosted on Google Cloud.

4.1.1. Apache Kafka – Event streaming platform to process, stream, and store event data. It is distributed, so it can scale and handle billions of streaming events. The Kafka clusters can be hosted on premise or can be utilized as a managed service in the cloud. These managed services maintain the infrastructure of Kafka. Managed services like Apache Kafka for BigQuery, Confluent Cloud, or Pub/Sub of Google can be used. Create your application to create your Kafka topic. Before topic creation, establish the cluster of the topic and the cloud details where the cluster is. Create the topic and select the number of custom partitions for the topic. The Schema of the topic can be defined along with the format.

4.1.2. Open-Source Kafka Connector – Open-source 'Kafka Connector' hosted on Google Cloud running on a GKE (Google Kubernetes Engine) cluster. Connectors has two flavors, Source Connectors, and Sink Connectors. Idempotent an EIP pattern is used to filter out duplicate messages.

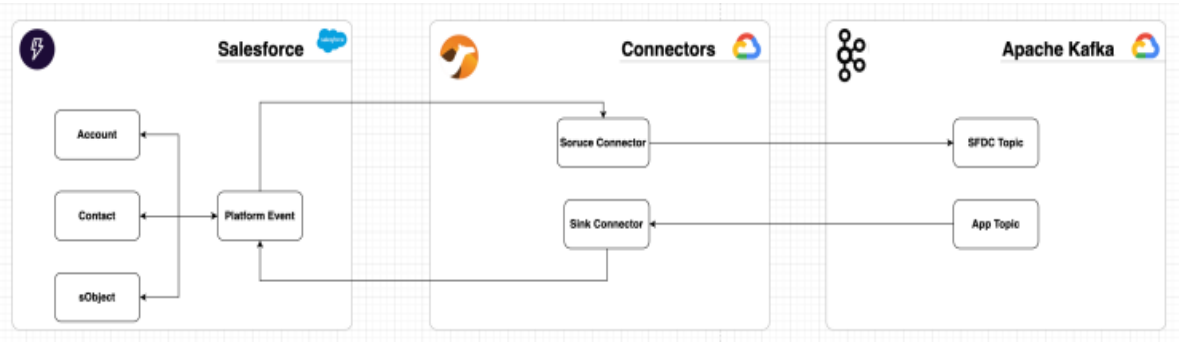

Figure 1. Solution Overview of Salesforce, Connectors, and Kafka

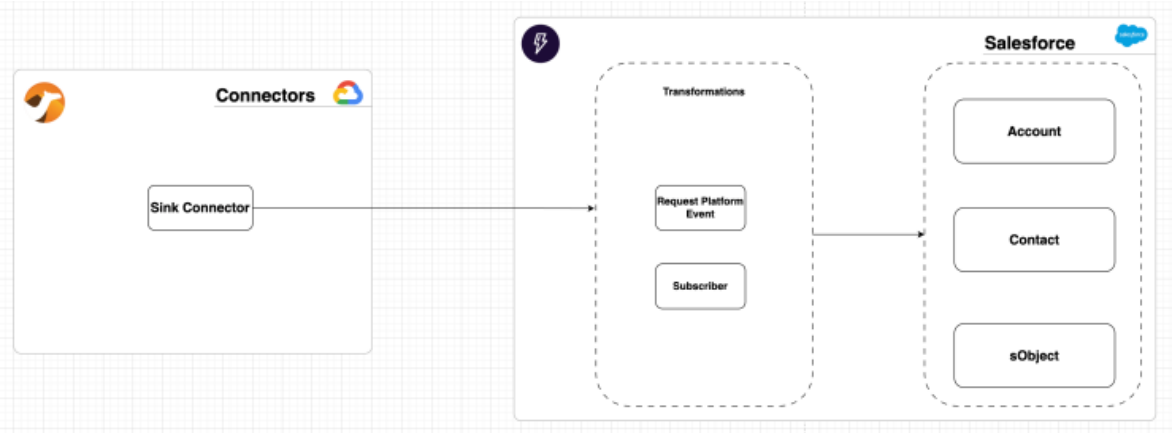

Figure 2. Sink Connector - Events into Salesforce

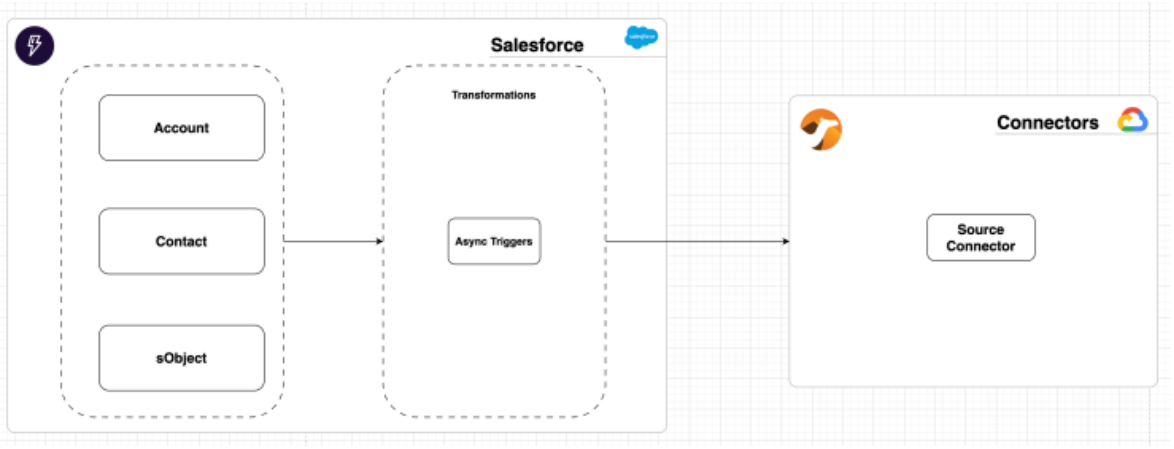

Figure 3. Source Connector - Events into Kafka from Salesforce

1. Source Connectors – These Source Connectors imports data from any application (in this scenario, events that are published on Salesforce) to Kafka. It uses different APIs to subscribe events from Salesforce to Kafka in JSON format.
2. Sink Connectors – These Sink Connectors import data from Kafka and publish/send them to any other application (in this scenario, events are published to Salesforce). Events are published to Salesforce using different APIs in JSON format.

4.1.3. Salesforce – CRM application where events are generated or published.

1. Inbound to Salesforce – Kafka Connector uses OAuth 2.0 to authenticate to Salesforce and events are transmitted using Pub\Sub API. An inbound event to Salesforce is transformed and processed using DataWeave Apex and then updated on to sObject of Salesforce.
