**Technology choice #1 — Webhooks**

If you are building an event-driven API, perhaps the most straightforward way to get started is to allow your consumers to register a Webhook to receive event notifications from your API.

A Webhook is a publicly accessible HTTP POST endpoint managed by an event consumer. An event producer, such as an API server, can send event notifications to a webhook when something interesting happens.

You can think of Webhooks as reverse APIs. They completely detach the HTTP request and response from each other.

As an API provider, you have to consider two things when building event-driven APIs for Webhooks-enabled consumers.

**1. Allow API consumers to subscribe to your APIs**

Consumers can subscribe to your API by registering a Webhook URL as the callback.

The Webhook subscription itself can be managed as a REST resource, and it must include the list of event types and the subscription(callback) URL at a minimum level. It should also offer a way to unsubscribe from the API.

**2. Deliver events to API consumers asynchronously**

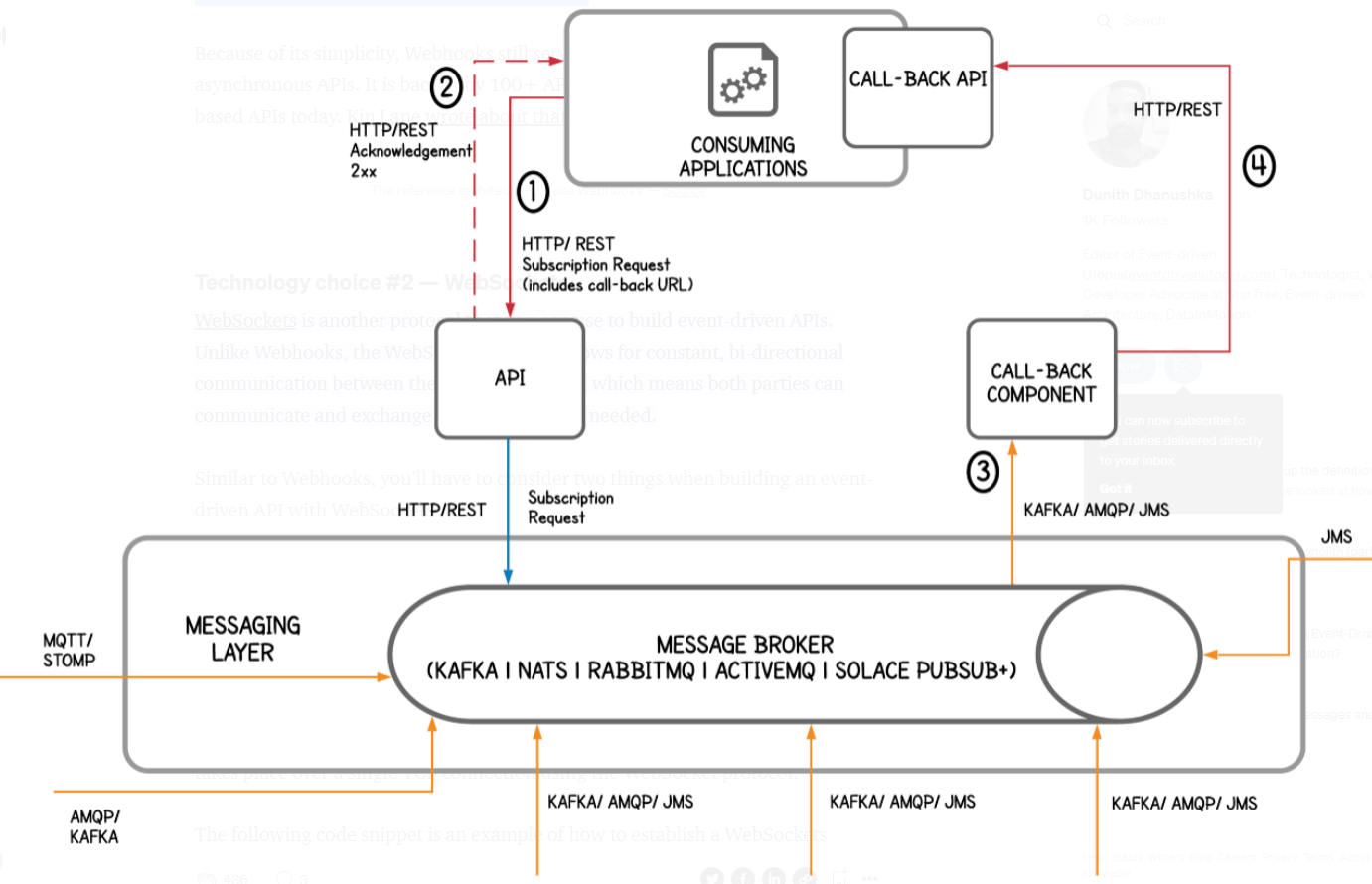
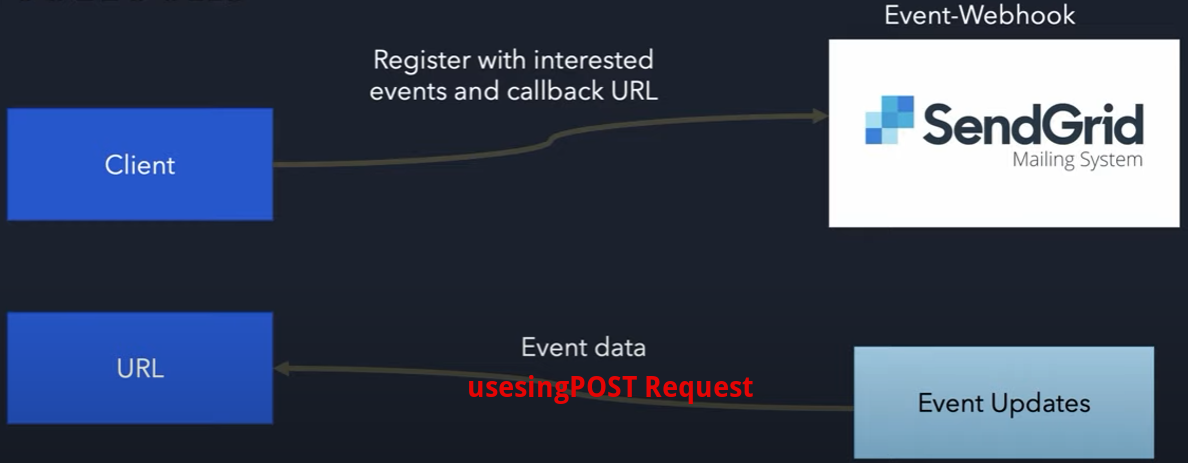
After sorting out the subscription, your next task is to deliver events to the consumers.

