



Webhooks

In this lesson, you'll learn about the need for webhooks and how they work.

We'll cover the following ^

- What are Webhooks?
- How do webhooks work?

What are Webhooks?#

Imagine you've written an *API* that provides information on the latest, most exclusive *Baseball* events. Now your *API* is consumed by a lot of third-party services that fetch the information from the API, add their own flavor to it, and present it to their users.

However, so many API requests every now and then, just to check if a particular event has occurred is crushing your server. The server can hardly keep up with the requests. There is no way for consumers to know that the new information isn't available on the server yet or that an event hasn't occurred yet. They just keep polling the API. This would eventually pile up the unwanted load onto the server and could bring it down.

What do we do? Is there a way we can cut down the load on our servers?

Yes!! *Webhooks*.

Webhooks are more like *call-backs*. It's like, "I will call you when new information is available. You carry on with your work."

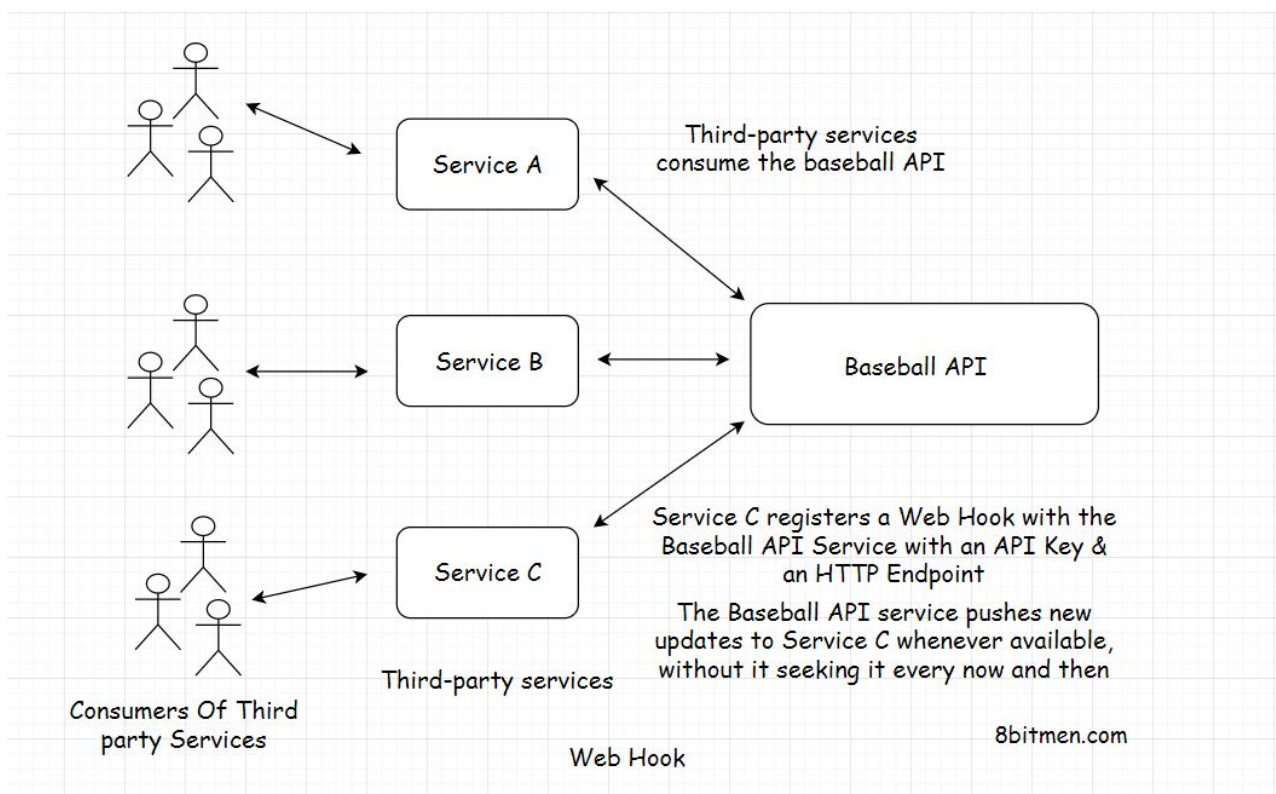
Webhooks enable communication between two services without the middleware. They have an *event-based* mechanism.

So, how do they work?

How do webhooks work?#


To use the *Webhooks*, consumers register an *HTTP* endpoint with the service with a unique *API Key*. It's like a phone number. Call me on this number, when an event occurs. I won't call you anymore.


Whenever new information is available on the backend, the server fires an *HTTP* event to all the registered endpoints of the consumers, notifying them of the new update.




Browser notifications are a good example of *Webhooks*. Instead of visiting the websites every now and then for new info, the websites notify us when they publish new content.

[← Back](#)



Next 



Event-Driven Architecture - Part 2

Shared-Nothing Architecture

☒ Mark as Completed

 Report an Issue