2. Outbound to Salesforce – When an event is generated on Salesforce sObject, that event from Salesforce is transformed using DataWeave Apex and processed. sObject async triggers performs the transformation logic using DataWeave Apex. Async trigger to publish the transformed event to a Platform Event. Source connector subscribers to the platform event using Pub\Sub API to get the events from Salesforce and publish them to Kafka topic in JSON format.

4.1.4. Data Transformation Techniques

1. Any Data transformation needs to take place in the Salesforce Platform for both inbound and outbound events. Apex language has a standard Mulesoft DataWeave library to read and parse data from one format, transform the data and export it in a different format. DataWeave Apex supports JSON and XML processing and makes data transformation easier to code, more scalable, and efficient.
2. On the Kafka Connector, you should not have any data transformations performed to keep the connector light and easy to maintain.

4.1.5. Security Considerations

1. Authentication - OAuth 2.0 Username-Password Flow authentication can be used between Salesforce and Camel Connector Using this authentication salesforce\ connected app uses the access token to call Salesforce API, like its Pub\Sub or REST or SOAP APIs. Associate the integration user to the connected app. OAuth 2.0 Username & Password based authentication can be used to connect Apache Kafka and Camel Connector.
2. Authorization to Salesforce resources is done based on the user associated with the connected app. This user's permission set assignment determines the level of resources or different APIs available for the user or the integration.
3. Platform Event Messages in the Event Bus are encrypted at rest while they are stored in the Event Bus on the Salesforce Platform using Shield Encryption and Events in Apache Kafka are encrypted as well. Pub\Sub API encrypts messages while in transit.