As the API provider, you can make an HTTP POST call to the consumer’s Webhook when something interesting happens at the backend. For example, a database record has been updated.

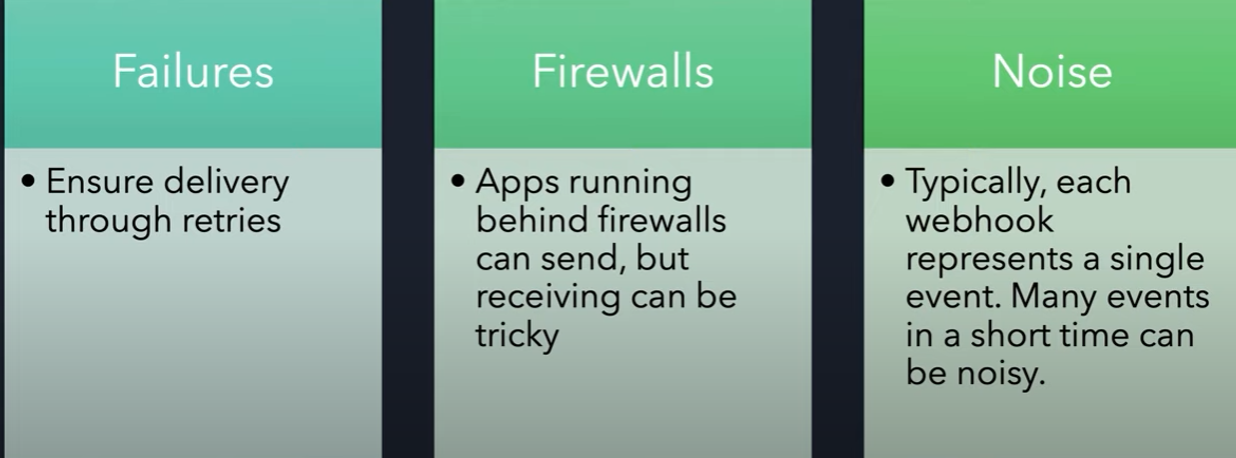
Webhooks forces the event consumer to establish a publicly accessible HTTP endpoint to receive events. It comes with other concerns like securing it with certificates, preventing DDoS attacks, etc. These could be a burden in the long run in terms of maintenance.

Moreover, Webhooks can’t be used to push event notifications to end-user consumers such as mobile and Single Page Applications (SPA) as they can not in possession of an HTTP endpoint.

Despite the above drawbacks, Webhooks can still be ideal for implementing a server-to-server event notification mechanism.

****

**Overview**

****

Клиент по сути начинает выступать в роли сервера, предоставляя другому приложению какой то адрес, куда тот может отправлять запросы. В catapulto вебхуки используют что бы дополучить данные по отправке (накладные, наклейки и тд), так как время когда они будут созданы не известно, и что бы не слать бесконечные запросы узнавая а создались ли уже накладные или нет, мы ждем того момента когда они будут созданы и сервер уже сам отправляет нам накладные по вебхукам на наш **callback url**

Вебхуки – это не замена SOAP или REST. Мы можем общаться со сторонней системой по SOAP протоколу, но дополучать какие то данные по вебхукам