5. Governor Limit Considerations and License Cost of Salesforce Platform

5.1. Governor Limits

As stated, there are many governor limits on the Salesforce multi-tenant platform which we need to consider during solutioning. Some of the common limits that we might hit during the code execution more frequently and how this solution overcomes it are stated below. These limits cannot be extended at any cost. So, it's very important that we don't run into any issues during the implementation and need to make sure the solutions are scalable.

Using Platform events and Async triggers for transformation logic will enhance the limits as the execution chain breaks by publishing the platform event or async trigger and will have its own execution limits. Also using this approach, integrations won't lock the sObject records while it performs the DML operations, and the long running transactions run at platform event level and won't cause any record row-lock issues on the main sObject. Platform Events can process up to 2000 records in execution, so the events are drained much faster.

5.2. License Cost (Add-on for Salesforce Only)

Platform Event and Change Data capture events that are subscribed by external applications (in this case Camel connector) have a contractual limit. These can be purchased additional as needed based on the demand.

Table 1. Salesforce Governor Limits

S.No	Governor Limit Name	Limit
1	Total Number of SOQL Queries	100
2	Total Number of DML Statements	150
3	Max CPU Time on SF Servers	10000 milliseconds
4	Number of synchronous concurrent transactions for long-running transactions that last longer than 5 seconds for each org	10
5	Total number of records retrieved by SOQL queries	50000
6	Total Heap Size	6MB

Table 2. Add-on Licenses needs. (for unlimited edition)

Description	Type	Default Limit	Add-on limits
Event Delivery to Pub\Sub API or CometD clients	Platform Event	50000 per day	Per add-on additional 100000 events per day
Event Publishing	Platform Event	250000K per hour	Per add-on additional 25000 events per hour
Event Delivery to Pub\Sub API or CometD clients	Change Data Capture(CDC)	50000 per day	Per add-on additional 100000 events per day
Event Publishing	Change Data Capture(CDC)	No limit	Per add-on additional 25000 events per hour

6. Conclusions

As more and more systems get connected, the more real-time events increase and the solution of handling millions of real-time events using packaged platforms like Salesforce sometimes can become super complex. The part of handling the complex transformation logic within the Salesforce platform and publishing or consuming events at a much faster phase is outlined in this article using an open-source connector but not by a middleware solution. Using middleware solution instead of a connector could ease the solution but there will be an associated cost to it in terms of both license and maintenance. This solution by using platform events and or Change Data Capture(CDC) can be scaled to consume or publish millions of events per day if needed.

REFERENCES

[1] https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_intro_what_is_apex.htm

[2] https://developer.salesforce.com/docs/platform/pub-sub-api/overview

[3] https://developer.salesforce.com/docs/atlas.en-us.platform_events.meta/platform_events/platform_events_intro.htm

[4] https://developer.salesforce.com/docs/atlas.en-us.change_data_capture.meta/change_data_capture/cdc_intro.htm

[5] https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_gov_limits.htm

[6] https://developer.salesforce.com/docs/atlas.en-us.248.0.change_data_capture.meta/change_data_capture/cdc_allocations.htm

[7] https://developer.salesforce.com/docs/atlas.en-us.platform_events.meta/platform_events/platform_event_limits.htm.

[8] https://camel.apache.org/camel-kafka-connector/4.0.x/reference/connectors/camel-salesforce-source-kafka-source-connector.html

[9] https://camel.apache.org/camel-kafka-connector/4.0.x/reference/connectors/camel-salesforce-update-sink-kafka-sink-connector.html

[10] https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/DataWeaveInApex.htm

[11] https://cloud.google.com/learn/what-is-apache-kafka#section-4

[12] https://help.salesforce.com/s/articleView?id=release-notes.rn_platform_events_encryption.htm&release=224&type=5